

CENTRO DE EDUCAÇÃO PROFISSIONAL DE TIMBÓ
CURSO TÉCNICO DE INFORMÁTICA COM HABILITAÇÃO EM
DESENVOLVIMENTO DE SOFTWARE

PROTOTIPO DE SISTEMA ONLINE DE
ACOMPANHAMENTO DE ENCHENTES E INUNDAÇÕES

JONATHAN ELI SUPTITZ
LUAN CARLOS PURIM
ROBERTO LUIZ DEBARBA

TIMBÓ
2014

PROTÓTIPO DE SISTEMA ONLINE DE

ACOMPANHAMENTO DE ENCHENTES E INUNDAÇÕES

Por

JONATHAN ELI SUPTITZ

LUAN CARLOS PURIM

ROBERTO LUIZ DEBARBA

Trabalho aprovado em sua forma final pelo
Curso Técnico de Informática com Habilitação
em Desenvolvimento de Software do Centro
de Educação Profissional de Timbó – CEDUP

Presidente: Prof. Marco Antônio Spiess – Orientador

Membro: Prof. Douglas Ropelato

Membro: Prof. Edésio Marcos Slomp

Membro: Prof. Monica Andreia Schneider

Membro: Prof. Sandro Gumz

Timbó, 04 de dezembro de 2014

AGRADECIMENTOS

Aos professores, por seu incentivo e auxílio incansável.

Ao orientador, pelo seu excelente direcionamento e paciência.

Aos familiares, por sua paciência e incentivo a conclusão desta jornada.

Aos amigos, pelo apoio ao tema apresentado.

Ao coordenador do curso, por seu excepcional trabalho e dedicação para tornar este curso possível.

Aos companheiros de trabalho da Senior Sistemas, por suas intermináveis ideias e orientações.

A todos que de alguma forma, direta ou indiretamente, contribuíram com o desenvolvimento e conclusão deste trabalho.

RESUMO

Nos últimos anos, com o crescimento populacional na região do vale do Itajaí, identificou-se um aumento no número de ocorrências de enchentes e inundações. Mesmo durante eventos pluviais não extremos, tem-se observado um maior número de pessoas afetadas. Tendo como objetivo prevenir e minimizar os danos causados por esses desastres naturais, esse trabalho apresenta um sistema online de acompanhamento de enchentes e inundações. O sistema proporcionará ferramentas onde os usuários poderão acompanhar o status de inundaçāo de sua região através de um site interativo ou aplicativo móvel, com informações obtidas de forma automatizada, por uma régua de medição de estado de chuva e nível do rio desenvolvida na ferramente de prototipação de hardware livre Arduino. Os usuários também contarão com mapas de inundações interativos e funções de notificação e alerta. No mercado existem múltiplas soluções do gênero, porém normalmente possuem baixa quantidade de recursos e difícil compreensão das informações expostas. O desenvolvimento do trabalho iniciou-se com uma pesquisa de mercado para levantamento de requisitos, considerando-se a necessidade dos usuários, viabilidade da implementação e custo de implantação. A seguir foram desenvolvidas as especificações, com partes como diagrama de atividades e casos de uso. Todas as etapas podem ser vistas no decorrer deste. A implementação do código fonte deu-se principalmente com as linguagens Java, HTML5, PHP e banco de dados não relacional MongoDB. Serão abordados formas de utilização das tecnologias empregadas, exemplos de código, bibliotecas, *frameworks* e a topologia utilizada, desde o ponto de usuário até o servidor de aplicação e dados.

Palavras-chave: Automação. Enchentes. Monitoramento.

SUMÁRIO

1 INTRODUÇÃO.....	7
1.1 OBJETIVOS DO TRABALHO.....	8
1.1.1 Objetivo Geral.....	8
1.1.2 Objetivos Específicos.....	8
1.2 ESTRUTURA DO TRABALHO.....	8
2 FUNDAMENTAÇÃO TEÓRICA.....	9
2.1 ENCHENTES E INUNDAÇÕES.....	9
2.1.1 Enchentes e Inundações no Vale do Itajaí.....	11
2.2 PROTOTIPAÇÃO.....	12
2.2.1 Rascunhos.....	13
2.2.2 Protótipos Visuais.....	13
2.2.3 Protótipos Interativos.....	13
2.3 PROGRAMAÇÃO EXTREMA.....	13
2.3.1 Principais Práticas.....	14
2.4 GEOPROCESSAMENTO.....	15
2.5 AMAZON WEB SERVICES.....	16
2.5.1 Computação Em Nuvem.....	17
2.5.2 Infraestrutura Como Serviço.....	18
2.5.3 Pagamento por Uso.....	18
2.6 MONGODB.....	18
2.6.1 Banco de Dados Não Relacional.....	20
2.6.2 Escalabilidade.....	21
2.6.3 Desempenho.....	21
2.6.4 JavaScript Object Notation (JSON).....	23
2.7 NAVEGADORES DE INTERNET.....	23
2.8 HTML.....	26
2.9 HTML5.....	27
2.10 CSS.....	28
2.11 JAVASCRIPT.....	29
2.12 JQUERY.....	30
2.13 AJAX.....	31

2.14 PHP.....	32
2.15 BOOTSTRAP.....	33
2.16 GOOGLE MAPS API.....	33
2.17 GOOGLE CHARTS API.....	35
2.18 FACEBOOK DEVELOPERS.....	36
2.19 JAVA.....	36
2.19.1 Características.....	37
2.19.2 Java Virtual Machine (JVM).....	37
2.20 LAMP.....	39
2.21 XAMPP.....	39
2.22 DAEMON (SERVIÇOS).....	40
2.22.1 Origem do Termo.....	40
2.22.2 Java Service Wrapper.....	41
2.23 SOCKET.....	41
2.24 WEB SERVICES.....	42
2.25 AXIS 2.....	42
2.25.1 Simple Object Access Protocol (SOAP).....	43
2.26 APACHE TOMCAT.....	44
2.27 APACHE WEB SERVER.....	45
2.28 ARDUINO.....	46
2.28.1 Arduino Uno.....	46
2.28.1.1 Especificações.....	47
2.28.2 Arduino IDE.....	48
2.28.3 Ethernet Shield.....	50
2.28.4 Sensor Ultrassônico HC SR-04.....	50
2.28.5 Sensor de Chuva YL-83.....	51
2.29 PHONEGAP.....	52
2.30 GENYMOTION.....	53
2.31 ECLIPSE.....	53
2.32 APTANA STUDIO.....	53
2.33 ROBOMONGO.....	54
2.34 STARUML.....	55
2.34.1 Unified Modeling Language (UML).....	55
2.35 PENCIL PROJECT.....	56

2.36 FRITZING.....	56
2.37 GIT.....	57
2.38 GITHUB.....	57
2.39 LIBREOFFICE.....	58
3 DESENVOLVIMENTO.....	59
3.1 REQUISITOS.....	59
3.1.1 Requisitos Funcionais.....	59
3.1.2 Requisitos Não Funcionais.....	62
3.2 ESPECIFICAÇÃO.....	63
3.2.1 Casos de Uso.....	63
3.2.1.1 Diagrama de Casos de Uso.....	64
3.2.1.2 Especificação dos Casos de Uso.....	64
3.2.2 Diagrama de Atividades.....	68
3.3 IMPLEMENTAÇÃO.....	70
3.3.1 Prototipação do software.....	70
3.3.2 Programação Extrema.....	72
3.3.3 Ponto de Medição Automatizado.....	72
3.3.4 Serviço de Controle de Leituras.....	74
3.3.5 Banco de Dados.....	77
3.3.6 Web Service.....	79
3.3.7 Site.....	81
3.3.8 Aplicativo Móvel.....	82
3.3.9 Mapa de Inundações.....	83
3.3.9.1 Simulação de Inundações.....	85
3.3.10 Níveis Atualizados e Alertas.....	85
3.3.11 Histórico de Medidas.....	87
3.3.12 Previsão do Tempo.....	88
3.3.13 Galeria Colaborativa.....	90
3.3.14 Verificador de Vulnerabilidade de Locais.....	90
3.3.15 Sistema de Alerta.....	92
3.3.16 Servidor em Nuvem.....	94
4 CONCLUSÕES.....	96
4.1 EXTENSÕES.....	97

1 INTRODUÇÃO

Nos últimos anos, com o crescimento populacional na região do vale do Itajaí, identificou-se um aumento no número de ocorrências de enchentes e inundações. Mesmo durante eventos pluviais não extremos, tem-se observado um maior número de pessoas afetadas. As características sociais e ambientais de diversas cidades brasileiras, tais como a impermeabilização excessiva do solo ocupação de fundos de vales e áreas de inundação, acarretaram problemas crônicos de enchentes nos últimos anos, causando grandes impactos em áreas urbanas, prejudicando as condições de vida e provocando prejuízos econômicos.

