UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Sistemas de Informação
Guilherme Rocha Leite

DESENVOLVIMENTO DE UM SISTEMA WEB PARA A ASSOCIAÇÃO DOS PROTETORES DA BACIA HIDROGRÁFICA DO RIO GORUTUBA "KURUATUBA"

Diamantina
2020

Guilherme Rocha Leite

DESENVOLVIMENTO DE UM SISTEMA WEB PARA A ASSOCIAÇÃO DOS PROTETORES DA BACIA HIDROGRÁFICA DO RIO GORUTUBA "KURUATUBA"

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, como parte dos requisitos exigidos para a conclusão do curso.

Orientador: Erinaldo Barbosa da Silva

Coorientador: Thales Francisco Mota Carvalho

Diamantina

2020

Guilherme Rocha Leite

DESENVOLVIMENTO DE UM SISTEMA WEB PARA A ASSOCIAÇÃO DOS PROTETORES DA BACIA HIDROGRÁFICA DO RIO GORUTUBA "KURUATUBA"

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, como parte dos requisitos exigidos para a conclusão do curso.

Orientador: Prof. Erinaldo Barbosa da Silva
Coorientador: Prof. Thales Francisco Mota Carvalho

Data de aprovação ____/____/____.

Prof. Erinaldo Barbosa da Silva
Departamento de Computação – UFVJM

Prof. Thales Francisco Mota Carvalho
Diretoria de Comunicação Social – UFVJM

Prof. Áthila Rocha Trindade
Departamento de Computação – UFVJM

Diamantina

A todos aqueles que contribuem para um mundo melhor.

AGRADECIMENTOS

A Deus, pela minha felicidade e saúde, e a familiares e amigos, pelos relacionamentos e amizades.

Se meus olhos mostrassem a minha alma, todos, ao me verem sorrindo, chorariam comigo. (Kurt Cobain).

RESUMO

O trabalho abordado neste documento teve por finalidade empregar conteúdos estudados durante o curso de Sistemas de Informação e tecnologias recentes de desenvolvimento de aplicações web utilizadas em órgãos nacionais e internacionais, na construção de um sítio eletrônico para a associação de protetores da bacia hidrográfica do Rio Gorutuba de Janaúba/MG (Kuruatuba), localizada no norte de Minas Gerais. A Kuruatuba é uma sociedade civil sem fins lucrativos que desempenha importantes funções não só na preservação da Bacia do Rio Gorutuba, como também na realização de atividades comunitárias para promoção de bem estar social para a comunidade em geral. O trabalho objetiva-se agregar mais visibilidade e notoriedade à associação e permitir a mobilização da comunidade local para as ações realizadas pela Kuruatuba e sua importância. Entre as ferramentas usadas no desenvolvimento do software, podemos destacar as seguintes: *Docker, Plone, Google Analytics, Apache JMeter* e *Git*. No decorrer do trabalho foram apresentados e justificados os seus reais objetivos, e, ao final, as metas alcançadas e propostas para uma possível continuação do projeto, após a obtenção dos resultados .

Palavras-chave: Engenharia de Software. Engenharia Web. *Docker*. *Plone*. *Scrum*.

ABSTRACT

Com as mesmas características e conteúdo do resumo em língua portuguesa, porém, deve ser escrito em língua inglesa.

Keywords: Word1. Word2. Word3. Word4. Word5.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

Atualmente os sistemas web fazem parte do cotidiano das pessoas que estão cada vez mais conectadas a um mundo virtual, também denominado internet, fundamentalmente sustentado pela comunicação com ou entre seus usuários. Deste modo, qualquer conteúdo existente nele tem um significado que deve ser interpretado por um ser humano, ou seja, havendo a transmissão de informação e pressupondo um processo de comunicação Araújo e Freire (2012). Para manter a interação entre as pessoas, existem as aplicações web, por exemplo as redes sociais, manipuladas diariamente por grande parte dos usuários interativos da internet, também conhecidos como internautas.

Existem diversos tipos de aplicações para a internet, Gonçalves *et al.* (2005) e Almeida (2003) citam como exemplares as páginas para a web, aplicações *E-business*, aplicações de comércio eletrônico, a educação on-line e a *E-Learning* (aprendizagem eletrônica). Antes de dar prosseguimento ao conteúdo, é interessante explicar a diferença entre os conceitos de *website* e sistemas web, pois são temas que muitos, inclusive programadores, consideram ter o mesmo significado.

Um *website* é basicamente um conjunto de páginas com caráter meramente expositivo, não sendo possível ocorrer consultas em bancos de dados. As páginas que contém conteúdo como informações para contato, missão e valores, e história são exemplos. Já nas aplicações ou sistemas web, ocorre a interação entre o software e o usuário, seja por formulários de cadastro, consultas a informações contidas no bancos de dados ou outros recursos específicos de cada tipo de aplicação. Segundo Garrett *et al.* (2005), o modelo clássico para aplicações web funciona da seguinte maneira: ações do usuário pela interface do sistema acionam uma solicitação para o servidor, que processa dados e informações retornando-os para o cliente (usuário) através de uma página HTML (*Hypertext Markup Language*).

Em algumas instituições é comum perceber que alguns ou vários processos administrativos ainda são feitos manualmente, sem uso de sistemas informatizados que auxiliem em seu desenvolvimento. Esse é um fato preocupante, porque tarefas básicas e rotineiras, que poderiam ser realizadas de forma mais rápida e eficaz por sistemas informatizados, acabam consumindo mais tempo para serem executadas e ainda tenderão a erros humanos que possam estar ocorrendo durante ou após o processo. (OTHMAN; ISMAIL; RAUS, 2009).

> Indústrias como fabricação, viagens e hospitalidade, bancos, educação e governo estão habilitados na web para melhorar e aprimorar suas operações. O comércio eletrônico expandiu-se rapidamente, atravessando fronteiras nacionais. Até os sistemas tradicionais de informações e bancos de dados herdados migraram para a Web. (GINIGE; MURUGESAN, 2001, p. 1).

Diante do cenário tecnológico global e da influência dos sistemas integrados a internet na vida das pessoas, manifestou-se a oportunidade de criação de um software, que

agregado às características típicas de aplicações web, conseguisse expandir seus benefícios para o dia-a-dia dos membros e desfrutadores das ações desempenhadas pela associação Kuruatuba.

1.2 Sobre a Kuruatuba

Em seu blog, a Kuruatuba (2011) discorre sobre seu histórico. Segundo o texto, a história da Kuruatuba está atrelada à utilização das areias da praia do Copo Sujo, localizada no município de Janaúba-MG, para prática de esportes juntamente com o combate às diversas ameaças ao meio ambiente ocorridas no ano de 1988. A partir de 1989, ano marcado pela realização do evento "Carnaval 40º Graus", que teve ampla repercussão e adesão pela comunidade local com o objetivo de valorização do ambiente, a praia foi zelada por um grupo de atletas de vôlei e futebol de areia, que em 1998 se uniram e fundaram a Associação de Futebol de Praia do Copo Sujo de Janaúba (AFPJ).

A AFPJ foi parceira de diversos órgãos e instituições durante a conservação da praia do Copo Sujo. Trabalhos foram desenvolvidos com apoio da Secretaria de Meio Ambiente da Prefeitura de Janaúba e Ruralminas, IEF, CODEMA, Poder Judiciário (Albergados), escolas, igrejas e diversos segmentos da sociedade. Em 2003, com uma evidência mais ambientalista, mesmo ainda empenhando-se na atividades de preservação do Rio Gorutuba, a AFPJ passou a ser intitulada KURUATUBA, que significa "sapo grande", homenagem ao rio.

Em novembro de 2003 foi assinado o estatuto de regimento da KURUATUBA - ASSOCIAÇÃO DOS PROTETORES DA BACIA HIDROGRÁFICA DO RIO GORUTUBA DE JANAÚBA-MG contendo 45 artigos que apresentam valores, normas, objetivos e diversos outros esclarecimentos sobre a mesma. Ainda segundo a (KURUATUBA, 2011), o objetivo da associação "é promover o esporte, lazer, cultura e preservação e conservação da Bacia do Rio Gorutuba, sendo sua área de atuação compreendida da nascente à foz, incluindo os seus afluentes."

1.3 Justificativa

A ideia para o projeto partiu da própria associação, presidida pelo professor Erinaldo Barbosa da Silva. Na época, a instituição passava por dificuldades para ampliar o número de apoiadores e associados, além de limitações ao publicar balanços patrimoniais, visto que não possuíam um meio "formal" para tal, dificultando até no recebimento de verbas por parte do Estado.

Em pesquisa realizada entre apoiadores e associados, cujo formulário está situado no apêndice A, apenas 7,1% dos participantes recebem notícias referentes à associação via site ou blog, e 56,3% gostariam de receber tais notícias por esses veículos. Com isso, renova-se a ideia de criar uma aplicação onde notícias e eventos possam ser cadastrados e publicados por meio eficiente para disseminação.

Com a intenção de reduzir as dificuldades mencionadas e criar um ambiente destinado a suprir as principais necessidades de associados, membros da gestão institucional, colaboradores e demais públicos, o projeto fora aprovado, dando início a um período de significante aprendizado e dedicação.

1.4 Objetivos

Nesta seção, serão apresentados os objetivos gerais e específicos que regem todo o projeto.

O objetivo geral pode ser compreendido na construção de um sistema eletrônico capaz de auxiliar na gestão das informações pertencentes à associação Kuruatuba, também possibilitando a divulgação de informes e a propagação de seus ideais de maneira mais rápida e abrangente.

Considerando-se, agora, os objetivos específicos escolhidos, encontram-se os listados abaixo:

- Oferecer aos seus usuários e visitantes as principais funcionalidades para a manutenção, segurança e disponibilidade das informações.
- Promover a divulgação de notas, informes e eventos organizados pela associação Kuruatuba.
- Aprofundar os estudos sobre temas relacionados a Engenharia de Software e segurança de dados.
- Possibilitar o gerenciamento de pessoas associadas à organização, por meio de sistema com interface agradável e opção de emissão de documento comprobatório de associativismo para os cadastrados.
- Tornar o sistema responsivo, sendo possível acessá-lo em diferentes dispositivos físicos com qualidade equivalente à de *desktops*.

2 REFERENCIAL TEÓRICO

Este capítulo objetiva a apresentar os principais conceitos que serão utilizados na metodologia. Tais conceitos são referentes ao desenvolvimento web e serão cruciais para o entendimento do trabalho essencialmente porque tais informações servirão de bases fundamentais para o disposto no capítulo 3.

Na seção 2.1 será revisado o significado de Engenharia Web e sua importância. Na seção 2.2 o conteúdo terá enfoque nas metodologias ágeis, ocorrendo, posteriormente, uma breve comparação entre as mais conhecidas. A 2.3, por sua vez, irá explanar sobre a utilização de Sistemas Gerenciadores de Conteúdo na construção de aplicações na internet. A seção 2.4 fará uma sucinto resumo sobre os Servidores Virtuais Privados e os benefícios do seu emprego. Ainda na revisão bibliográfica, mais precisamente na seção 2.5, serão dispostos alguns dos tipos de testes de software e seu destaque no desenvolvimento dos mesmos. Por fim, conceitos referentes às ferramentas e tecnologias empregadas entre os materiais e métodos do trabalho serão tratados na seção 2.6, a fim de proporcionar ao leitor uma compreensão antecipada sobre tais assuntos.

2.1 Engenharia Web

Ginige e Murugesan (2001) classificam a Engenharia Web como um conjunto de técnicas que propõe o estabelecimento de informações científicas para desenvolver princípios e abordagens disciplinadas para a conquista do sucesso no desenvolvimento.

Para facilitar o entendimento, pode-se concluir que a Engenharia Web se caracteriza pela estipulação de métodos e procedimentos que podem ser adotados por toda a equipe para auxiliar no desenvolvimento do projeto de aplicações web.

As metodologias de desenvolvimento de software, estudadas em subáreas da engenharia de software, podem ser adotadas para organizar todo o projeto de desenvolvimento de um software. Elas podem ser divididas entre tradicionais e ágeis, e são operadas para auxiliar na produção de software em processos como a análise de requisitos e a codificação, como mencionado por Soares (2004). Sommerville (2003 apud SOARES, 2004, p. 1) define quatro atividades comuns a todos os processos de desenvolvimento:

- **Especificação de Software**: etapa em que ocorre um contato maior da equipe com os clientes, com o intuito de determinar os requisitos e funcionalidades do programa. Para tal, existem diversas formas de coletar essas informações, como por exemplo realização de entrevistas, questionários, observação naturalista, entre outras.
- **Projeto e Implementação de Software**: etapa de construção de diagramas para gerar modelos que serão posteriormente implementados.
- **Validação de Software**: é a fase de verificação dos requisitos. Nesta etapa acontece uma análise de requisitos atendidos e não atendidos.