Segundo o informativo de prevenção de desastres da editora Organic Trading (2006), para cada um real investido em prevenção, equivale em média entre vinte cinco e trinta reais em obras de reconstrução pós-evento.

Um sistema de monitoramento e alerta para eventos meteorológicos extremos têm o objetivo de minimizar os impactos causados. As Tecnologias da Informação (TI) vêm tendo grande repercussão em vários setores, principalmente aplicados Ciências da Terra e, exercendo o papel de integrar as partes que compõem um sistema de emergência. (TUCCI, 2005, p. 308)

No mercado existem múltiplas soluções do gênero, porém normalmente possuem baixa quantidade de recursos e difícil compreensão das informações expostas. Como alternativa, no decorrer deste trabalho é apresentado um Sistema Online de Acompanhamento de Enchentes e Inundações, que entrega ferramentas de acompanhamento de nível fluvial com mapa interativo de visualização do estado da inundação, aplicativo de fácil acesso, notificações de alerta e equipamentos de medição automatizados.

Diante do objetivo exposto, será desenvolvida uma integração entre diversas tecnologias e produtos disponíveis no mercado. Dentre elas, a plataforma de prototipação de *hardware* livre, Arduino; serviço de processamento de dados em Oracle Java 7 , hospedado em *Cloud Computing* (computação em nuvem), na Amazon Web Services (conjunto de serviços globais de computação, armazenamento, banco de dados, análise, aplicativos e implementações); mapa interativo utilizando a API (*Application Programming Interface – Interface de Programação de Aplicações*) Google Maps, combinada com as linguagens para WEB HTML5 (*HyperText Markup Language – Linguagem de Marcação de Hipertexto*) e PHP (*PHP: Hypertext Preprocessor*); banco de dados não relacional MongoDB; entre outras.

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivo Geral

Desenvolver um protótipo sistema online de acompanhamento de enchentes, onde os usuários possam acompanhar o status de inundação de sua região através de um site interativo ou aplicativo móvel, com informações obtidas de forma automatizada.

1.1.2 Objetivos Específicos

- Disponibilizar uma ferramenta online de acompanhamento de enchentes.
- Permitir o acesso através de navegadores de internet ou aplicativos móveis.
- Desenvolver um mapa interativo que apresentará o estado de alagamento.
- Automatizar a medição do nível de rios, disponibilizando a informação via internet.
- Entregar um serviço de notificações de alerta de enchentes via aplicativo para dispositivos móveis.

1.2 ESTRUTURA DO TRABALHO

O primeiro capítulo introduz o leitor ao sistema online de acompanhamento de enchentes e inundações, apresentando os objetivos esperados em seu desenvolvimento.

O segundo capítulo apresenta o fundamento e contextualização teórica sobre os temas, linguagens de programação, ferramentas e técnicas utilizadas no decorrer deste trabalho.

O terceiro capítulo trata o desenvolvimento dos objetivos apresentados. São demonstrados os requisitos e especificações detalhadas do *software*, com diagramas de casos de uso e atividades. Também serão apresentados detalhes da implementação, com exemplos de código e diagramas estruturais e lógicos.

Finalizando, no quarto capítulo são apresentadas as conclusões sobre os temas abordados, descrevendo as dificuldades encontradas e sugestões para futuras implementações.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ENCHENTES E INUNDAÇÕES

As inundações no Brasil são fenômenos que ocorrem em diversas regiões do país, caracterizados geralmente por chuvas intensas e contínuas. O fenômeno é frequente e pode ser o resultado de uma chuva que não foi suficientemente absorvida pelo solo e outras formas de escoamento, causando transbordamentos.

A inundação é um caso extremo de aumento da vazão que resulta de um pequeno período de desequilíbrio entre a entrada e saída de água. A medida que a vazão aumenta, a velocidade do fluxo também sobe, e a água gradualmente preenche o canal. Com o aumento contínuo da vazão, o rio atinge o estágio da inundação (o ponto em que a água extravasa sobre as margens). (GROTZINGER; JORDAN, 2013, p. 524)

Popularmente usa-se o termo enchentes, que é geralmente uma situação natural de transbordamento de água do seu leito original. Já as inundações, que podem também ser denominadas de alagamentos, estão relacionadas diretamente com a população, causando danos, prejuízos e modificações dos elementos naturais e artificiais de um aglomerado urbano.



IMAGEM 01: Definição de termos. Fonte: Defesa Civil de São Bernardo do Campo (2011)

Segundo a Defesa Civil de São Bernardo do Campo, São Paulo, uma inundação representa o transbordamento das águas de um curso d'água, atingindo a planície de inundaçāo ou área de várzea. As enchentes ou cheias são definidas pela elevação do nível

d'água no canal de drenagem devido ao aumento da vazão, atingindo a cota máxima do canal, porém, sem extravasar. O alagamento é um acúmulo momentâneo de águas em determinados locais por deficiência no sistema de drenagem. A enxurrada é escoamento superficial concentrado e com alta energia de transporte, que pode ou não estar associado a áreas de domínio dos processos fluviais. Vide a ilustração na IMAGEM 01:

É importante compreender que as enchentes dos rios são fenômenos naturais, que ocorrem com frequência variável e muitas vezes inesperada. Em muitas situações, o leito maior do rio é ocupado, principalmente em locais onde as enchentes demoram a acontecer novamente, fazendo com que a enchente do rio se transforme em inundação, com perdas humanas e patrimoniais.

[...] Uma das maiores dificuldades quanto à prevenção dos impactos pluviais reside no fato de que toda a estrutura urbana é “planejada” usando como parâmetro apenas o estado médio dos fenômenos meteorológicos, desconsiderando as anomalias que fazem parte do clima local, e que fatalmente ocorrem. (PEREZ FILHO et al., 2006, p. 44)

Nos últimos anos, com o crescimento populacional na região do vale do Itajaí, identificou-se um aumento de ocorrências de enchentes e inundações. Mesmo durante eventos pluviais não extremos, tem-se observado um maior número de pessoas afetadas.

As características sociais e ambientais de diversas cidades brasileiras, tais como a impermeabilização excessiva do solo e parcela considerável da população ocupando fundos de vale e áreas de inundação, acarretaram problemas crônicos de enchentes nos últimos anos, causando grandes impactos em áreas urbanas, prejudicando as condições de vida da população e provocando prejuízos econômicos. (PEREZ FILHO et al., 2006, p. 44)

Algumas obras podem ser realizadas para controle das inundações no meio urbano, tais como construção e manutenção de bueiros, barragens de defesa contra inundações, valas, tanques de contenção ou ainda obras de revitalização de rios, muito utilizadas na Holanda e na Alemanha. Também é recomendada a utilização de ferramentas de acompanhamento e prevenção. Segundo Kobiyama et al. (2006), para cada um real investido em prevenção, equivale em média entre vinte cindo e trinta reais em obras de reconstrução pós-evento.

Um sistema de monitoramento e alerta para eventos meteorológicos extremos têm o objetivo de minimizar os impactos causados. As Tecnologias da Informação (TI) vêm tendo grande repercussão em vários setores, principalmente aplicados Ciências da Terra e, exercendo o papel de integrar as partes que compõem um sistema de emergência. (TUCCI, 2005, p. 308)

2.1.1 Enchentes e Inundações no Vale do Itajaí

Os problemas de longa data que compreendem o histórico das enchentes sucedidas na Bacia Hidrográfica do Itajaí em Santa Catarina, somado ao processo de degradação do solo foi se construindo junto à história da colonização ocorrida a partir do ano de 1850. A história de Blumenau pode ser pautada sob a ótica das enchentes. Desde o período inicial da colonização já se contabilizam mais de 60 cheias (vide ANEXO B para consultar os picos de enchentes na região). Grande parte das funções vitais da cidade concentra-se em áreas inundáveis, bem como inúmeras construções residenciais.