- **Evolução do Software**: diferentemente das demais, esta fase é posterior à entrega do produto ao cliente. Ela corresponde basicamente às manutenções e atualizações de software que serão produzidas com o passar do tempo para que ele possa continuar sendo utilizado pelos usuários.

2.2 Metodologias Ágeis

Os métodos ágeis surgiram na década de 90 para alterar a mentalidade das equipes de desenvolvimento da época, contradizendo os métodos tradicionais eliminando gasto com documentação excessiva, enfatizando a comunicação e aumentando a proximidade com o cliente para produzir software de qualidade. Sato (2007).

Segundo Conforto (2009), a assinatura do manifesto ágil, disponível em: http://agilemanifesto.org/, foi um marco para o nascimento das Metodologias Ágeis. O manifesto se trata de um documento em que a eficácia das metodologias tradicionais é questionada em cenários que envolvam incertezas e constantes mudanças no ambiente empresarial.

É importante ressaltar que tais metodologias podem ser aplicadas tanto em equipes menores, de desenvolvimento, por exemplo, quanto em projetos inteiros através do Gerenciamento Ágil de Projetos (GAP), constituídos por várias equipes. Também é interessante saber que um dos fundamentos do processo ágil é a construção de ciclos, cujo objetivo é adaptar e avaliar constantemente o produto antes que chegue no estado final, objetivando atribuí-lo qualidade perante os requisitos estipulados e a satisfação do cliente.

A título de conhecimento dos diferentes métodos ágeis, o artigo de Sato (2007) emerge sobre os seguintes métodos:

- ***Scrum***: desenvolvido nas décadas de 80 e 90. Este modelo propõe interações (*sprints*) de 2 semanas a um mês com acompanhamento de reuniões em pé. O líder da equipe é denominado *Scrum master* e as tarefas e funcionalidades, que são incluídas em uma lista, são chamadas de "*Product Backlog*";
- ***Lean Software Development***: baseado no sistema de produção da *Toyota*, este método revolucionou a manufatura, o desenvolvimento de produtos e o gerenciamento da cadeia de suprimentos. Sustentado sobre sete princípios: "Elimine Desperdícios", "Inclua a Qualidade no Processo", "Crie Conhecimento", "Adie comprometimentos", "Entregue rápido", "Respeite as pessoas" e "Otimize o Todo" (SATO, 2007);
- **Metodologia da família *Crystal***: possui uma peculiaridade, a "habitabilidade", que significa "o mínimo de processo necessário para que a equipe consiga ter sucesso" (SATO, 2007). Além disso, seus métodos compartilham propriedades como: entrega frequente, reflexão e comunicação;
- ***Feature Driven Development* (FDD)**: criada na década de 90, ele se aproveita de diagramas UML para representar classes de diferentes responsabilidades e é constituído por

duas fases: a de concepção e planejamento e a iterativa de construção, compostas por 5 processos internos ao todo.

- ***Adaptive Software Development***: método que propõe três fases possivelmente sobrepostas: especulação, colaboração e aprendizado. Aqui, o conhecimento sobre falhas, oriundas de falsas premissas, é adquirido por meio de curtas iterações, sendo corrigidas vagarosamente.
- ***Dynamic systems development method*** **(DSDM)**: seu desenvolvimento se iniciou em 1994 e possui duas fases primárias, uma de viabilidade, cujo objetivo é verificar se é viável a execução deste método na situação decorrente, e um estudo de negócio para definir os requisitos iniciais e a arquitetura de software. No decorrer do processo ainda são previstas etapas para prototipação, construção do sistema e entrega do produto. O DSDM ainda possui princípios básicos relacionados à frequência de entregas e de testes, e participação direta do usuário durante todo o processo.
- ***Extreme programming*** **(XP)**: juntamente com o *Scrum* é o mais utilizado. Busca "a excelência no desenvolvimento de software, visando baixo custo, poucos defeitos, alta produtividade e alto retorno de investimento" (SATO, 2007).

Concluindo o tema, é aconselhável exibir as diferenças entre os dois tipos de metodologias, para isso os quadros 1 e 2 foram criadas.

Quadro 1 – Diferenças básicas entre as metodologias tradicionais e ágeis - Gerenciamento Tradicional

Características	Gerenciamento Tradicional
Ter definido a priori	Escopo
Responsável pela organização para atingir os objetivos do projeto	Gerente de projeto
Frequência de reuniões de status	Dependendo da complexidade/necessidade do projeto
Escopo	Bem definido nas fases iniciais do projeto
Tempo	Cronograma detalhado para realização de todo o projeto
Custo	Monitoração das alterações para que não altere o custo planejado
Qualidade	Processo de verificação, validação e plano de testes
Riscos	Análise de riscos durante todo o ciclo de vida do projeto
Comunicação	Documentação e formal

Fonte: Silva, Souza e Camargo (2013). Adaptado.

Quadro 2 – Diferenças básicas entre as metodologias tradicionais e ágeis - Gerenciamento Ágil

Características	Gerenciamento Ágil
Ter definido a priori	Tempo (*sprints*)
Responsável pela organização para atingir os objetivos do projeto	*Scrum master*
Frequência de reuniões de status	Diárias
Escopo	Definido em alto nível e os requisitos definidos de forma iterativa
Tempo	Cronograma orientado a produto com entregas incrementais
Custo	Maior controle em função da rapidez em alterações
Qualidade	Programação em pares, testes incrementais e refatoração
Riscos	Aplica-se o mesmo conceito do gerenciamento tradicional
Comunicação	Implícita, interpessoal e colaborativa

Fonte: Silva, Souza e Camargo (2013). Adaptado.

Segundo conteúdo publicado no diário Dingsøyr *et al.* (2012) as duas metodologias ágeis mais comuns em equipes de desenvolvimento de software são o *Extreme Programming* e o *Scrum*, ambas estão diferenciadas na tabela 1.

Tabela 1 – Comparação entre o XP e o *Scrum*

Recursos	***Extreme Programming***	***Scrum***
Tamanho do projeto	Pequeno	Todos
Tamanho das equipes (pessoas)	2 a 10	Menores que 10
Duração das *sprints* (semanas)	1 a 3	4
Elicitação de requisitos	Histórias de usuário	Não definido
Mudanças durante a iteração	Permitido	Não permitido
Ordem de desenvolvimento definida por	Cliente	Equipe *Scrum*
Envolvimento do *stakeholder*	Durante o processo	Não definido

Fonte: Anwer *et al.* (2017). Adaptado.

2.3 Sistemas Gerenciadores de Conteúdo (CMS)

O *Content Management System* (CMS), ou Sistema de Gerenciamento de Conteúdo (SGC), em português, surgiu, segundo (CHAGAS; CARVALHO; SILVA, 2018), no final da década de 90 com o intuito de melhorar a gestão do conteúdo dos *websites* das organizações da época.

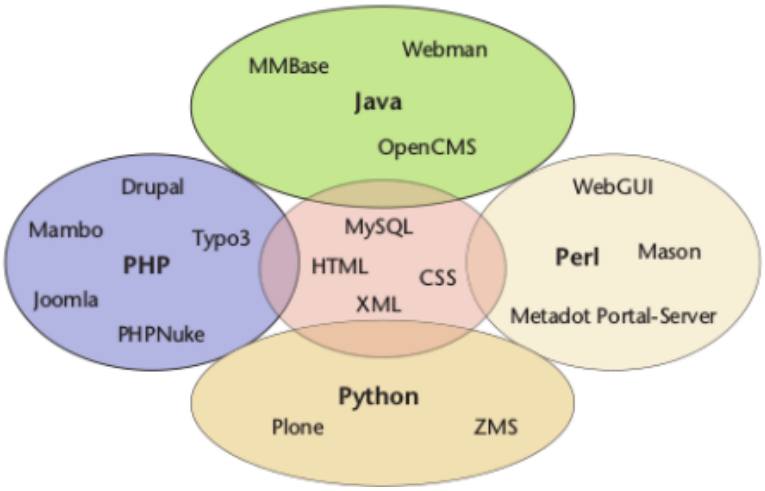
Meike, Sametinger e Wiesauer (2009) e Chagas, Carvalho e Silva (2018) citam qualidades proporcionadas aos desenvolvedores que os utilizam. Segundo eles as organizações podem utilizar um gerenciador de conteúdo para construir *websites*, lojas on-line ou portais, havendo redução de erros de publicação que facilitam o processo de validação e sendo manuseados por muitas pessoas sem que seja necessário editar o código-fonte e possuir conhecimento especializado na área de programação.

Outro fator positivo encontrado nesses sistemas é a possibilidade de colaboração entre os desenvolvedores. Porque uma aplicação web desenvolvida em um CMS pode ser gerenciada por diversas pessoas, podendo ter restrições de acesso de cada uma delas a diferentes partes do sistema, ou seja, é possível definir níveis de acesso a usuários para que cada um deles possa administrar um ou mais tipos de conteúdo do mesmo. Para sintetizar, (BOIKO, 2001 apud CHAGAS; CARVALHO; SILVA, 2018, p. 1) conclui que "um SGC possibilita a criação, o gerenciamento, a distribuição, a publicação e a recuperação de informações corporativas".

Esses sistemas possuem características próprias que variam de acordo com o tipo de gerenciador. Existem CMSs baseados nas linguagens Python, Java, PHP e Perl, sendo aspectos disjuntos aos sistemas, porém todos eles são capazes de usar linguagens padrão da web e bancos de dados relacionais, como visto na figura 1.

As próximas subseções foram dedicadas a gerar uma maior compreensão sobre os Sistemas Gerenciadores de Conteúdo e sua utilização em âmbito institucional. Através delas será possível assimilar os diferentes tipos de gerenciadores e seus principais atributos, estabelecendo

Figura 1 – Sistemas de gerenciamento de conteúdo da web de código aberto e tecnologias relacionadas



Fonte: Meike, Sametinger e Wiesauer (2009).

ainda um comparativo entre eles no que diz respeito à suas características e utilização em meio corporativo.

2.3.1 Tipos de Sistemas Gerenciadores de Conteúdo

É possível encontrar centenas de ferramentas de gerenciamento de conteúdo hoje em dia, sendo que, as questões que os diferem são diversas, citando-se requisitos do sistema, segurança, suporte, facilidade de uso e desempenho. SILVEIRA (2010).

No relatório *Open Source CMS Market Share Report* divulgado pela agência digital Stone (2011), foi possível obter a correlação entre os diferentes CMSs com os principais sites encontrados no *Google PageRank*. O *Google PageRank* é "um método usado pela *Google* para determinar a importância ou a relevância de uma página. Páginas relevantes têm altos escalões e provavelmente aparecerão no topo nos resultados de pesquisa" (MIRDHA; JAIN; SHAH, 2014, p. 3, tradução nossa).

A relevância das páginas é proporcional ao ranking e indica que quanto maior ele seja, mais chance a página tem de aparecer no topo da lista dos resultados de uma pesquisa do mecanismo de busca da *Google*.

Por conveniência, os quatro sistemas mais bem colocados serão os escolhidos para dar início ao processo de comparação entre os principais. Tal processo se encontra na subseção 2.3.2, desenvolvida adiante. Porém, antes de iniciá-la, é importante haver uma apresentação dos sistemas escolhidos para deixar claro as distinções básicas entre eles antes de adentrar em seus detalhes e especificidades.

Drupal: o CMS é derivado de um projeto escrito por um universitário holandês que tinha como objetivo fornecer meios de compartilhar notícias e eventos. Menezes *et al.* (2016).

Tabela 2 – Ranking das páginas por CMS

Content Management System	*Page Rank*
Joomla!	9
Drupal	9
WordPress	9
Plone	9
Typo3	8
Concrete5	7
DotNetNuke	7
Alfresco	7

Fonte: Stone (2011 apud MIRDHA; JAIN; SHAH, 2014). Adaptado.

Seu projeto foi considerado *open source* (código aberto) em 2001 e baseou-se na linguagem PHP. Para empregá-lo é necessário possui uma máquina que suporte um servidor web do PHP com versão de 5.2 ou superior, exemplos: *Apache, nginx* e *IIS*, além de um banco de dados, exemplos: *MySQL, SQLite* e *PostgreSQL* Tomlinson (2010).