Estado	Região	Sigla	total desastres 2008	inundações 2008	vítimas 2008*
Distrito Federal	Centro-Oeste	DF	3	2	900
Acre	Norte	AC	3	3	13826
Alagoas	Nordeste	AL	7	4	2130
Amapá	Norte	AP	4	2	7500
Amazonas	Norte	AM	16	8	48806
Bahia	Nordeste	BA	13	3	6417
Espirito Santo	Sudeste	ES	37	21	78778
Goiás	Centro-Oeste	GO	3	2	84029
Maranhão	Nordeste	MA	73	70	176106
Mato Grosso	Centro-Oeste	MT	27	12	80338
Mato Grosso do Sul	Centro-Oeste	MS	10	5	29230
Minas Gerais	Sudeste	MG	67	32	94990
Paraíba	Norte	PA	32	25	175327
Paraná	Sul	PR	31	0	0
Pernambuco	Nordeste	PE	4	2	800
Rio De Janeiro	Sudeste	RJ	64	39	287667
Rio Grande do Norte	Nordeste	RN	94	38	234736
Rio Grande do Sul	Sul	RS	146	25	37134
Rondonia	Norte	RO	4	3	2014
Roraima	Norte	RR	0	0	0
Santa Catarina	Sul	SC	94	51	94071
Sergipe	Nordeste	SE	5	2	2060
Tocantins	Norte	TO	8	0	0
Sao Paulo	Sudeste	SP	16	2	1450
Ceará	Nordeste	CE	46	29	106206
Piauí	Nordeste	PI	21	19	112042
Paraíba	Nordeste	PB	3	3	12139

QUADRO 01: Ocorrências de desastres registrados pela Defesa Civil. Fonte: Silva (2010)

O município de Blumenau (SC) sofreu no mês de novembro de 2008 uma das maiores catástrofes naturais já registradas em toda sua história. Esta catástrofe foi o resultado de uma soma de diversos fatores, sendo principal causadora a enorme quantidade de precipitação registrada em curto espaço de tempo. Inicialmente devido a alta intensidade de chuva, ocorreu uma enxurrada alagando diversos pontos da cidade, após veio a enchente do rio Itajaí Açu, que atingiu um nível máximo de 11,52 m, e concomitantemente ocorreram muitos

deslizamentos de terra em diversos pontos do município. Neste evento, a pluviosidade máxima registrada em um dia foi 250,9 mm, em dois dias foi de 494,40 mm e a soma do mês de novembro foi de 1001,7 mm. Nos 65 anos de registros existentes, nunca havia sido registrada uma pluviosidade dessa magnitude na cidade.

Analizando os dados sobre ocorrências de desastres registrados pela Defesa Civil (QUADRO 01), é possível inferir que em 2008 ocorreram 831 desastres no Brasil, destes eventos, 403 foram inundações, que vitimaram o total de 1.533.524 pessoas. As regiões Sul e Sudeste merecem destaque nesta análise, pois ambas concentraram no período de análise, 55% do total de desastres ocorridos no Brasil, e 594.100 vítimas de inundações. (SILVA, 2010, p. 12)

2.2 PROTOTIPAÇÃO

Prototipação é um processo que tem como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software. Desta forma, pode-se propor uma solução adequada para o problema do cliente, aumentando sua percepção de valor. É uma abordagem baseada em uma visão evolutiva do desenvolvimento de software, afetando o processo como um todo. Envolve a produção de versões iniciais de um sistema futuro com o qual é possível realizar verificações e experimentos, com o intuito de avaliar algumas de suas características antes que o sistema venha realmente a ser construído de forma definitiva.

A prototipação envolve a produção de uma versão limitada do produto com a finalidade de responder a perguntas específicas [...]. Os protótipos fornecem uma melhor impressão da experiência do usuário do que simples descrições, e existem diferentes tipos de protótipos que são adequados para diferentes estágios de desenvolvimento e para transmitir diferentes tipos de informação. (CHAPMAN, 2013, p. 340)

Os protótipos são grandes aliados das metodologias ágeis de desenvolvimento, uma vez que garantem maior alinhamento entre a equipe e o cliente. Eles podem ser desenvolvidos em diferentes níveis de fidelidade: quanto maior ela for, mais o protótipo se assemelhará ao resultado entregue. No entanto, um protótipo de alta-fidelidade leva mais tempo para ser criado ou modificado. A escolha do protótipo ideal varia de acordo com o nível de entendimento do negócio, a complexidade dos requisitos, prazo e orçamento para elaboração. Um protótipo promove a identificação antecipada de erros e modificações, reduzindo o custo de desenvolvimento.

2.2.1 Rascunhos

São protótipos de baixa fidelidade, rápidos para se desenvolver e modificar. Rascunhos não vão mostrar detalhes visuais ou interações de tela, mas vão ajudar a validar requisitos e regras de negócio de maneira eficiente. Esse modelo é a melhor escolha para representar cenários complexos onde um fluxo ou processo precisa ser compreendido.

2.2.2 Protótipos Visuais

Criados com programas de edição gráfica, estes protótipos têm maior apelo visual, entretanto, não possuem interações de tela e demandam mais tempo para se fazer ajustes e melhorias. São uma ótima opção para telas com maior ênfase em estética e usabilidade, quando os requisitos já foram entendidos.

2.2.3 Protótipos Interativos

São protótipos completos e representativos. Além da parte visual, englobam uma série de detalhes de estética e efeitos de interação, proporcionando uma experiência rica e realista. Também ajudam a equipe a identificar novos requisitos, oportunidades e futuros problemas. As consequências destes para o software são lucros maiores a longo prazo e riscos menores durante o desenvolvimento. Em contrapartida, protótipos interativos demandam uma equipe de maior conhecimento técnico e demoram mais tempo para serem criados.

2.3 PROGRAMAÇÃO EXTREMA

Programação extrema (*eXtreme Programming – XP*), é uma metodologia ágil para equipes pequenas e médias e que necessitam desenvolver *softwares* com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento do projeto.

Seria errado concluir que tudo que é necessário para desenvolver *software* é programação desenfreada. Desenvolver *software* é difícil, e desenvolver *software* de qualidade no prazo combinado é ainda mais difícil. Para funcionar, é preciso o uso disciplinado de práticas de modelo (*best practices*) adicionais. (BECK, 2004, p. 3)

Os cinco valores fundamentais da metodologia XP são: comunicação, simplicidade, *feedback*, coragem e respeito. A partir desses valores, possui como princípios: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade.

Para aplicar os valores e princípios durante o desenvolvimento de software, XP propõe uma série de práticas. Há uma confiança muito grande na sinergia entre elas, os pontos fracos de cada uma são superados pelos pontos fortes de outras.

2.3.1 Principais Práticas

- Jogo de Planejamento (*Planning Game*): O desenvolvimento é feito em iterações curtas. No início, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planejamento. Nela, o cliente identifica prioridades e os desenvolvedores as estimam. Ao final de cada jogo, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção.
- Fases pequenas (*Small Releases*): A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando.
- Design Simples (*Simple Design*): Simplicidade é um princípio da XP. Projeto simples significa dizer que caso o cliente tenha pedido que na primeira versão apenas o usuário “teste” possa entrar no sistema com a senha “123” e assim ter acesso a todo o sistema, você vai fazer o código exato para que esta funcionalidade seja implementada, sem se preocupar com sistemas de autenticação e restrições de acesso.
- Time Coeso (*Whole Team*): A equipe de desenvolvimento é formada por pessoas engajadas e de forma multidisciplinar.
- Propriedade Coletiva (*Collective Ownership*): O código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema.

- Reuniões em pé (*Stand-up Meeting*): Reuniões rápidas para não perder o foco nos assuntos, apenas abordando tarefas realizadas e tarefas a realizar pela equipe.
- Programação Pareada (*Pair Programming*): é a programação em par/dupla num único computador. Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor.
- Padronização do Código (*Coding Standards*): A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras.
- Desenvolvimento Orientado a Testes (*Test Driven Development*): Primeiro crie os testes unitários (*unit tests*) e depois crie o código para que os testes funcionem.
- Refatoração (*Refactoring*): É um processo que permite a melhoria contínua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refazer melhora a clareza do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte;
- Integração Contínua (*Continuous Integration*): Sempre que produzir uma nova funcionalidade, nunca esperar uma semana para integrar à versão atual do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte.

2.4 GEOPROCESSAMENTO

Geoprocessamento é o tratamento das informações geográficas, ou de dados georreferenciados, por meio de softwares específicos e cálculos. Ou, ainda, o conjunto de técnicas relacionadas ao tratamento da informação espacial. O geoprocessamento consiste nas seguintes etapas: coleta, armazenamento, tratamento e análise de dados e uso integrado das informações.

As tecnologias que são englobadas nesta concepção, e que a cada momento fazem cada vez mais parte do nosso dia a dia, são o Sensoriamento Remoto (SR), o Sistema de Informação Geográfica (SIG) e o Sistema de Posicionamento Global (GPS), este último mais conhecido pela sua sigla em inglês.

A cada ano, as aplicações das tecnologias de Geoprocessamento tornam-se mais necessárias ao desenvolvimento das sociedades que necessitem planejar e implementar o seu desenvolvimento.

2.5 AMAZON WEB SERVICES

Amazon Web Services (AWS) é um ambiente de computação em nuvem, disponibilizado pela Amazon, Inc, com características de escalabilidade, disponibilidade, elasticidade e desempenho para aplicações executadas nesse ambiente. Disponibiliza uma infraestrutura completa para computação em diversos níveis de processamento, desde tarefas simples até de alto desempenho e possui uma gerência eficaz dos recursos.

Com a AWS, você tem a flexibilidade de escolher qualquer plataforma de desenvolvimento ou modelo de programação que se adapte ao seu modelo de negócio ou solução. Seu modelo de negócio é baseado em Infraestrutura como Serviço (*Infrastructure as a Service*, IaaS) e Pagamento por Uso (*pay-per-use*).

Amazon Web Services (AWS) é a principal oferta do tipo *cloud computing* da atualidade. Essa arquitetura permite às empresas o acesso a serviços de infraestrutura de forma *on demand*. A ideia é que está forma de aquisição de serviços de infraestrutura de TI, conhecida como serviços de infraestrutura em nuvem, reduza custos, minimize os riscos do negócio e maximize as oportunidades. (VERAS, 2013, p. 7)

Para ajudar os novos clientes a começarem na nuvem, a AWS apresenta um nível de uso gratuito. Esse nível pode ser usado para: executar novos aplicativos, testar aplicativos existentes na nuvem ou simplesmente obter experiência prática com a AWS. Nessa forma são fornecidos os seguintes serviços:

1. Amazon EC2: Serviço da web que fornece uma capacidade computacional redimensionável na nuvem.
2. Amazon S3: Armazenamento de dados escalável e de baixa latência.
3. Amazon RDS: Hospedagem de bancos de dados MySQL, Oracle e SQL Server.
4. Amazon CloudWatch: Monitoramento de recursos e aplicativos da nuvem.
5. AWS Data Pipeline: Orquestração para fluxos de trabalho orientados por dados.
6. Amazon DynamoDB: Serviço de banco de dados NoSQL.
7. Amazon EBS: Volumes de armazenamento de dados.
8. Amazon ELB: Serviço da web que oferece escalabilidade e alta disponibilidade.
9. Amazon ElastiCache: Armazenamento em cache com escalabilidade horizontal.
10. Amazon SES: Serviço econômico de envio de e-mail.
11. Amazon SNS: Serviço da web para configurar, operar e enviar notificações.

12. Amazon Elastic Transcoder: Conversor de arquivos de mídia escalável.
13. Amazon SQS: Fila escalável para armazenamento de mensagens à medida que transitam entre computadores.
14. Amazon SWF: Serviço de fluxo de trabalho para a criação de aplicativos escaláveis e resilientes.
15. AWS Marketplace: Software de parceiros pré-configurado para execução na AWS.
16. Amazon CloudFront: Serviço de entrega de conteúdo na web.

2.5.1 Computação Em Nuvem

A evolução constante da tecnologia computacional e das telecomunicações está fazendo com que o acesso à internet se torne cada vez mais amplo e cada vez mais rápido. Com a *cloud computing*, muitos aplicativos, assim como arquivos e outros dados relacionados, não precisam mais estar instalados ou armazenados no computador do usuário ou em um servidor próximo. Este conteúdo passa a ficar disponível nas *nuvens*, isto é, na internet. Ao fornecedor da aplicação cabe todas as tarefas de desenvolvimento, armazenamento, manutenção, atualização, *backup*, escalonamento, etc.

[...] A Computação em Nuvem é um termo para descrever um ambiente de computação baseado em uma imensa rede de servidores, sejam esses virtuais ou físicos. Uma definição simples pode ser “um conjunto de recursos como capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e servidores disponibilizados na internet”. (TAURION, 2009, p. 2)

Um exemplo prático desta nova realidade é o Google Docs, serviço onde os usuários podem editar textos, fazer planilhas, elaborar apresentações de slides, armazenar arquivos, entre outros, tudo pela internet, sem necessidade de ter programas como Microsoft Office ou OpenOffice.org instalados em suas máquinas.

A computação em nuvem cria uma ilusão de disponibilidade de recursos infinitos, acessíveis sobre demanda.

A computação em nuvem elimina a necessidade de adquirir e provisionar recursos antecipadamente.

A computação em nuvem fornece elasticidade, permitindo que as empresas usem os recursos na quantidade que forem necessários, aumentando e diminuindo a capacidade computacional de forma dinâmica.

O pagamento de serviços em nuvem é pela quantidade de recursos utilizados (*pay-per-use*). (TAURION, 2009, p. 2)

2.5.2 Infraestrutura Como Serviço

Infraestrutura como Serviço (*Infrastructure as a Service*) é um modelo de negócio, a infraestrutura é contratada como serviço. Como vantagem em relação ao modelo tradicional, são adquiridos servidores virtuais (e outros dispositivos de infraestrutura) ao invés da compra de servidores, roteadores, *racks* e outros dispositivos de *hardware*. O contratante é tarifado por alguns fatores, como o número de servidores virtuais, quantidade de dados trafegados, dados armazenados e outros itens, dependendo do fornecedor.

2.5.3 Pagamento por Uso

Pagamento por uso (*pay-per-use*) é um modelo de negócio onde o pagamento é efetuado com base nos recursos utilizados, dispensando cobranças mensais. Esse método é geralmente utilizado em computação na nuvem, onde um contratante paga pelo processamento, armazenamento ou tráfego de internet utilizado, poupano o valor em momentos onde os serviços hospedados não são utilizados.

2.6 MONGODB

MongoDB (do inglês *humongous*, “gigantesco”) é um banco de dados de código aberto não relacional (noSQL, nonREL) orientado a documentos. É desenvolvido na linguagem de programação C++. Seus documentos utilizam o formato BSON (*Binary Script Object Notation*, formato computacional binário para armazenamento e transmissão de informações. Estende o formato JSON).

O desenvolvimento do MongoDB iniciou em outubro de 2007 pela 10gen. A primeira versão pública foi lançada em fevereiro de 2009. Por possuir código aberto, seu desenvolvimento contém participação da comunidade e sua distribuição é gratuita. O modelo de negócios da 10gen consiste em suporte e certificações.

Diversas linguagens e plataformas já possuem *drivers* (software que permite que o computador se comunique com o hardware ou com os dispositivos, ou que uma linguagem

ganhe suporte a uma ferramenta) para o MongoDB, entre elas destacam-se: C, C#, C++, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby e Scala. Além disso, o MongoDB possui binários para diversas plataformas como Windows, Mac OS X, Linux e Solaris.

Entre as empresas que já utilizam o MongoDB destacam-se: Globo.com, SourceForge, FourSquare, MailBox (serviço de e-mail do Dropbox), LinkedIn, SAP, MTV, Pearson Education, e várias outras.