Joomla!: segundo Menezes *et al.* (2016) o *Joomla!* foi desenvolvido em 2005 após a separação entre a equipe de desenvolvedores do *Mambo*; um gerenciador de conteúdo de software livre e código aberto, desenvolvido em PHP e que utiliza o *MySQL* como banco de dados SILVEIRA (2010); e a empresa *Miro*. Patel, Rathod e Prajapati (2011), em seu artigo, salienta que sua aplicação é recomendada para criação de web sites em um curto tempo, dividindo a preferência dos usuários com o *WordPress* no quesito de criação de portais, blogs e aplicações *E-commerce*.

Plone: o Plone é um CMS usado amplamente e principalmente por órgãos do governo. É livre e de código aberto, sendo executado sobre o *Zope*, um sistema operacional construído em *Python* para aplicações web, e utiliza o banco de dados *ZODB*. O projeto *Plone* teve início no ano de 1999 e sua primeira versão foi lançada em 2001 Menezes *et al.* (2016), Alves (2017) e SILVEIRA (2010).

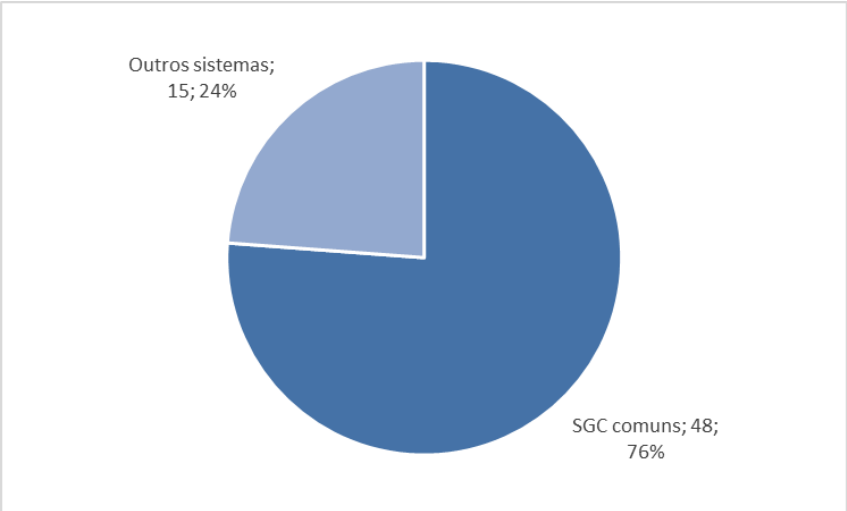
WordPress: o *WordPress* é um CMS disponibilizado em 2003 que tem seu uso facilitado por não exigir ao desenvolvedor qualquer conhecimento específico na área de programação. Por conta de sua intuitividade, comunicabilidade e usabilidade, ele é usado atualmente em milhões de sites, assim como mantido por uma grande comunidade de desenvolvedores e usuários. Vale a pena ressaltar que ele consiste em um sistema extremamente personalizável e completo, visto que possui milhares de *widgets, plugins* e *temas* que podem ser manipulados de acordo com as necessidades do usuário Oliveira *et al.* (2017). Para concluir, ele é desenvolvido em PHP, assim como o *Drupal* e o *Joomla!*, e integrado ao banco de dados *MySQL*.

2.3.2 Comparativos entre os principais sistemas

Nesta subseção, o âmago girará em torno de comparativos entre os principais Sistemas de Gerenciamento de Conteúdo em relação à sua aplicação em instituições públicas e no mercado em geral, e aos ganhos de cada um em relação a facilidade de uso, desempenho, flexibilidade e itens de suporte. Inicialmente a pesquisa abordará as instituições públicas brasileiras, representadas por universidades federais, encerrando-se com uma análise no cenário mundial, destacando seus pontos fortes, fracos e sua utilização no mercado.

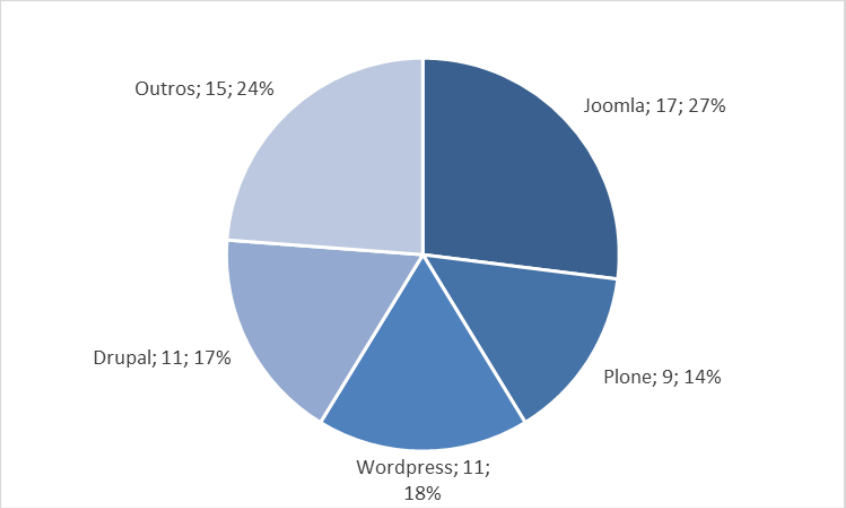
Baseando-se nas 63 universidades brasileiras pesquisadas no trabalho de Alves (2017), chega-se à conclusão, a partir do demonstrado na figura 2, que a maioria delas adotaram Sistemas Gerenciadores de Conteúdo como ferramenta para desenvolvimento de seus respectivos portais institucionais. Diante do exposto na figura 3, obtemos que entre nessas universidades há a predominância dos quatro principais sistemas constatados na subseção anterior, sendo eles: *Drupal, Joomla!, WordPress* e *Plone*.

Figura 2 – Taxa de opção por CMS entre as universidades públicas federais



Fonte: Alves (2017).

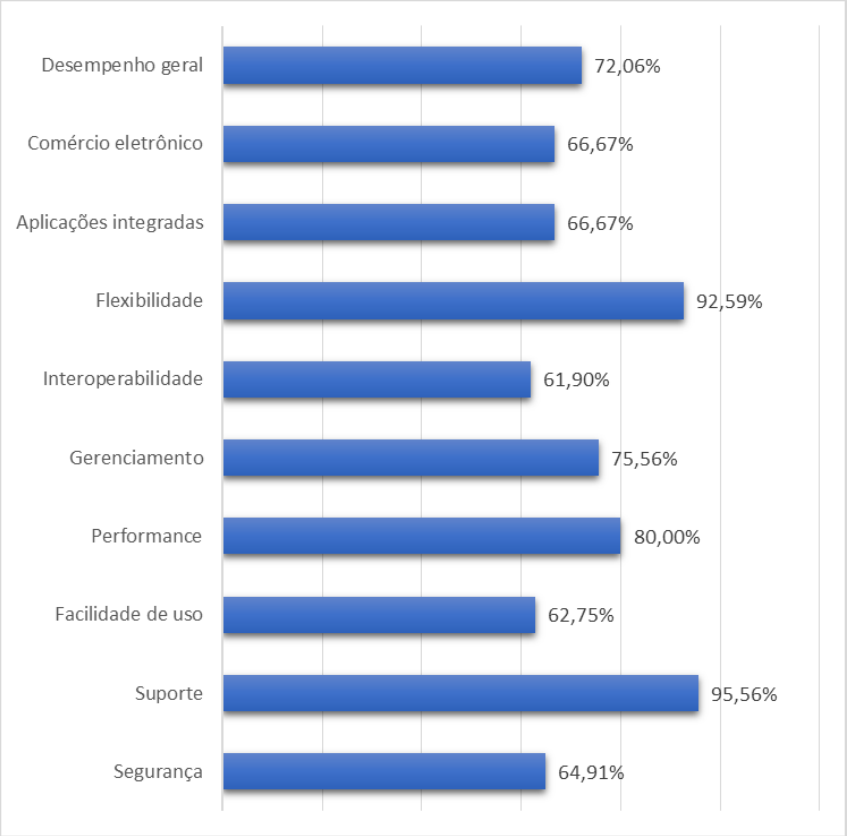
Figura 3 – Tipos de CMSs adotados pelas universidades federais



Fonte: Alves (2017).

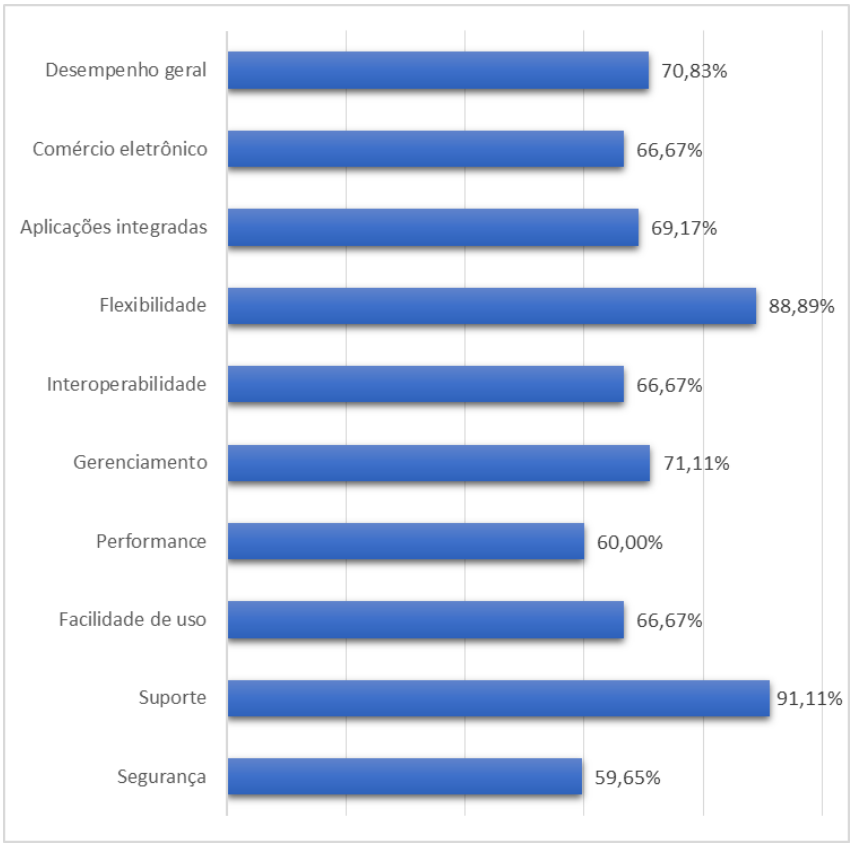
Após as observações anteriores, serão comparados os quatro sistemas predominantes com o objetivo de avaliá-los e determinar qual deles possui o melhor resultado geral diante das seguintes métricas: desempenho geral, comércio eletrônico, aplicações integradas, flexibilidade, interoperabilidade, gerenciamento, performance, facilidade de uso, suporte e segurança. As figuras 4, 5, 6 e 7 referem-se ao desempenho obtido pelo respectivo sistema para cada uma das métricas propostas. Por fim, a figura 8 demonstra o desempenho geral obtido por cada um.

Figura 4 – Desempenho obtido pelo *Drupal*



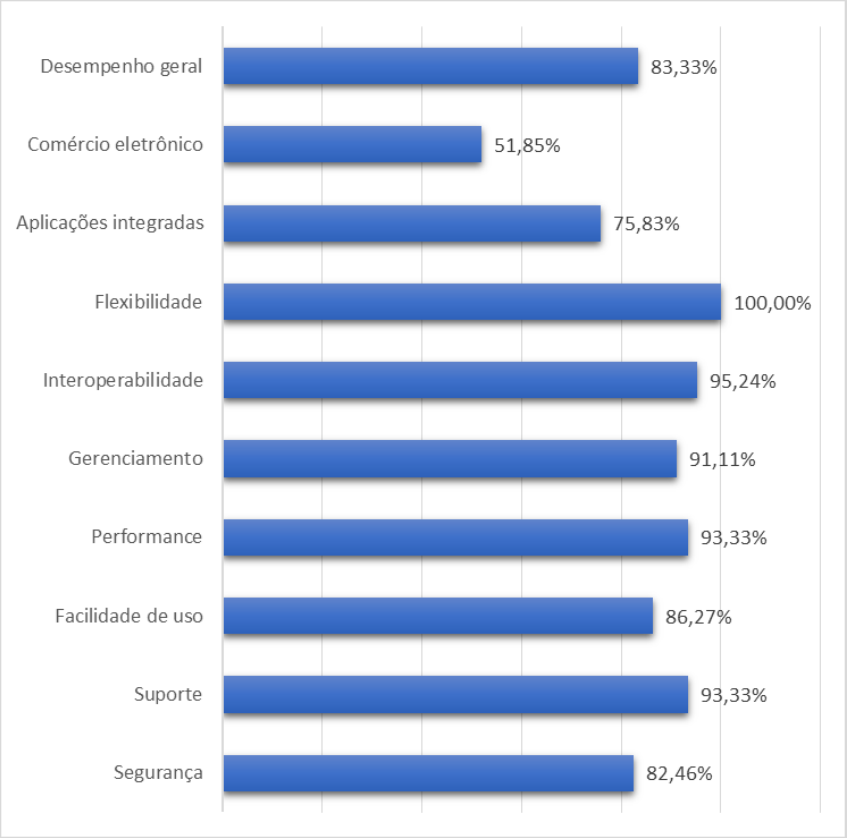
Fonte: Alves (2017).