O MongoDB possui consultas bastante simples de serem realizadas. Visto que não existem transações e junções, as consultas são mais fáceis de escrever e mais fáceis de ajustar.

Quando se trata de bancos de dados NoSQL, é difícil algo melhor que a facilidade de uso oferecida pelo MongoDB. Ele não é apenas bem documentado e suportado por uma ampla e prestativa comunidade, mas também é amigável a desenvolvedores com histórico em SQL – muitas consultas e uma grande quantidade de raciocínio relacional podem ser diretamente aplicadas ao MongoDB a partir do SQL –, tornando-o um sistema especialmente atrativo para iniciantes no mundo NoSQL. (WILSON, 2013, p. 45)

A seguir, serão expostos exemplos de operação básicas no MongoDB, com seus respectivos comandos. No QUADRO 02 é apresentada uma operação de inserção de dados:

```
db.contato.insert({
  1.name: 'Marco Bruno',
  2.email: 'marco.bruno.br@gmail.com',
  3.mensagem: 'Inserindo dados de Exemplo no MongoDB'
  4.})
```

QUADRO 02: MongoDB – Inserção

No QUADRO 03 é possível visualizar uma operação de consulta, que deve retornar todos os documentos na coleção, seguido por seu resultado:

```
db.contato.find() {
  1._id" : ObjectId("21252b31e3b6e54896b6c8010113123f"),
  2."name" : "Marco Bruno",
  3."email" : "marco.bruno.br@gmail.com",
  4."mensagem" : "Inserindo dados de Exemplo no MongoDB"
}
```

QUADRO 03: MongoDB – Consulta

No QUADRO 04 é apresentado um exemplo de alteração de dados:

```
db.contato.update({name: 'Marco Bruno'},
{$set: {email: 'marco.bruno@pinceladasdaweb.com.br'}})
```

QUADRO 04: MongoDB – Alteração

Finalmente, no QUADRO 05, é apresentada uma remoção com parâmetro:

```
db.contato.remove({name: 'Marco Bruno'})
```

QUADRO 05: MongoDB – Remoção

2.6.1 Banco de Dados Não Relacional

O termo NoSQL foi usado pela primeira vez em 1998, como o nome de um banco de dados relacional de código aberto que não possuía uma interface SQL (linguagem de pesquisa declarativa padrão para banco de dados relacional). Seu autor, Carlo Strozzi, alega que o movimento NoSQL “é completamente distinto do modelo relacional e, portanto, deveria ser mais apropriadamente chamado ‘NoREL’ ou algo que produzisse o mesmo efeito”.

Esses bancos de dados também são chamados de Bancos NoSQL (*Not Only SQL*). Esse termo é devido à ausência do SQL, mas esse tipo de banco de dados não se resume apenas a isso, por isso o termo não é o mais correto. No entanto, ele é aceito por sua popularização na comunidade. De forma resumida, esse tipo de banco de dados não traz consigo as ideias do modelo relacional e nem a linguagem SQL.

Uma diferença fundamental entre os dois modelos surge quando precisamos criar relacionamentos nos bancos de dados relacionais. Os bancos orientados a documentos não fornecem relacionamentos entre documentos, o que mantém seu design sem esquemas. Desta forma, em vez de armazenar dados relacionados em uma área de armazenamento separado, os bancos NoSQL integram esses dados ao próprio documento. Enquanto no modelo relacional somos obrigados a pensar em evitar o tempo todo a redundância de dados, motivo pelo qual existem os relacionamentos, no MongoDB temos uma situação diferente em que não há relacionamentos e a duplicação por vezes é até mesmo incentivada como nos casos observados acima onde os dados são armazenados do jeito que quisermos.

Os bancos de dados NoSQL não utilizam esquemas de tabela fixa e, geralmente, não suportam instruções e operações de junção SQL. NoSQL atende a necessidade crescente de promover serviços escaláveis. Bancos de dados NoSQL não visam eliminar bancos de dados relacionais do tipo SQL, mas oferecer uma alternativa a esses bancos. (WILSON, 2013, p. 19)

No mercado atual, existem diversos bancos de dados NoSQL. Os exemplos mais conhecidos são: MongoDB, DynamoDB, Azure Table Storage, Berkeley DB, Hadoop,

Cassandra, Hypertable, Amazon SimpleDB, CouchDB, RavenDB, Neo4J, Infinite Graph e InforGrid.

2.6.2 Escalabilidade

A utilização de bancos de dados não relacionais tende a se tornar mais barata com o aumento da quantidade de dados armazenados. Enquanto bancos relacionais são projetados para trabalhar sobre uma máquina, modelos não relacionais possuem otimização para escalonamento horizontal. No primeiro caso, para realizar uma melhoria de performance no servidor do banco de dados, é necessário melhorar seus componentes, com troca de memória ou processador. Esse modelo é chamado escalonamento vertical. Já no escalonamento horizontal, um *upgrade* consiste em adicionar mais máquinas ao servidor, permitindo que o banco de dados dívida seus processos entre vários processadores e dispositivos de memória.

[...] Um sistema escalável horizontal significa adicionar mais nós ao sistema para crescimento, tais como adicionar um novo servidor a um sistema de banco de dados em *cluster*. O scale-out (escalonamento horizontal) é uma arquitetura de software otimizada pelo hardware. Normalmente essa modalidade apresenta menor investimento inicial e é mais flexível a introdução de novas tecnologias. (VERAS, 2011, p. 125)

2.6.3 Desempenho

Com o aumento da quantidade e do fluxo de informações e a certeza de que sempre haverá mais dados para armazenar, o tradicional modelo relacional começa a sofrer limitações de escalabilidade. A ideia do MongoDB é que tenhamos documentos autocontidos obtendo todas as informações que necessitamos sem que seja necessário realizarmos vários *joins*, dessa forma fazemos apenas uma consulta e o retorno será o documento inteiro com todas as informações resultando num ganho significativo de performance.

A seguir será apresentado um teste de performance realizado por Politowski e Maran (2014), onde é possível visualizar seus resultados comparado ao banco de dados relacional Postgre. Os testes foram divididos em três categorias: inserção, busca simples e busca complexa. Cada teste foi executado em uma repetição de 1, 10, 100, 1.000, 10.000 e 100.000 vezes:

“[...] Utilizando os dois bancos de dados na configuração padrão, sem ajustes de otimização e baseado no ambiente de testes proposto, podemos concluir que o MongoDB obteve melhores resultados (IMAGEM 02 e IMAGEM 03). Isso se dá pelo fato de que os bancos NoSQL terem nascido para suprir a demanda por performance, deixando outros detalhes, como atomicidade, por exemplo, em segundo plano.

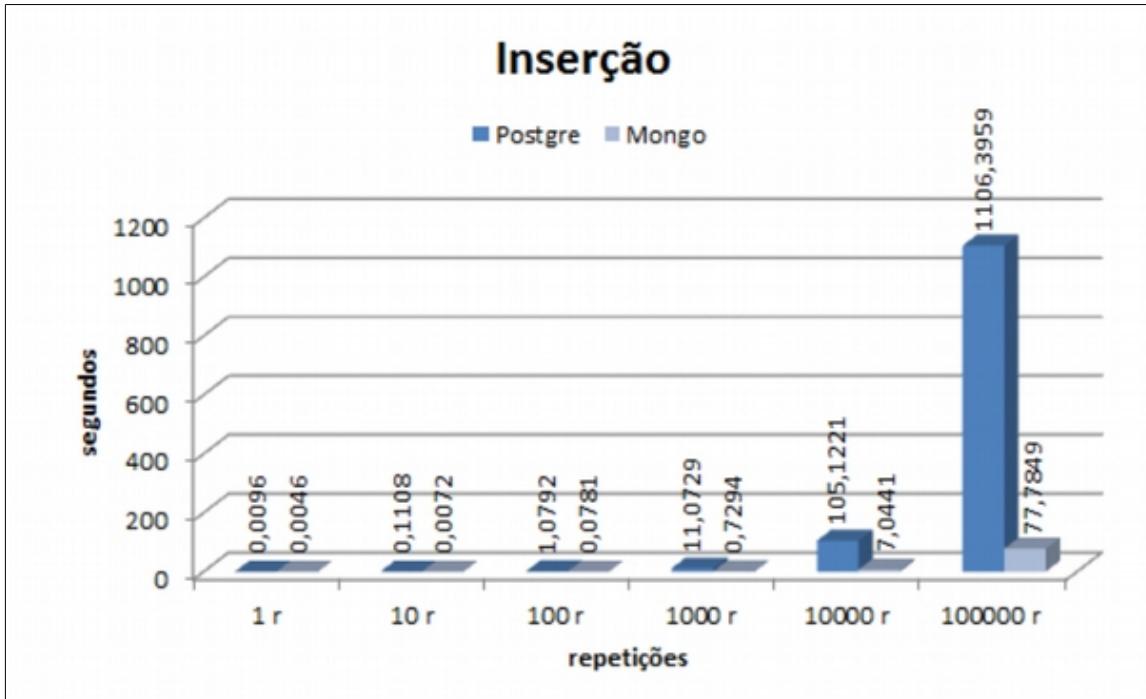


IMAGEM 02: Restado Inserção. Fonte: Politowski e Maran (2014)

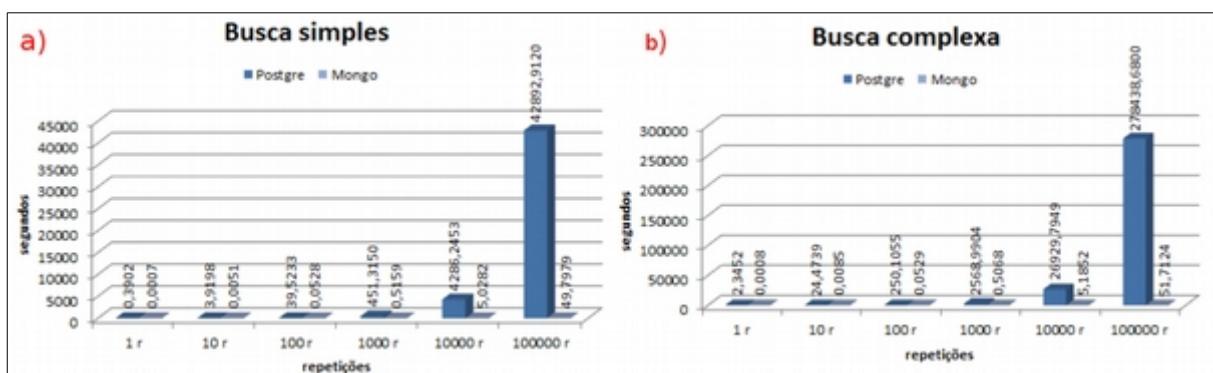


IMAGEM 03: Restado Busca. Fonte: Politowski e Maran (2014)

Bancos de dados NoSQL e Relacionais utilizam paradigmas diferentes e, por sua vez, possuem finalidades diferentes, mas com o mesmo propósito: persistir dados. Segundo os testes de performance, para uma aplicação com alta carga de consultas à base de dados, como serviços web, por exemplo, o banco MongoDB é uma ótima alternativa. Entretanto, se a

aplicação necessita de uma camada de segurança mais robusta, com controle de acessos simultâneos à base de dados, o banco PostgreSQL é a melhor alternativa. [...]".

2.6.4 JavaScript Object Notation (JSON)

JSON (*JavaScript Object Notation*) é um formato para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML (*eXtensible Markup Language – Linguagem de Marcação Extensível*). Recomendada pela W3C para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais).

A ideia utilizada pelo JSON para representar informações é muito simples: para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado. Esta sintaxe é derivada da forma utilizada pelo JavaScript para representar informações.

Um objeto JSON pode representar, virtualmente, qualquer tipo de informação. O exemplo do quadro a seguir (QUADRO 06) mostra a representação dos dados de um filme:

```
{
  "titulo": "JSON x XML",
  "resumo": "o duelo de dois modelos de representação de informações",
  "ano": 2012,
  "genero": [
    "aventura",
    "ação",
    "ficção"
  ]
}
```

QUADRO 06: Exemplo de formato JSON

2.7 NAVEGADORES DE INTERNET

Um navegador (também conhecido como WEB browser ou simplesmente browser) é um programa que habilita seus usuários a interagirem com documentos HTML hospedados

em um servidor Web. É ele quem recebe as páginas de internet sejam em HTML ou PHP, e fazem as devidas interpretações do código para mostrar aos usuários.

Segue abaixo a linha do tempo dos navegadores mais famosos:

- 1993 – Mosaic – Foi o primeiro navegador a rodar no Windows, fator determinante para a abertura da web para o público em geral. Marc Andreessen, o líder do time que desenvolveu o Mosaic, saiu da NCSA e, com Jim Clark, um dos fundadores da Silicon Graphics, Inc. (SGI) e outros quatro estudantes formados e nomeados da Universidade de Illinois, iniciaram o *Mosaic Communications Corporation*. Mosaic Communications finalmente se tornou a Netscape Communications Corporation, produzindo o Netscape Navigator.
- 1994 – Netscape – O Netscape trouxe todas as características que um navegador moderno oferece nos dias de hoje, como por exemplo a navegação por abas, o bloqueio de *pop-ups*, suporte a *cookies* e histórico de visitas, entre outros. Reinou absoluto durante anos, mas já em 2002 seus usuários se resumiam a alguns poucos gatos pingados. Um dos motivos foi o fato da Microsoft passar a incluir, já em 1995, o Internet Explorer junto com o sistema operacional Windows. Era o início de um novo reinado no mundo dos *browsers*.
- 1995 – Microsoft Internet Explorer 1.0 – Em 1995, a Microsoft entra na briga dos navegadores com seu Internet Explorer 1.0, parte integrante do pacote “Plus” do Windows 95. Com isso, teve início a ‘guerra dos navegadores’. A guerra dos navegadores Web é o nome dado a um período de quatro anos (de 1995 a 1999) no qual a empresa Netscape, produtora do *software browser* homônimo, perde a sua liderança absoluta no mercado de navegadores para a Microsoft.
- Este período resultou em uma reversão total no uso de um software para outro, além de gerar projetos como o Mozilla e o Opera.
- 1998 – Mozilla – A Netscape anuncia a liberação do código-fonte de seu navegador. Com isso, o download do programa se torna grátis e sua programação, *Open-Source*, livre para ser usada e modificada por qualquer um. Para divulgar o código, a Netscape cria a comunidade Mozilla, que anos depois lançaria o Firefox.
- 2000 – Opera – O Opera é um navegador criado em 1994 pela empresa estatal de telecomunicações da Noruega e foi a primeira alternativa leve para os usuários. Recentemente perdeu seu posto de “navegador alternativo” para o Mozilla Firefox, mas conta ainda com uma fiel comunidade de usuários. Diversos dos recursos mais

modernos existentes entre os navegadores vieram do Opera e foram copiados para os demais. Em sua quinta versão, o navegador Opera tenta o modelo *adware*(gratuito para usuário, mas sustentado por anúncios embutidos no navegador).

- 2003 – Apple Safari – Até 2003, a plataforma Mac usava navegadores Netscape. Em 2003, a Apple anuncia seu próprio navegador, o Safari, incluído como o navegador padrão a partir do sistema operacional Mac OS X. Com uma interface simples, suas funções são básicas: Abas, bloqueador de *pop-ups*, gerenciador de *downloads* de arquivos, leitor de notícias RSS e modo privado que evita que terceiros monitorem sua navegação.
- 2004 – Mozilla Firefox – Em 2004 é lançado o Firefox 1.0, que surgiu como uma versão mais simplificada do Mozilla. Além de ser gratuito e de código aberto, o software ficou conhecido por sua navegação por abas, apesar de não ser pioneiro na funcionalidade – o navegador Ibrowse e o Opera disponibilizaram o recurso antes.
- 2008 – Chrome – Depois de muita especulação, o Google finalmente se lança no mercado de navegadores com o Chrome, um navegador “projeto do zero” e com a promessa de ser mais rápido, seguro e estável que os concorrentes. Entre seus pontos altos, a estrutura de processamento do programa, em que cada aba roda um processo em paralelo, o que, segundo o Google, pouparia recursos do sistema e preveniria vazamentos de memória e travamentos do computador.