Figura 5 – Desempenho obtido pelo *Joomla!*



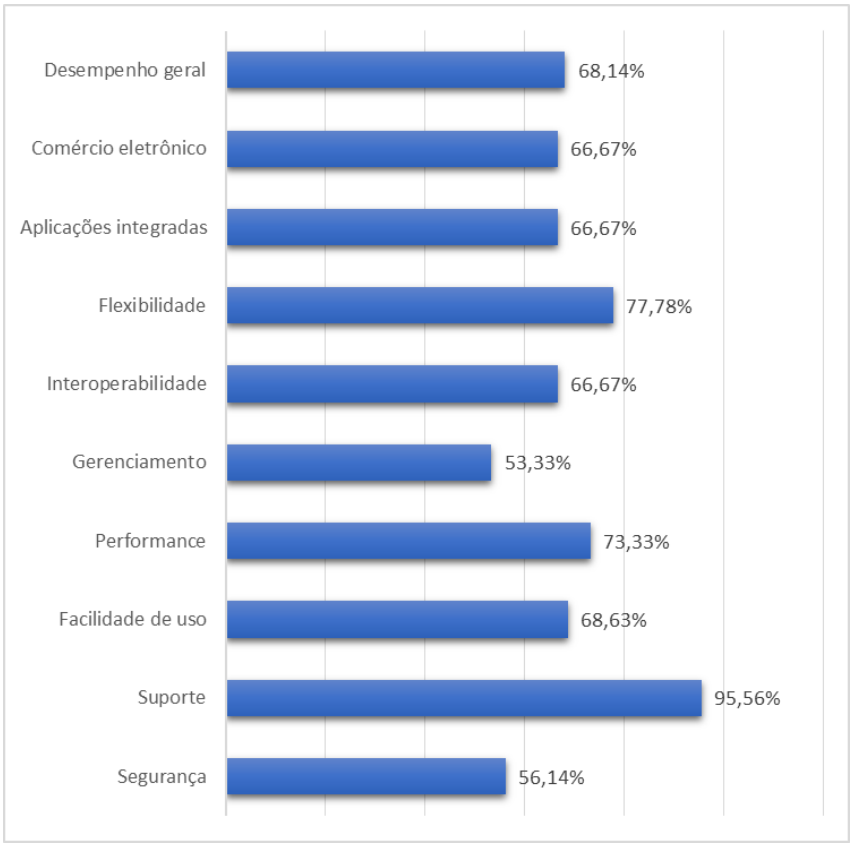
Fonte: Alves (2017).

Figura 6 – Desempenho obtido pelo *Plone*



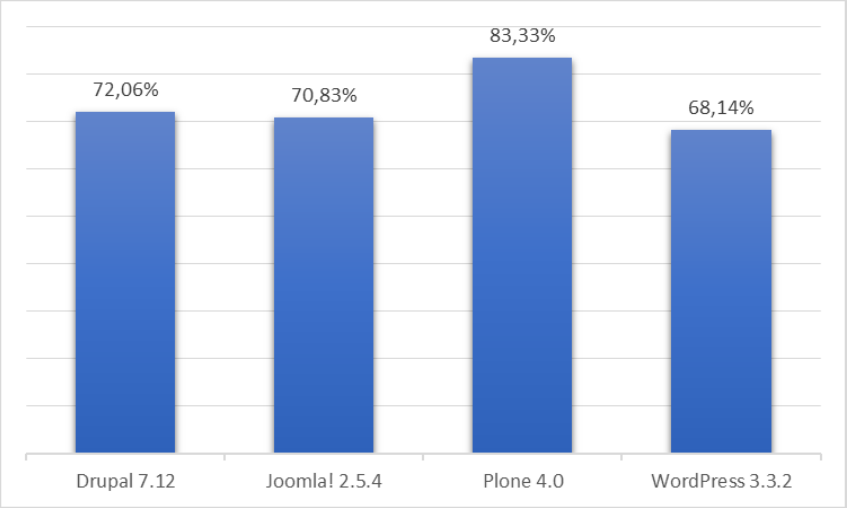
Fonte: Alves (2017).

Figura 7 – Desempenho obtido pelo *WordPress*



Fonte: Alves (2017).

Figura 8 – Desempenho geral obtido pelos gerenciadores

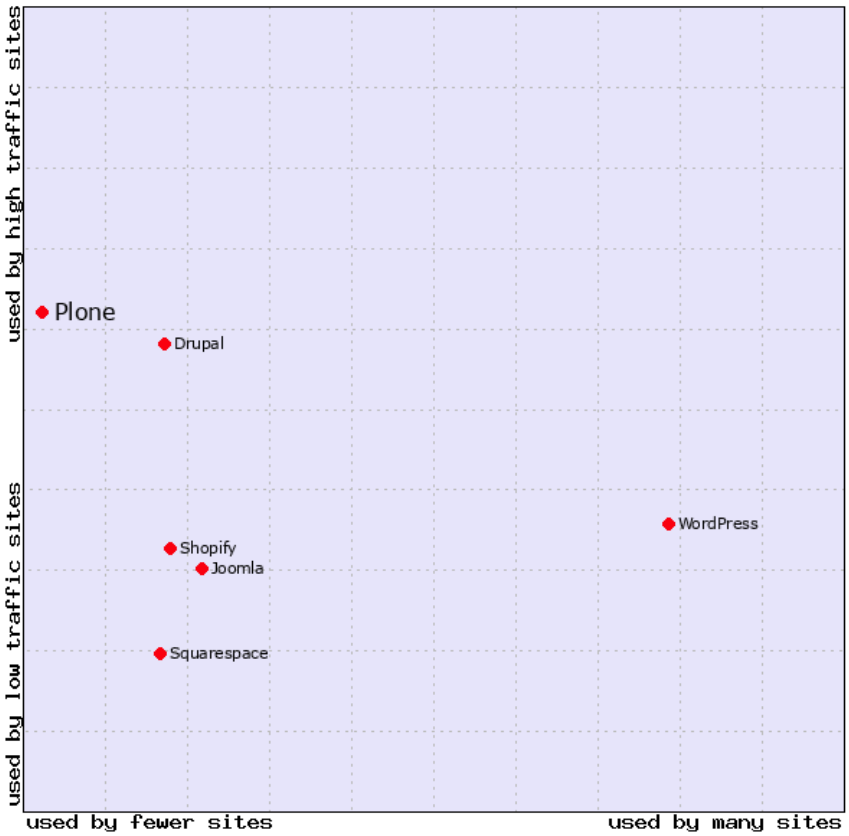


Fonte: Alves (2017).

Então pode-se considerar, diante de todas as circunstâncias e os testes executados cujos resultados estão exibidos nas figuras, que o *Plone* é o mais balanceado em termos de aspectos negativos e positivos, fornecendo uma maior confiança aos desenvolvedores de sistemas web que precisam de sistemas que reajam bem a módulos mais robustos, que sejam seguros e com grande comunidade virtual para auxílio ao desenvolvimento.

A imagem 9 mostra uma análise de posicionamento de mercado dos CMSs mais usados no mundo:

Figura 9 – Análise de posicionamento de mercado com destaque para o *Plone*



Fonte: (W3TECHS, 2019 apud ALVES, 2017).

Através dela percebemos que o *Plone* é muito preferido em situações que envolvem sistemas que receberão grande número de requisições de visitantes e usuários. Outro fator importante a se destacar é a baixa adesão das pessoas a esse gerenciador de conteúdo. Pela imagem percebe-se que o *WordPress* é disparadamente o gerenciador mais utilizado, e o *Plone*, o que está em menor número de sites.

O *Idealware*, um site que tem o propósito de ajudar organizações sem fins lucrativos a tomar decisões inteligentes sobre tecnologia, publicou o seguinte artigo: *Consumers Guide to Open Source Content Management Systems for Nonprofits: Comparing WordPress, Drupal, and Plone*, detalhando todos estes três sistemas.

Para a comparação, houve a criação da tabela 3 contendo 15 atributos, na qual está atribuído o nível dos softwares para cada um deles, sendo 1 significando um nível ruim de satisfação, 2 significando um nível médio e 3, por sua vez, determinando que o gerenciador cobre muito bem tal aspecto.

Tabela 3 – Nível de satisfação do CMS por requisito

Requisito	***Drupal***	***Plone***	***WordPress***
Facilidade de hospedagem e instalação	3	1	3
Facilidade de configurar um site simples	3	3	3
Curva de aprendizado para configurar um site complexo	2	1	3
Facilidade edição de conteúdo	3	3	3
Facilidade de gerenciar um site	2	2	3
Flexibilidade estrutural	3	3	2
Flexibilidade gráfica	3	3	3
Flexibilidade de suporte móvel	3	3	3
Integração com dados constituintes	2	2	1
Funções de usuário e fluxo de trabalho	2	3	2
Comunidade/Funcionalidade na Web 2.0	3	2	3
Acessibilidade	3	3	2
Otimização para mecanismos de pesquisa	2	3	2
Estendendo além da funcionalidade existente	3	3	3
Apoio e força da comunidade	3	3	3

Fonte: Idealware (2017). Adaptado.

A tabela 3 expressa uma informação importante: o *Plone* proporciona pouca facilidade para hospedagem, instalação e uma baixa curva de aprendizado por parte do desenvolvedor ao se deparar com a construção de um site complexo. Tais fatores associados à baixa popularidade do mesmo em relação ao outros gerenciadores o torna menos acessível aos usuários, impactando diretamente no número de sites que o utilizam.

Essas desvantagens se tornam pouco relevantes ao se considerar que a pessoa responsável pela implantação do sistema é capacitada para desempenhar a tarefa, além de que a aplicação poderá ou não ser complexa. Logo, é necessária a realização de um estudo de caso para avaliar se realmente compensa a utilização do *Plone* para a construção de um sistema web, observando as vantagens e desvantagens anteriormente mencionadas.

2.4 Servidor Virtual Privado (VPS)

Os Servidores Virtuais Privados surgiram em decorrência da continuação do processo de virtualização de máquinas. Com a VPS é possível ter sua própria máquina hospedada na nuvem, porém, assim como na Máquina Virtual (VM), o seu responsável não possui acesso direto ao hardware, somente ao software.

Segundo Chuchuca e José (2016) a hospedagem VPS é ideal para quem possui demandas que podem ser satisfeitas de forma compartilhada, mas desejando pagar menos que na contratação de um servidor dedicado. Uma curiosidade sobre a VPS é a permissão de gerenciar

o software de uma máquina inteira, inclusive no que diz respeito a configuração de servidores locais.

Esse tipo de serviço requer maior responsabilidade e conhecimento do administrador da máquina, pois ele deve entender o funcionamento e divisão da mesma dependendo de seu sistema operacional, e ainda seus detalhes técnicos e comandos de configuração e conexão via SSH. Em relação à segurança, (ELIZABETH; DUQUE; RAMÍREZ, 2017) revela este ser um fator positivo para os Servidores Privados Virtuais. Para isso, ela faz menção aos sistemas VoIP e às máquinas físicas, que são mais propensos a ataques, ameaças e riscos principalmente no caso de máquinas físicas, que estão sujeitas a blecautes, curto-circuitos, dano no dispositivo ou ainda a roubo de informações. Um outro problema em relação aos servidores físicos está na compatibilidade entre hardware e software, pois todos os equipamentos da máquina devem ser compatíveis. Esses transtornos não são encontrados na VPS, visto que a própria empresa que oferece o serviço de hospedagem já é responsável por manter a compatibilidade dos elementos e a segurança da máquina.

2.5 Testes de Software

Vou deixar po último porque não sei se vai dar tempo de executar.

2.6 Ferramentas

Nesta seção, o objetivo será a junção e uma explicação sucinta das ferramentas utilizadas no processo de desenvolvimento e avaliação do sistema web.