Atualmente o navegador Chrome é o navegador mais utilizado seguido de Safari e Internet Explorer. O Ranking completo pode ser visualizado na IMAGEM 04.

Web Browser	Sep-13	Oct-13	Nov-13	Dec-13	Jan-14	Feb-14	Mar-14	Apr-14	Change from Sept'13-Apr'14
Chrome	34.68%	35.02%	35.04%	34.91%	34.80%	36.32%	35.49%	34.65%	-0.11% -0.04 pp
Safari	16.15%	15.18%	14.81%	15.79%	15.47%	15.49%	15.46%	15.19%	+5.98% +0.97 pp
Internet Explorer	15.62%	15.94%	15.35%	13.70%	13.87%	12.84%	10.54%	12.49%	-20.06% -3.13 pp
Firefox	16.60%	16.17%	15.15%	14.75%	13.71%	13.65%	12.58%	11.94%	-28.08% -4.66 pp
Safari (in-app)	4.86%	6.12%	7.57%	7.93%	9.11%	9.03%	11.99%	10.76%	121.15% 5.89 pp
Android Browser	7.39%	7.31%	7.86%	8.44%	8.50%	8.51%	9.71%	10.55%	42.71% 3.16 pp
Other	2.02%	1.79%	1.75%	1.90%	2.09%	1.94%	1.78%	1.84%	-9.05% -0.18 pp
Opera Mini	1.56%	1.40%	1.40%	1.49%	1.40%	1.19%	1.55%	1.75%	11.77% 0.18 pp
Opera	1.10%	1.07%	1.07%	1.00%	1.02%	1.02%	0.90%	0.85%	-22.94% -0.15 pp

IMAGEM 04: Ranking de uso dos navegadores realizado pela empresa Shareaholic.

2.8 HTML

HTML é a abreviação para a expressão inglesa *HyperText Markup Language*, que significa Linguagem de Marcação de Hipertexto. Como o nome já diz, é uma linguagem de marcação de texto, utilizada para produzir as famosas páginas WEB.

A Linguagem HTML vaticinou o início de um novo capítulo na criação de páginas em HTML para a Web com recurso a elementos de texto som e imagem. Com esta nova ferramenta, os documentos na Web são organizados com uso de comandos que são interpretados pelos *browsers*. Esta linguagem também define a estética dos documentos que são apresentados através de comandos de formatação. (MORGADO, 2010, p 3)

Tudo começou em meados dos anos 90 no *CERN (European Council for Nuclear Research)*, na Suíça, quando Tim Berners-Lee propôs um projeto baseado no conceito de hipertexto para que fosse possível o acesso comum e uma troca de informações sobre pesquisas entre cientistas de diferentes universidades. Depois de muita pesquisa e testes, surgia a Internet como a conhecemos hoje, e também o HTML como a linguagem a ser utilizadas no desenvolvimento das páginas de Internet.

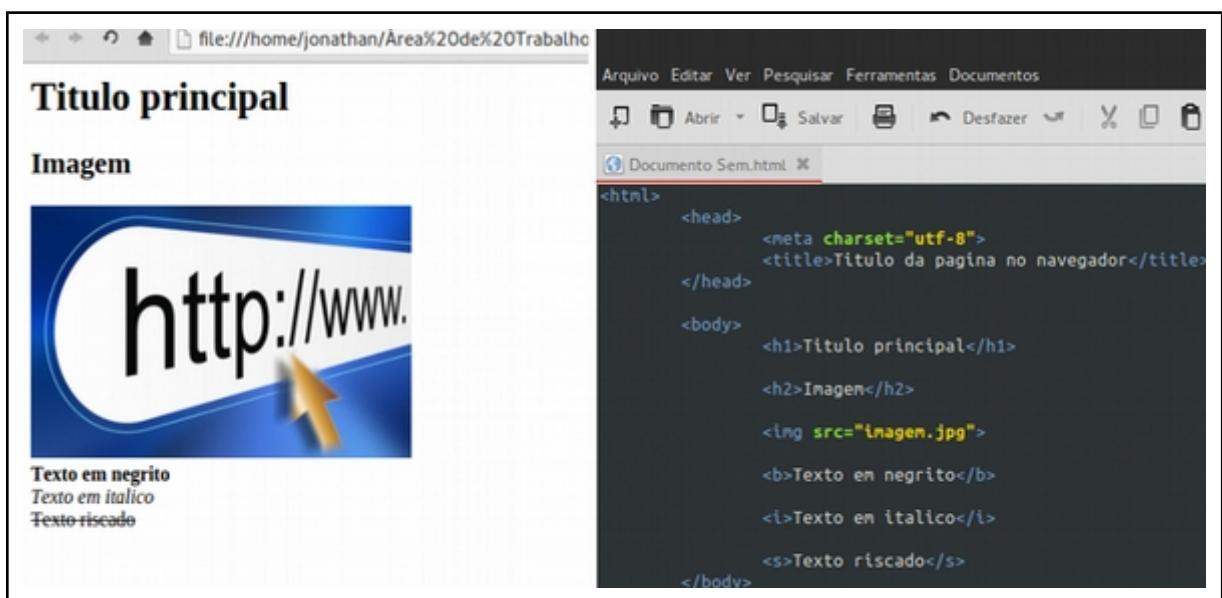


IMAGEM 05: Código HTML com visualizado no navegador

Basicamente essa linguagem funciona com *Tags* (marcadores) que definem como será cada trecho do texto por elas delimitadas. Por exemplo a *Tag* `<i>` seguida por seu fechamento de Tag `</i>`, define que o texto entre elas será exibido como Itálico. Essas *Tags* são lidas pelo

navegador de Internet que então exibe o texto conforme lhe é informado, como no exemplo da IMAGEM 05.

Ao passar dos anos essa linguagem foi se desenvolvendo e aderindo novas *Tags* e novas tecnologias:

- 1992 – Primeira aparição do HTML.
- 1993 – HTML+ Algumas definições da aparência, tabelas, formulários.
- 1994 – HTML v2.0 Padronização para as características principais.
- 1994 – HTML v3.0 Uma extensão do HTML+ entendido como um rascunho de padrão.
- 1994 – Criado o *World Wide Web Consortium(W3C)*: organização de padronização da *World Wide Web (www)*.
- 1995 – HTML v3.2 *Netscape* e *Microsoft Internet Explorer* definem seus próprios padrões baseados nas implementações correntes.
- 1995 – É introduzida o JavaScrip.
- 1996 – CSS1 é apresentada pela primeira vez a Folha de Estilo, criada para complementar a linguagem HTML. Possuía uma formatação simples e cerca de 60 propriedades.
- 1997 – HTML v4.0 São lançados os navegadores *Netscape v4.0* e *Internet Explorer v4.0* que apresentaram um conjunto de tecnologias que disponibilizaram diversos recursos tornando o HTML dinâmico. Surge então o DHTML.
- 1998 – CSS2 Em maio é lançado a segunda versão da Folha de Estilo que, além de incluir todas as propriedades do CSS1 ainda apresenta por volta de 70 novas propriedades.
- 1999 – HTML v4.01 Algumas modificações da versão anterior.
- 2000 – XHTML v1.0 É criado e consiste de uma versão XML do HTML v4.01.

2.9 HTML5

O HTML5 é a quinta versão para o padrão da linguagem HTML. Ele foi remodelado, incluindo novas *tags*, como `<nav>` usado para definir uma seção que contém apenas links de navegação, `<video>` e `<audio>` para adicionar essas mídias diretamente ao código.

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final. Ao contrário das versões anteriores, o HTML5 fornece ferramentas para a CSS e o JavaScript fazerem seu trabalho da melhor maneira possível. O HTML5 permite por meio de suas APIs a manipulação das características destes elementos, de forma que o *website* ou a aplicação continue leve e funcional, sem a necessidade de criações de grandes blocos de *scripts*. (EIS; FERREIRA, 2012, p 219)

O HTML5 também traz novos estilos com o CSS3 e novas funções JavaScript, para que as páginas ficassem mais ricas e que fosse possível exibir qualquer tipo de conteúdo como vídeos, músicas, gráficos e etc, sem a necessidades de instalar complementos adicionais aos navegadores, como Flash ou Java. Ele também permite a que o *design* das páginas sejam as mesmas em quaisquer plataformas, sendo elas computadores, *tablets* ou *smartphones*.