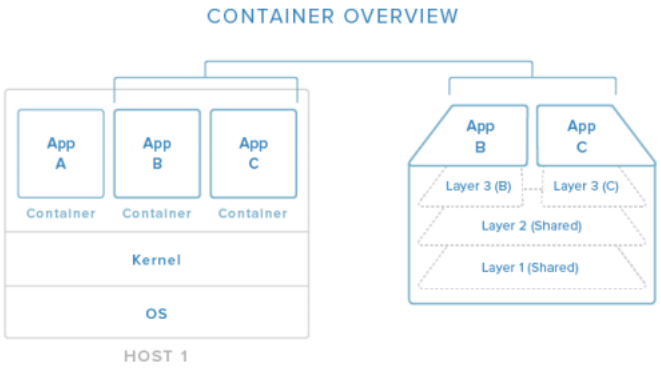
Apache JMeter: é uma ferramenta capaz de realizar testes em aplicações web. Por ser desenvolvido em Java, ela é também multiplataforma, ou seja, pode ser executada em vário sistemas operacionais. Um teste típico com o *JMeter* envolve o envio de requisições a uma aplicação através de um *loop*, cujo limite de laços e número de requisições são definidos pelo usuário.

Docker: é uma tecnologia ligada à infraestrutura de máquinas. Surgiu em 2013 após a mudança do nome *dotCloud* para o atual. O *Docker* usa a ideia de "camadas" e *containers*. *Containers* são aplicações que utilizam o mesmo *kernel* do sistema operacional onde está rodando. Uma aplicação pode ser praticamente qualquer software, inclusive outro sistema operacional, desde que tenha uma "imagem" para ela, porque um *container* é executado a partir de uma imagem.

Alguns podem confundí-lo com Máquinas Virtuais (VMs), mas no livro Vitalino e Castro (2016), os autores deixam claro as suas diferenças. Na Máquina Virtual, o sistema operacional a ser virtualizado é inicializado carregando todo o seu hardware e mais recursos que os da máquina *host*, diferentemente dos *containers*, que utilizam os mesmos recursos da máquina hospedeira, havendo, assim, um ganho em desempenho em relação à VM.

Compreender o conceito de *containers* é fundamental para entender o conceito de camadas, isto porque eles são formados por *layers* (camadas). Essa ideia parte do princípio de compartilhamento e reutilização, visto que vários *containers* podem compartilhar camadas subjacentes, diminuindo o uso de recursos Prada *et al.* (2017). A figura 10 esquematiza a estrutura da arquitetura de *containers* e camadas.

Figura 10 – Esquema explicativo da estrutura de *containers*



Fonte: (PRADA *et al.*, 2017).

Entre as diversas vantagens que o *Docker* proporciona podemos citar a teoria da escalabilidade, que corresponde à segmentação de todo um sistema, sua divisão em partes realiza o carregamento parcial dos *websites* contribuindo para a diminuição do tempo de acesso do visitante à página web. Tal modularização também auxilia na manutenção e disponibilidade das partes, isto porque, dependendo da situação, *containers* podem estar funcionando normalmente mesmo que um deles esteja em manutenção ou tenha sido derrubado, possibilitando ao usuário acessar um componente desde que o mesmo não dependa do inacessível.

Também é importante mencionar que com a correta orquestração dos *containers*, os mesmos poderão ser inseridos em uma imagem, através do recurso do *Docker Compose*, e carregados em qualquer ambiente, desde que ele possua o *Docker*. Esta funcionalidade garante o princípio de portabilidade e faz com que um sistema inteiro funcione em várias máquinas e sistemas operacionais com apenas os comandos de importar a imagem e iniciar um *container* da mesma.

Ainda pelo *Docker* é possível verificar o comportamento e eventuais problemas relacionados aos *containers* através de seus *logs*.

Git: é um sistema de versionamento de código lançado em 2005. O controle de versão possibilita acompanhar o desenvolvimento de um software sem alterar a versão principal, restaurar uma determinada versão anterior e até compartilhar o mesmo código com outros desenvolvedores Palestino (2015). "O Git possui uma ênfase em velocidade, integridade dos dados e suporte para *workflows* não lineares e distribuídos" (GHEZZI, 2015, p. 10).

Para hospedar códigos do *Git* existem diversos sites, porém o mais usado e também um dos poucos que oferecem opções de hospedagem pública e privada é o *GitHub*, que em 2019,

inclusive, tornou possível aos usuários a criação de repositórios privados gratuitamente, o que antes era uma funcionalidade paga Chacon e Straub (2010).

Google Analytics: é uma ferramenta da *Google* utilizada para obtenção de relatórios sobre número, origem, tempo de duração e muitas outras informações sobre as visitas que um sistema recebe. Com ela é possível identificar padrões relacionados a essas visitas, taxa de aumento ou diminuição no número de visitantes, páginas da aplicação mais acessadas, entre outros dados.

MySQL: a SQL (Linguagem de Consulta Estruturada, em português) é um tipo de programação especializada em trabalhar com bancos de dados relacionais (SUEHRING, 2001, p. 7, tradução nossa). Um dos diversos bancos de dados relacionais existentes é o *MySQL*, desenvolvido pela empresa *Microsoft* e considerado um servidor e gerenciador de banco de dados (SGBD) relacional de código aberto (*open source*). O *MySQL* possui todos os atributos necessários a um banco de dados de grande porte e é reconhecido por muitos como o *open source* mais concorrente de SGBDs de código fechado Milani (2007).

3 METODOLOGIA

No capítulo de metodologia o objetivo é descrever como ocorreu o processo de construção do sistema, desde a justificativa para a escolha dos métodos de desenvolvimento até a completa entrega do software ao cliente.

O capítulo está dividido no seguinte escopo: as seções 3.1 e 3.2, que justificam a escolha da metodologia e do gerenciador de conteúdo para a implementação do sistema; a seção 3.3 apresenta como todas as etapas de implementação e reuniões com o cliente foram estabelecidas; e, por último, a seção 3.4 aborda como os testes de software foram feitos e o desempenho obtido pela aplicação web.

3.1 Definição do tipo de metodologia ágil de desenvolvimento

A definição de uma metodologia ágil de desenvolvimento para o atual projeto é consequência das informações apresentadas na seção 2.2, principalmente em relação às informações levantadas nos quadros 1 e 2, que dizem respeito a características das metodologias ágeis e tradicionais, principalmente relativas a tempo, comunicação e riscos tendo em vista a baixa complexidade do projeto, o curto tempo para desenvolvimento e produção da documentação.

Após a certificação de que uma metodologia ágil deve ser empregada, pode-se dizer que vários são os motivos para a escolha do *Scrum* como metodologia de desenvolvimento e Vasconcelos *et al.* (2015, p. 560) divulga diversos deles em seu trabalho. Pode-se citar a diminuição de reclamações (MANN; MAURER, 2005 apud VASCONCELOS *et al.*, 2015), o aumento do retorno do investimento em projetos de novos produtos (SULAIMAN; BARTON; BLACKBURN, 2006 apud VASCONCELOS *et al.*, 2015), a melhoria da qualidade do produto produzido e diminuição dos custos de produção (SUTHERLAND *et al.*, 2008 apud VASCONCELOS *et al.*, 2015) e a diminuição no tempo gasto para terminar projetos de desenvolvimento de novos produtos (SANDERS, 2007 apud VASCONCELOS *et al.*, 2015).

Três características pertencentes ao *Scrum* são a leveza, a simplicidade de entender e a dificuldade de se dominá-lo (SUTHERLAND; SCHWABER, 2007 apud PAINKA[1]; MARCHI[1], 2013), isto porque é fácil entender suas etapas e o papel de cada um no projeto, mas para adequar essa metodologia à realidade da empresa é necessário experiência e conhecimento.

Assim, às vezes pode ser conveniente alterar prazos, papéis e etapas da metodologia, visando torná-la uma ferramenta de auxílio customizada, e fatores como a disponibilidade de clientes, curtos prazos ou complexidade das tarefas devem interferir em decisões dessa espécie, por isso cada projeto utiliza o *Scrum* de uma maneira, ficando a critério das equipes, em especial o *Scrum Master*, quando e onde remodelar.

Outros fatores que contribuiram para definição do *Scrum* como metodologia de desenvolvimento para o projeto atual estão esclarecidos na tabela 1 e são alusivos ao tamanho da equipe, que corresponde a uma única pessoa, e à elicitação de requisitos, que, por sua vez,

não possui uma única forma de ser elaborada. Também contribui para o processo de definição a familiaridade do autor com a metodologia mencionada, já tendo sido estudada e colocada em prática em situações ocorridas durante o curso de Sistemas de Informação.

3.1.1 Sobre o Scrum

O *Scrum* é um processo para construção incremental de softwares em ambientes complexos e provê o desenvolvimento de softwares em curtas iterações, denominadas *sprints* Rising e Janoff (2000).

Sprints: cada *sprint* inclui todas as fases de um software modelo de ciclo de vida de desenvolvimento, como design, implementação, testes, revisão de clientes, etc (MATHARU *et al.*, 2015, p. 2, tradução nossa). Elas têm duração de até 30 dias e também compreendem o procedimento de adaptação a mudança de variáveis (requisitos, tempo, recursos, tecnologia), pois em seu término há sempre uma reflexão sobre as tarefas, definidas antes do início do ciclo, que foram realizadas com sucesso e os próximos incrementos ou revisões as serem executados nas próximas *sprints*, de acordo com o *feedback* passado na reunião com o cliente Awad (2005).

Existem cinco características únicas para o desenvolvimento baseado em *Scrum* Matharu *et al.* (2015), são eles:

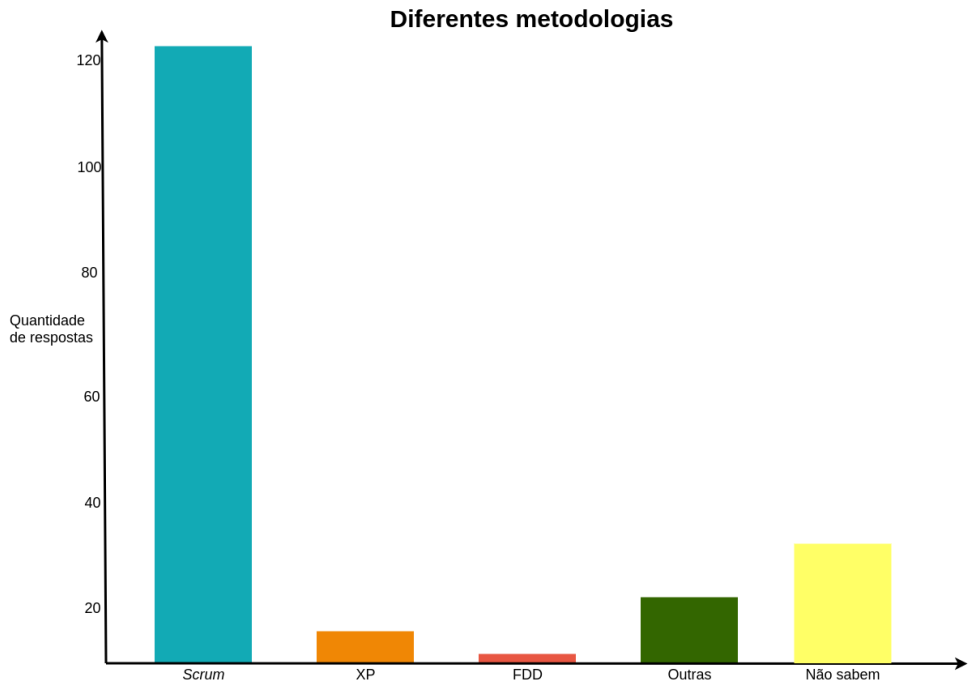
1. Colaboração: a promoção da colaboração se dá pelo fato do desenvolvimento ser conduzido por equipes compostas por pessoas multifuncionais, envolvendo programadores, arquitetos de software e especialistas em qualidade de software.
2. Encontros diários: são reuniões de curta duração, comandadas pelo *Scrum Master*, onde as equipes de desenvolvimento se comunicam e discutem o progresso com que os requisitos encontrados no *product backlog* estão sendo implementados. Rising e Janoff (2000) afirma que esses encontros podem ser realizados três ou quatro vezes na semana e neles os assuntos mais frequentes se baseiam nas dificuldades encontradas e como vencer os obstáculos até a próxima reunião.
3. *Product Backlog*: o *product backlog* captura os requisitos que devem ser implementados, sendo ordenados por nível prioridade. Ele ainda contém erros resolvidos, características gerais e requisitos não funcionais do software.
4. *Sprint Backlog*: este item registra a lista de tarefas que deverão ser realizadas durante a próxima *sprint*. Logicamente a lista de tarefas deve ser baseada no rendimento das equipes para que não haja um planejamento de quantidade de itens exorbitante que dificilmente serão cumpridos.
5. Regras: são regras fundamentais da metodologia segundo Matharu *et al.* (2015):
    - PO *(Product Owner)*: “responsável pela definição, priorização e comunicação dos requisitos de produto e guias do processo de desenvolvimento” (MATHARU *et al.*, 2015, p. 3, tradução nossa).

- Time de desenvolvimento: equipes compostas de 3 a 9 indivíduos responsáveis por executar as tarefas previstas pelo PO.
- *Scrum Master*: responsável por guiar as equipes no respeito às regras e princípios do *Scrum*. Ele remove impedimentos e auxilia no processo de desenvolvimento.

Em pesquisa realizada pelo *State of Agile Report* (Estado de Relatório Rápido, em português), em 2019, 72% das empresas que participaram responderam praticar com *Scrum* ou uma metodologia híbrida que o utiliza como metodologia de desenvolvimento. Tal pesquisa, encontrada em Agile (2019), foi realizada pela décima terceira vez em 2019 e coletou informações sobre metodologias de processos e desenvolvimento de empresas e organizações da Europa, Ásia, América do Sul e África dos mais variados ramos, tais como de tecnologia, de transporte, industrial, governamental e energético.

Os resultados obtidos com a pesquisa identificam a importância de se conhecer a metodologia estudada nesta seção e como ela ainda é usada por empresas do mundo inteiro. Com o objetivo de assegurar e dar ainda mais veracidade a esta conclusão, outra pesquisa, esta realizada em funcionários da empresa de tecnologia *Microsoft* e via web, detalhada em Begel e Nagappan (2007), aponta que, das 192 respostas recebidas, a maioria das pessoas que trabalham com desenvolvimento, testes e funções de gerenciamento diretamente ligadas a produção de software aplicam o *Scrum* como a metodologia padrão para construir seus produtos, como representado na figura 11.

Figura 11 – Diferentes metodologias utilizadas pelos funcionários



Fonte: Begel e Nagappan (2007). Adaptado.

3.2 Definição do gerenciador de conteúdo

Analisando as informações extraídas na subseção 2.3.2, no capítulo 2, o gerenciador de conteúdo escolhido pelo autor foi o *Plone*. Os principais fatores que influenciaram nesta decisão estão atrelados ao desempenho e segurança de *websites* construídos com esse CMS. Também é importante destacar que o *Plone* é bastante usado para criação de portais, por exemplo o portal da UFVJM, que possuem características semelhantes ao sistema da Kuruatuba, como a publicação de notícias e eventos.

O *Plone* possui ferramentas personalizadas e de fácil manipulação que facilitam o gerenciamento de notícias e eventos, primeiramente por possuir campos e opções já pré definidos para o tipo de conteúdo que o usuário irá inserir, e por último por inserir automaticamente o conteúdo criado na lista de conteúdos, devido a uma configuração realizada somente uma vez ao criar o sistema. Sendo assim, basta que o usuário preencha os dados relacionados ao tipo de conteúdo (notícia ou evento) para que ele seja criado e já exibido na página web.

Outro ponto positivo do *Plone* é que sua utilização não necessita da instalação de *plug-ins*, extensões ou complementos, podendo ter seus elementos estáticos, como rodapé e cabeçalho, personalizados via *Portlets*, que segundo UFRGS (2012), são aplicativos e ferramentas prontas para uso em qualquer instalação padrão, podendo ser usadas para calendário, notícias, eventos, menu de busca, enquetes, etc.

3.3 Desenvolvimento

A seção de desenvolvimento sinaliza o início, de fato, da realização das tarefas e demais atividades relacionadas à construção do sistema da Kuruatuba. É a partir desse momento que serão informados dados como domínio, hospedagem, requisitos, versões das tecnologias utilizadas e descrição das atividades desempenhadas.

Para o projeto a VPS contratada é de posse da empresa *Hostinger*, encontrada pelo endereço eletrônico https://www.hostinger.com.br/, com o sistema operacional *Ubuntu* 16.04 e o seguinte domínio: http://www.kuruatuba.org. Também usadas as seguintes ferramentas e suas respectivas versões: *Plone* 5.2.1, PHP 7.4.2, *MySQL* 5.7, *Docker* 19.03.5 e *Git* 2.17.1, este utilizado para o versionamento de código do sistema para cadastro de associados uma vez que ele poderá ser necessário, futuramente, a outro desenvolvedor para atualizações.

A seguir, no decorrer da seção, os temas levantados serão os que se seguem: as subseções 3.3.1 e 3.3.2 tratarão da elaboração das histórias de usuário e da coleta de requisitos, respectivamente, a subseção 3.3.3 apresentará os casos de uso e fluxos alternativos do sistema como forma de auxiliar no entendimento de seu funcionamento, a subseção 3.3.4 apresentará como ficaram os elementos do *Scrum* aplicados ao contexto do presente trabalho, algumas telas do software ficarão disponíveis na subseção 3.3.5 e de mesma maneira a estrutura de *containers* do *Docker* será ilustrada na subseção 3.3.6.

3.3.1 Histórias de usuário

Segundo Longo e Silva (2014) as histórias de usuário são muito importantes tanto para a criação de requisitos nos métodos ágeis *Extreme Programming* e *Scrum* quanto para provocar o envolvimento do cliente, porém não poderão substituir a comunicação entre clientes e equipe Carvalho, Barbosa e Silva (2014).

Cada história de usuário deve conter três expressões: “como um... ”, que identifica o papel do usuário na história; “eu quero...”, funcionalidades requisitadas; e “de modo que...”, benefício conquistado ao realizar a tarefa (COHN, 2004 apud LONGO; SILVA, 2014); além de seis atributos indispensáveis identificados pelo acrônimo inglês INVEST (*Independent, Negotiable, Valuable to users or customers, Estimatable, Small and Testable*) (COHN, 2004) e explicados em Carvalho, Barbosa e Silva (2014):

- Independente: uma história deve poder ser desenvolvida, testada e entregue de forma isolada, uma não depende de outras;
- Negociável: através das histórias é possível discutir requisitos, e por elas ainda deve haver possibilidade de mudanças de funcionalidades e datas de entrega;
- Valoroso: assim como os métodos ágeis as histórias de usuário devem proporcionar valor para o cliente, através delas é importante que sinta satisfação e sentimento de realização com as informações discutidas;
- Estimável: uma equipe deve ser capaz de definir o nível de complexidade, a mão de obra necessária e o tempo para conseguir implementar uma história de usuário. Caso isso não seja possível a história deve ser dividida em histórias menores até que a estimativa seja validada.
- Tamanho pequeno: as histórias devem ser pequenas para prover maior agilidade na implementação.
- Testável: testes devem ser realizados em cada história criada pois além dos conteúdos ficarem mais organizados a equipe também conseguirá codificar cada uma caso os testes sejam aceitos.

A seguir estão as histórias de usuário para o trabalho em questão já testadas e validadas junto ao cliente, uma observação a ser feita corresponde ao fato de que os nomes utilizados nas histórias são fictícios, não fazendo alusão ou referência a nenhum dos envolvidos.

História 1: Como o presidente da associação, João deseja cadastrar e remover, via computador ou *smartphone*, usuários do sistema que vão contribuir nas divulgações e no gerenciamento de associados, de modo a estabelecer possíveis trocas de contribuidores.

História 2: Como presidente da associação, João também deseja publicar, via computador ou *smartphone*, eventos e notícias e alterar o conteúdo estático das páginas quando

for necessário, com o objetivo de executar tais tarefas sem necessitar de outro encarregado a prestar tais serviços.

História 3: Como presidente da associação, João precisa gerenciar, via computador ou *smartphone*, o cadastramento de associados, possibilitando o cadastro, a edição, a remoção, a pesquisa e o fornecimento de carteirinha associativa aos mesmos, com o objetivo de adiantar a realização de tarefas sem necessitar de seus subordinados ou responder à solicitação de colaboradores ou visitantes.

História 4: Como um secretário da associação, Paulo pretende publicar, via computador ou *smartphone*, eventos e notícias e alterar o conteúdo de páginas conforme solicitação de seu superior.

História 5: Como uma secretária da associação, Maria pretende gerenciar, via computador ou *smartphone*, o cadastramento de associados, possibilitando o cadastro, a edição, a remoção, a pesquisa e o fornecimento de carteirinha associativa aos mesmos de acordo com as solicitações vindas do presidente ou dos próprios associados, no âmbito da geração da carteirinha.

3.3.2 Coleta de requisitos

Uma etapa extremamente importante no desenvolvimento de produtos é a coleta de requisitos, pois é com base nela que artefato será construído, tendo em vista que este é o momento em que o cliente tenta exprimir o que deseja que o software tenha e seja capaz de processar, envolvendo também a abstração dessas informações pela equipe responsável por passá-las para o desenvolvimento.

Para que tudo ocorra da melhor maneira a qualidade na comunicação é essencial e uma abordagem superficial das ideias deve ser evitada a fim de eliminar falsas convicções, ideias distintas entre cliente e desenvolvedor e desperdício de tempo e demais recursos. Para isso existem as maneiras de extrair os requisitos de sistema do cliente e usuário, como explanado no tópico de especificação de software, na seção 2.1. Dentre as opções lá citadas e as histórias de usuário particularizadas na subseção 3.3.1, a coleta de requisitos para a aplicação web da Kuruatuba consistiu-se em um formulário e em curtas reuniões com o atual presidente da associação, o professor Erinaldo.

O formulário, disponibilizado no apêndice B, foi acessível para os futuros utilizadores do sistema para que opinassem sobre o que deveria e o que não deveria estar no produto final, além do grau de importância de cada requisito de acordo com suas exigências e necessidades. De acordo com as respostas registradas no formulário, com reuniões anteriormente mencionadas e com as histórias de usuário descritas obteve-se os seguintes requisitos de sistema:

1. Sistema de login e cadastro para os usuários;
2. Publicação de notícias, eventos e mais informações sobre a associação;
3. Cadastro, atualização e remoção de associados;
4. Sistema para gerar carteirinha para associados;

5. Formulário para receber mensagens dos visitantes;
6. Publicação de eventos organizados pela associação;
7. *Link* para baixar o estatuto da Kuruatuba.

3.3.3 Casos de uso e fluxos de eventos

A presente subseção está dividida em duas partes: a primeira, exposta em 3.3.3.1, aborda de maneira geral sobre diagramas de casos de uso e apresenta o diagrama associado ao trabalho em desenvolvimento; e a segunda, encontrada em 3.3.3.2, demonstra a versão final dos fluxos de eventos dos casos de uso.

3.3.3.1 Diagrama de casos de uso

O diagrama de casos de uso pertence à Linguagem de Modelagem Unificada (UML) e é comumente utilizado para documentar requisitos de sistema, modelando seu contexto e facilitando seu entendimento por exibí-lo externamente aos desenvolvedores Uso (2001).

Em geral, os diagramas de casos de uso apresentam os seguintes elementos ZAPATA *et al.* (2007):

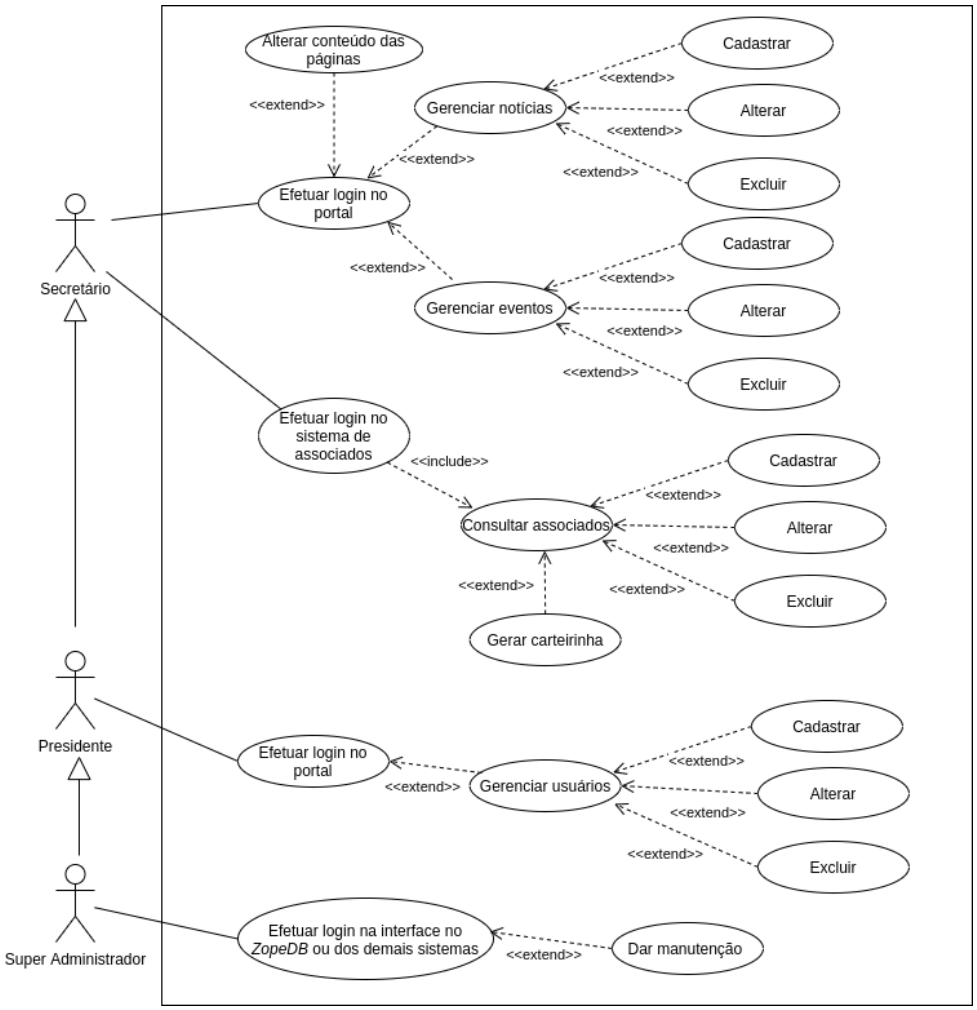
- Casos de uso: ações realizadas pelos atores associados que descrevem do início ao fim do processo;
- Atores: são os usuários, sendo humanos ou outros sistemas, que poderão realizar os casos de uso e as funcionalidades do sistema;
- Relacionamentos: são interações entre dois casos de uso, dois atores ou um caso de uso e um ator. Elas podem der divididas em quatro tipos:

    1. Associação: estabelece relação entre um ator e um caso de uso;
    2. *include*: inclui o comportamento de um caso de uso em outro, ou seja, ao realizar um caso de uso o outro ligado por meio do *include* também será realizado;
    3. *extend*: a presença do *extend* ligando um caso de uso A a um caso de uso B significa que através de A o *use case* B poderá ser realizado também. Diferentemente do relacionamento anterior, um caso de uso não é obrigatoriamente executado quando o outro for, tal ocorrência se dará por meio de condições;
    4. Herança: este relacionamento simboliza o compartilhamento das especificações de um ator a outro, ou seja, um ator irá herdar todas as permissões em realizar os casos de uso do outro relacionado.

- Cenário: uma sequência de ações que ilustra um comportamento. São usados para ilustrar a interação ou execução de uma instância a um caso de uso (ZAPATA *et al.*, 2007, p. 239, tradução nossa).

Após a breve explicação sobre os diagramas de casos de uso, sua importância e seus elementos, fica representado na figura 12 o diagrama referente ao sistema da associação Kuruatuba.

Figura 12 – Diagrama de casos de uso do sistema da Kuruatuba



Fonte: Autor.

3.3.3.2 Fluxos de eventos

A seguir estão os fluxos de eventos para os casos de uso definidos anteriormente. Tais fluxos auxiliam ainda mais na compreensão de como o sistema se comportará após cada requisição do usuário e quais possibilidades de interação com a interface do software este terá ao navegar pelas telas.

3.3.4 Definição das sprints

A presente subseção irá apresentar os ciclos de desenvolvimento, cada um deles possui as funcionalidades a serem desenvolvidas, requisitos atendidos (vide a subseção 3.3.2), seu nível de prioridade e o prazo para desenvolvê-las. É relevante saber que grande parte do desenvolvimento foi feita em máquina local para só depois ocorrer a migração da aplicação, mesmo que incompleta, para o servidor virtual. Considera-se o nível de prioridade em uma escala de 1 a 5, onde 1 representa pouquíssima prioridade e 5 altíssima prioridade.

Tabela 4 – *Sprint* 1

Nome	Prioridade	Prazo (dias)	Requisitos atendidos
-	-	-	-
-	-	-	-

Fonte: Autor.

3.3.5 Apresentação das telas

3.3.6 Estrutura de containers do sistema

Como o sistema deve gerenciar dois tipos de público, um de associados e outro de usuários, resolveu-se criar três *containers*: um responsável pelo *Plone* que, por sua vez, apresenta o *website* da Kuruatuba para os visitantes; um *container* responsável por comportar o sistema de gerenciamento de associados, desenvolvido na linguagem PHP, que é conhecida como uma linguagem de programação da era da Web 2.0 que permite o desenvolvimento ágil de software do lado do servidor Suzumura *et al.* (2008); e um *container* abrigando o banco de dados, construído com a ferramenta *MySQL*, que possui as informações relacionadas aos associados e onde são realizadas as consultas por parte do sistema escrito em PHP.

A criação e comunicação dos *containers* compactua com as vantagens descritas na seção 2.6 do capítulo 2, principalmente no que diz respeito à portabilidade e disponibilidade dos recursos. Tais benefícios também se aplicam à segurança, visto que para uma possível invasão ao sistema, é necessário que o invasor conheça o IP e porta de cada *container* e ainda os dados de acesso tanto ao *website* no *Plone* quanto à aplicação destinada a armazenar os dados sobre os associados.

Outra justificativa para a divisão das aplicações web está na complexidade de manipulação dos SGBDs - Sistema de Gerenciamento de Banco de Dados - não relacionais. Como já explicado na subseção 2.3.1, no capítulo 2, o *Plone* utiliza o banco de dados não relacional *ZODB*, e o mesmo é bastante complexo quando o desenvolvedor necessita realizar consultas ou inserções a seus objetos, sendo inviável a criação de páginas ou elementos para tal tarefa.

3.4 Execução de testes de desempenho

3.5 Trabalhos correlatos

REFERÊNCIAS

AGILE, S. of. **Annual State Of Agile Report**. 2019. Disponível em: ⟨https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report⟩. Acesso em: 11-3-2020.

ALMEIDA, M. E. B. de. Educação a distância na internet: abordagens e contribuições dos ambientes digitais de aprendizagem. **Educação e pesquisa**, SciELO Brasil, v. 29, n. 2, p. 327–340, 2003.

ALVES, E. d. C.

RECONSTRUÇÃO DO PORTAL INSTITUCIONAL DA UFVJM: adoção da Identidade Digital do Governo Federal e implementação do PloneGov-BR como novo Sistema de Gerenciamento de Conteúdo — Universidade dos Vales do Jequitinhonha e Mucuri, Diamantina, MG, Brasil, 2017.

ANWER, F.; AFTAB, S.; SHAH, S. M.; WAHEED, U. Comparative analysis of two popular agile process models: Extreme programming and scrum. **International Journal of Computer Science and Telecommunications**, v. 8, n. 2, p. 1–7, 2017.

ARAÚJO, V. M. H. de; FREIRE, I. M. A rede internet como canal de comunicação, na perspectiva da ciência da informação. **Transinformação**, v. 8, n. 2, 2012.

AWAD, M. A comparison between agile and traditional software development methodologies. **University of Western Australia**, v. 30, 2005.

BEGEL, A.; NAGAPPAN, N. Usage and perceptions of agile software development in an industrial context: An exploratory study. In: IEEE. **First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)**. [S.l.], 2007. p. 255–264.

BOIKO, B. Content management bible. John Wiley & Sons, New York, NY, USA, 2001.

CARVALHO, L. A. C.; BARBOSA, M. W.; SILVA, V. B. Proposta e avaliação de uma abordagem lúdica para o ensino de histórias de usuário e scrum. **Revista de Gestão e Projetos-GeP**, v. 5, n. 3, p. 44–58, 2014.

CHACON, S.; STRAUB, B. Pro git. **Recuperado de: http://labs. kernelconcepts. de/downloads/books/Pro% 20Git**, 2010.

CHAGAS, F.; CARVALHO, C. L. de; SILVA, J. C. da. Um estudo sobre os sistemas de gerenciamento de conteúdo de código aberto. **Revista Telfract**, v. 1, n. 1, 2018.

CHUCHUCA, F.; JOSÉ, C. **Implementación de una central telefónica opensource en la nube, a través de un servidor virtual privado (VPS), para servicio de hosting compartido para Empresas PYME.** Tese (Doutorado) — Universidad de Guayaquil Facultad de Ciencias Matemáticas y Físicas Carrera . . . , 2016.

COHN, M. **User stories applied: For agile software development**. [S.l.]: Addison-Wesley Professional, 2004.

CONFORTO, E. C. **Gerenciamento ágil de projetos: proposta e avaliação de método para gestão de escopo e tempo**. Tese (Doutorado) — Universidade de São Paulo, 2009.

DINGSØYR, T.; NERUR, S.; BALIJEPALLY, V.; MOE, N. B. **A decade of agile methodologies: Towards explaining agile software development**. [S.l.]: Elsevier, 2012.

ELIZABETH, M. I. R.; DUQUE, L. A. A.; RAMÍREZ, J. E. P. Diseño de un servicio pbx hospedado en un servidor virtual privado vps en la nube para uso de empresas pymes que no cuentan con servicios de telefonías de voip. **Dominio de las Ciencias**, Polo de Capacitación, Investigación y Publicación (POCAIP), v. 3, n. 2, p. 866–889, 2017.

GARRETT, J. J. *et al.* Ajax: A new approach to web applications. 2005.

GHEZZI, A. P. Api para auxílio de mineração de repositórios git e svn. 2015.

GINIGE, A.; MURUGESAN, S. Web engineering: An introduction. **IEEE multimedia**, IEEE, v. 8, n. 1, p. 14–18, 2001.

GONÇALVES, R. F.; GAVA, V. L.; PESSÔA, M. S. D. P.; SPINOLA, M. D. M. Uma proposta de processo de produção de aplicações web. **Production**, SciELO Brasil, v. 15, n. 3, p. 376–389, 2005.

IDEALWARE. Consumers guide to open source content management systems for nonprofits: Comparing wordpress, drupal, and plone. 2017.

KURUATUBA. **Histórico da ONG KURUATUBA**. 2011. Disponível em: ⟨http://ongkuruatuba.blogspot.com/⟩. Acesso em: 3-10-2019.

LONGO, H. E. R.; SILVA, M. P. A utilização de histórias de usuários no levantamento de requisitos ágeis para o desenvolvimento de software. **International Journal of Knowledge Engineering and Management (IJKEM)**, v. 3, n. 6, p. 1–30, 2014.

MANN, C.; MAURER, F. A case study on the impact of scrum on overtime and customer satisfaction. In: IEEE. **Agile Development Conference (ADC'05)**. [S.l.], 2005. p. 70–79.

MATHARU, G. S.; MISHRA, A.; SINGH, H.; UPADHYAY, P. Empirical study of agile software development methodologies: A comparative analysis. **ACM SIGSOFT Software Engineering Notes**, ACM New York, NY, USA, v. 40, n. 1, p. 1–6, 2015.

MEIKE, M.; SAMETINGER, J.; WIESAUER, A. Security in open source web content management systems. **IEEE Security & Privacy**, IEEE, v. 7, n. 4, p. 44–51, 2009.

MENEZES, J. S. S. d. *et al.* Processo de avaliação de software aplicado à seleção de sistemas gerenciadores de conteúdo. Universidade Federal de Sergipe, 2016.

MILANI, A. **MySQL-guia do programador**. [S.l.]: Novatec Editora, 2007.

MIRDHA, A.; JAIN, A.; SHAH, K. Comparative analysis of open source content management systems. In: IEEE. **2014 IEEE International Conference on Computational Intelligence and Computing Research**. [S.l.], 2014. p. 1–4.

OLIVEIRA, H. P. C. de; CÓRDULA, A. C. C.; FIUZA, N. J. A.; BRITO, M. P. de *et al.* Repositórios digitais utilizando wordpress e mysql. **BiblioCanto**, v. 3, n. 1, p. 144–157, 2017.

OTHMAN, M.; ISMAIL, S. N.; RAUS, M. I. M. The development of the web-based attendance register system (ars) for higher academic institution: From feasibility study to the design phase. **International Journal of Computer Science and Network Security**, v. 9, n. 10, p. 203–208, 2009.

PAINKA[1], M. A. L.; MARCHI[1], K. R. da C. Utilização das metodologias ágeis xp e scrum para o desenvolvimento rápido de aplicações. 2013.

PALESTINO, C. M. C. Estudo de tecnologias de controle de versões de software. 2015.

PATEL, S. K.; RATHOD, V.; PRAJAPATI, J. B. Performance analysis of content management systems-joomla, drupal and wordpress. **International Journal of Computer Applications**, Citeseer, v. 21, n. 4, p. 39–43, 2011.

PRADA, D. L.; THIEL, N. K.; CACHOEIRA, R.; REIS, W. dos. Docker–apresentação da ferramenta. In: **C749 Congresso Catarinense de Ciência da Computação (4.: 2017: Rio do Sul, SC). Anais do IV Congresso Catarinense de Ciência da Computação. Rio do Sul, Santa Catarina, Junho 12-14; 20-21, 2017./Organizador: Wesley dos Reis Bezerra.-Rio do Sul, SC, 2017.** [S.l.: s.n.], 2017. p. 46.

RISING, L.; JANOFF, N. S. The scrum software development process for small teams. **IEEE software**, IEEE, v. 17, n. 4, p. 26–32, 2000.

SANDERS, D. Using scrum to manage student projects. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 23, n. 1, p. 79–79, 2007.

SATO, D. T. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. **Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo**, v. 139, 2007.

SILVA, D. E. dos S.; SOUZA, I. T. de; CAMARGO, T. Metodologias ágeis para o desenvolvimento de software: Aplicação e o uso da metodologia scrum em contraste ao modelo tradicional de gerenciamento de projetos. **Revista Computação Aplicada-UNG-Ser**, v. 2, n. 1, p. 39–46, 2013.

SILVEIRA, C. S. D. Uma solução para o auxílio à geração de páginas web acessíveis com uso de gerenciadores de conteúdo. 2010.

SOARES, M. dos S. Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de software. **INFOCOMP Journal of Computer Science**, v. 3, n. 2, p. 8–13, 2004.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: Addison-Wesley, 2003.

STONE, W. . **Open Source CMS Market Share**. 2011. Disponível em: ⟨http://www.waterandstone.com/book/2011-open-source-cms-market-share-report⟩. Acesso em: 28-10-2019.

SUEHRING, S. **MySQL Bible**. 2001.

SULAIMAN, T.; BARTON, B.; BLACKBURN, T. Agileevm-earned value management in scrum projects. In: IEEE. **AGILE 2006 (AGILE'06)**. [S.l.], 2006. p. 10–pp.

SUTHERLAND, J.; SCHOONHEIM, G.; RUSTENBURG, E.; RIJK, M. Fully distributed scrum: The secret sauce for hyperproductive offshored development teams. In: IEEE. **Agile 2008 Conference**. [S.l.], 2008. p. 339–344.

SUTHERLAND, J.; SCHWABER, K. The scrum papers. **Nuts, Bolts and Origins of an Agile Process**, 2007.

SUZUMURA, T.; TRENT, S.; TATSUBORI, M.; TOZAWA, A.; ONODERA, T. Performance comparison of web service engines in php, java and c. In: IEEE. **2008 IEEE International Conference on Web Services**. [S.l.], 2008. p. 385–392.

TOMLINSON, T. **Beginning Drupal 7**. [S.l.]: Apress, 2010.

UFRGS. **Portlets — Tutorial Plone 4**. 2012. Disponível em: ⟨http://www.ufrgs.br/tutorial-plone4/posicionamento-e-estrutura/portlets⟩. Acesso em: 8-3-2020.

USO, D. d. C. de. Diagramas de casos de uso. **URL: http://www. vico. org/MuestrarioDiagCU. pdf**, 2001.

VASCONCELOS, B.; CARVALHO, D.; HENRIQUE, C.; MELLO, P. Implementation of scrum agile methodology in software product project in a small technology-based company MÉTRICAS E INDICADORES DE INOVAÇÃO: PROPOSTA DE DESENVOLVIMENTO DE SISTEMA DE MEDIÇÃO DE DESEMPENHO DA INOVAÇÃO ABERTA EM CENTROS DE TECNOLOGIA View project Product development View project. 2015. Disponível em: ⟨https://www.researchgate.net/publication/262656547⟩. Acesso em: 9-3-2020.

VITALINO, J. F. N.; CASTRO, M. A. N. [S.l.]: Brasport, 2016.

W3TECHS. **Usage Statistics and Market Share of Plone**. 2019. Disponível em: ⟨https://w3techs.com/technologies/details/cm-plone⟩. Acesso em: 2-12-2019.

ZAPATA, C. M.; TAMAYO, P. A.; ARANGO, F. *et al.* Conversión de esquemas preconceptuales a diagrama de casos de uso empleando atom3. **Dyna**, Universidad Nacional de Colombia, v. 74, n. 153, p. 237–251, 2007.

APÊNDICE A – PESQUISA ENTRE PESSOAS DA COMUNIDADE

Associação Kuruatuba

Este questionário tem o intuito de coletar informações gerais sobre atividades desempenhadas pela associação e sua importância para a sociedade.

*Obrigatório

Anonimato

Todas as respostas obtidas serão utilizadas apenas para avaliação, não existe risco de informações serem passadas para outras organizações. Suas respostas serão mantidas anônimas! A Diretoria da Kuruatuba agradece sua participação, ela é muito importante para nós.

Responsáveis pelo projeto

Os responsáveis por tal questionário são vinculados à UFVJM, sendo um aluno, Guilherme, e um docente, Erinaldo. Para mais informações, estão disponibilizados os e-mails de ambos.

Erinaldo: erinaldo.silvaifnmg@gmail.com
Guilherme: gleite98@gmail.com

Informações pessoais

1. **Qual a sua faixa etária de idade?** *
   Marcar apenas uma oval.
   - ◯ Até 20 anos
   - ◯ 21 - 30
   - ◯ 31 - 40
   - ◯ 41 - 50
   - ◯ 51 - 60
   - ◯ Mais que 60 anos

2. **Sexo** *
   Marcar apenas uma oval.
   - ◯ Feminino
   - ◯ Masculino

3. **Estado Civil** *
   Marcar apenas uma oval.
   - ◯ Solteiro
   - ◯ Casado
   - ◯ Divorciado
   - ◯ Outro: _____

Sobre a Kuruatuba

4. **Você é um associado?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

5. **Sim, há quanto tempo? Se não, por que ainda não se associou?**

_____

_____

_____

_____

_____

Sem título

6. **Você costuma participar dos eventos da Kuruatuba?** *
Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 7.*
- ◯ Não

Sobre a Kuruatuba

7. **Em quais destes eventos você teve participação?** *
Marque todas que se aplicam.

- ☐ Nunca participei
- ☐ Guisado do fofo e outros eventos gastronômicos
- ☐ Beach Soccer (pelada de domingo)
- ☐ Plantio de árvores
- ☐ Copa, torneio ou Campeonato de Beach Soccer
- ☐ Reunião/Assembleia
- ☐ Outro: _____

8. **Você tem algum comentário que gostaria de fazer a respeito desses eventos?**

_____

_____

_____

_____

_____

Ir para a pergunta 9.

Planejamento de eventos

9. **Quais destes eventos você gostaria que a Kuruatuba proporcionasse?**
   Marque todas que se aplicam.

   - [ ] Cursos sobre esportes de areia (beach soccer, futevôlei, futebol de areia, etc).
   - [ ] Intercâmbio esportivo (viagens para jogar noutras cidades)
   - [ ] Rifas, sorteios, shows musicais, etc
   - [ ] Capacitação e eventos na área ambiental

10. **Outro evento? Qual?**

    _____

11. **De modo geral, qual o seu grau de satisfação com a nossa Kuruatuba?** *
    Marcar apenas uma oval.

	1	2	3	4	5	
Insatisfeito	◯	◯	◯	◯	◯	Satisfeito

12. **Com que frequência você visita as praias do rio Gorutuba?** *
    Marcar apenas uma oval.

    - ◯ Diariamente *Ir para a pergunta 13.*
    - ◯ Uma vez por semana *Ir para a pergunta 13.*
    - ◯ Uma vez por mês *Ir para a pergunta 13.*
    - ◯ Uma vez por ano *Ir para a pergunta 13.*
    - ◯ Nunca *Ir para a pergunta 15.*

Ir para a pergunta 13.

Sobre as praias do rio Gorutuba

13. **Com quem você prefere ir ao rio Gorutuba** *
    Marcar apenas uma oval.

    - ◯ Sozinho *Após a última pergunta desta seção, ir para a pergunta 15.*
    - ◯ Com o cônjugue *Após a última pergunta desta seção, ir para a pergunta 15.*
    - ◯ Com familiares *Após a última pergunta desta seção, ir para a pergunta 15.*
    - ◯ Com amigos *Após a última pergunta desta seção, ir para a pergunta 15.*
    - ◯ Outro: _____

14. **Em suas palavras, o que pode ser melhorado nas praias do rio?**

    _____

    _____

    _____

    _____

    _____

Ir para a pergunta 15.

Sobre associados e representantes

15. **Os associados colaboram uns com os outros?**
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Não sou capaz de opinar

16. **Os diretores e conselheiros da associação são representativos?**
Marcar apenas uma oval.

	1	2	3	4	
Nada representativos	◯	◯	◯	◯	Extremamente representativos

Ir para a pergunta 17.

Redes sociais e notícias

17. **Você tem o hábito de utilizar redes sociais para se informar?** *
Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 18.*
- ◯ Não *Ir para a pergunta 20.*

Redes sociais e notícias

18. **Quais redes você utiliza?** *
Marque todas que se aplicam.

- ☐ Facebook
- ☐ YouTube
- ☐ Twitter
- ☐ Instagram
- ☐ Whatsapp
- ☐ Outro: _____

19. **Com qual frequência você as utiliza?** *
Marcar apenas uma oval.

- ◯ Várias vezes no dia
- ◯ Uma vez no dia
- ◯ Uma ou mais vezes na semana
- ◯ Uma ou mais vezes no mês

Ir para a pergunta 20.

Redes sociais e notícias

20. **Você tem dificuldade em receber notícias sobre a Kuruatuba?** *
Marcar apenas uma oval.

- Sim *Ir para a pergunta 22.*
- Não *Ir para a pergunta 21.*

Redes sociais e notícias

21. **Por onde você recebe as notícias?** *
Marque todas que se aplicam.

- [ ] Por site ou blog da associação
- [ ] Pelas redes sociais (Ex.: Facebook, YouTube, Twitter, etc)
- [ ] Por amigos
- [ ] Por outro meio de comunicação (Ex.: rádio, jornal, etc)
- [ ] Outro: _____

Redes sociais e notícias

22. **Por onde você gostaria de receber essas notícias?** *
Marque todas que se aplicam.

- [ ] Por site ou blog da associação
- [ ] Pelas redes sociais (Ex.: Facebook, YouTube, Twitter, etc)
- [ ] Por amigos
- [ ] Por outro meio de comunicação (Ex.: rádio, jornal, etc)
- [ ] Outro: _____

23. **Qual a primeira palavra que lhe vem à cabeça quando fala ou escuta falar da Kuruatuba?**

_____

APÊNDICE B – COLETA DE REQUISITOS PARA O SISTEMA

Associação Kuruatuba

Este questionário tem o intuito de coletar requisitos e especificações para o sistema que será desenvolvido para a associação Kuruatuba. O mesmo foi elaborado para receber sugestões de todos aqueles que futuramente manipularão o sistema e é totalmente anônimo, somente importando as opiniões nele registradas.
*Obrigatório

Responsáveis pelo projeto

Os responsáveis por tal questionário são vinculados à UFVJM, sendo um aluno, Guilherme, e um docente, Erinaldo. Para mais informações, estão disponibilizados os e-mails de ambos.

Erinaldo: erinaldo.silvaifnmg@gmail.com
Guilherme: gleite98@gmail.com

Perguntas

1. Abaixo se encontram os principais requisitos já elaborados, marque aquele(s) que você não considera(m) importante(s). *

Marque todas que se aplicam.

- [ ] Sistema de login e cadastro para os usuários
- [ ] Opção de publicar notícias sobre a associação
- [ ] Opção de cadastrar associados
- [ ] Sistema para gerar carteirinha para associados
- [ ] Formulário para receber mensagens dos visitantes (sobre dúvidas, elogios, sugestões, etc)
- [ ] Opção de publicar eventos organizados pela associação
- [ ] Link para baixar o estatuto da Kuruatuba
- [ ] Opção de exibição, alteração e remoção de associados
- [ ] Nenhum dos requisitos citados

2. São as que considero básicas *

3. Nos dê mais sugestões de funcionalidades que poderiam existir no site da Kuruatuba!