Possibilita ainda a criação de jogos apenas com o uso do HTML5, como o *Don't Starve*, vide IMAGEM 06. Exemplos das infinitas possibilidades, podem ser encontradas na página <http://www.chromeexperiments.com/>.



IMAGEM 06: Jogo *Don't Starve* desenvolvido em HTML5.

2.10 CSS

O CSS(*Cascading Style Sheets*, em português: camadas de estilo em cascata) é uma simples ferramenta embutida no HTML a partir de 1996 que define os estilos das páginas WEB, como fonte, espaçamentos, cores, e posicionamentos, entre outros.

As CSS têm por finalidade devolver à marcação HTML/XML o propósito inicial da linguagem. A HTML foi criada para ser uma linguagem exclusivamente de marcação e estruturação de conteúdos. Isso significa que, segundo seus idealizadores, não cabe à HTML fornecer informações ao agente do usuário sobre a apresentação dos elementos. Por exemplo: cores de fontes, tamanhos de textos, posicionamentos e todo o aspecto visual de um documento não devem ser funções da HTML. Cabem às CSS todas as funções de apresentação de um documento, e essa é sua finalidade maior. (SILVA, 2012, p 25)

Håkon Wium Lie e Bert Bos apresentaram a proposta do CSS que logo foi apoiada pela W3C. A ideia geral era, utilizar HTML somente para estruturar o site, e a tarefa de apresentação fica com o CSS disposto em um arquivo separado .css ou no próprio HTML demarcado pelas *tags* <style></style> como visto no QUADRO 07.

```
<style type="text/css">
#Layer1 {
    position:absolute;
    left:17px;
    top:27px;
    width:226px;
    height:121px;
    z-index:1;
}
```

QUADRO 07: Exemplo condigo CSS.

2.11 JAVASCRIPT

JavaScript foi originalmente desenvolvido por Brendan Eich da *Netscape* sob o nome de Mocha, posteriormente teve seu nome mudado para LiveScript e por fim JavaScript.

LiveScript foi o nome oficial da linguagem quando foi lançada pela primeira vez na versão beta do navegador Netscape 2.0 em setembro de 1995, mas teve seu nome mudado em um anúncio conjunto com a *Sun Microsystems* em dezembro de 1995.

A mudança de nome de LiveScript para JavaScript coincidiu com a época em que a *Netscape* adicionou suporte à tecnologia Java em seu navegador. A escolha final do nome causou confusão dando a impressão de que a linguagem foi baseada em java, sendo dito por muitos que foi uma estratégia de marketing da empresa para aproveitar a popularidade do recém-lançado Java.

JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que *scripts* pudessem ser executados do

lado do cliente e interagissem com o usuário sem a necessidade deste *script* passar pelo servidor. Permitindo que fosse utilizado lógica de programação para controlar os conteúdos, elementos e a comunicação das páginas.

A linguagem de marcação HTML destina-se a estruturar uma página web, não se devendo empregá-la para adicionar estilos ou apresentação visual aos elementos que constituem a página, sendo tais tarefas funções das folhas de estilo em cascata. A HTML, em sua versão atual – HTML 4.01, também não possui funcionalidades que permitam adicionar interatividade avançada à página, sendo tal tarefa função das linguagens de programação. (SILVA, 2010, p 36)

Seu código é definido dentro das *tags* HTML `<script></script>`. Possui sintaxe semelhante a linguagens de programação tradicionais como c, c++. Um exemplo de sua utilização pode ser visto no QUADRO 08.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8" />
    <title>Wikipédia</title>
    <script>
        window.onload = function () {
            document.getElementById("hello").addEventListener("click", function() {
                alert("Bem-vindo à Wikipédia!");
            }, false);
        };
    </script>
</head>
<body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <button id="hello">Dizer "Olá"</button>
</body>
</html>
```

QUADRO 08: Exemplo código JavaScript dentro de um arquivo HTML.

2.12 JQUERY

JQuery é um *framework* de JavaScript de código aberto, mantido pela comunidade. Um *Framework* é uma coleção de funções e métodos prontos para serem utilizados, amplamente testados e que devem ser usados de forma pré-definida para que o resultado esperado seja alcançado.

Em alguns casos um *framework* chega a ser um estilo completamente novo de programar em certa linguagem, no caso do Javascript, o jQuery é, sem dúvida, um estilo novo, atrativo, fácil e interessantíssimo de programar.

O jQuery foi criado sob o mantra do “Escreva menos, faça mais”, e é exatamente por causa disso que ele é tão surpreendente, com algumas poucas linhas de código você consegue fazer as mesmas coisas que antes custavam dezenas de linhas de código com JavaScript puro, como visto na no QUADRO 09.

```
jQuery
$(‘body’).css(‘background’, ‘#ccc’);
```

```
JavaScript
Function changeBackground(color) {
    Document.body.style.background = color;
}
Onload="changeBackground ('red');"
```

QUADRO 09: Exemplo de código em jQuery e JavaScript.

2.13 AJAX

Basicamente AJAX é carregar e renderizar uma página, utilizando recursos de scripts rodando pelo lado cliente, buscando e carregando dados no plano de fundo sem a necessidade de recarregar a página.

AJAX é acrônimo para “*Asynchronous JavaScript And XML*” e foi gerado por Jesse James Garret, em um artigo no site da sua empresa *Adaptive Path*, em fevereiro de 2005.

Ajax não é uma tecnologia, mas sim um conjunto de tecnologias. O conceito de AJAX se resume em conhecer bem JavaScript, trabalhando com DOM(*Document Object Model*), CSS e XML.

No carregamento da página, toda a lógica de processamento de dados é passado ao cliente. Quando o usuário faz uma requisição, quem busca e traz essas informações é o JavaScript, de forma assíncrona, não causando assim o chamado “recarregamento” na tela. O tratamento dos dados, seu formato e exibição ficam todos por conta do *script* que foi carregado inicialmente quando se acessou a página. O processo inicial de carregamento é

mais lento que de uma aplicação comum, pois muitas informações são pré-carregadas. Mas depois, somente os dados são carregados, tornando assim o site mais rápido.

2.14 PHP

Segundo o Manual do PHP (2014), PHP (um acrônimo recursivo para PHP: *Hypertext Preprocessor*) é uma linguagem de *script* código aberto e de uso geral, muito utilizada para o desenvolvimento de aplicações WEB embutidas dentro do HTML.

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial. (Barreto, 2000, p. 6)

O PHP sucede de um produto mais antigo, chamado PHP/FI. PHP/FI foi criado por Rasmus Lerdorf em 1995, inicialmente como simples *scripts* para estatísticas de acesso para seu currículo online. Ele nomeou esta série de *script* de '*Personal Home Page Tools*'. Como mais funcionalidades foram aparecendo, Rasmus escreveu uma implementação C muito maior, e que era capaz de comunicar-se com base de dados possibilitando a usuários desenvolver simples aplicativos dinâmicos para Web. Ele resolveu então disponibilizar o código fonte do PHP/FI para que todos pudessem ver e também usar, bem como corrigir problemas e melhorar o código.

O PHP teve várias versões lançadas, mas as melhorias mais significativas surgiram em 1997, quando Andi Gutmans e Zeev Suraski o reescreveram a partir do zero, e o novo código foi batizado de PHP versão 3. Uma grande quantidade de código do PHP/FI foi aproveitado para o PHP 3, sendo outra parte completamente reescrita.

Uma das maiores virtudes do PHP 3 era sua funcionalidade de extensão, sendo este o motivo que atraiu inúmeros desenvolvedores a aderirem a linguagem, criando e submetendo novos módulos à comunidade, ajudando muito para seu sucesso. Criaram então o mecanismo Zend e com alguns ajustes foi lançada a versão 4 do PHP, essa já tinha suporte a orientação a objetos, mas não como seu foco principal.

Depois de uma longa pesquisa e desenvolvimento, contando com várias versões de pré-lançamento, finalmente o PHP 5 foi lançado em julho de 2004, baseado no Zend Engine 2.0 e contando com um novo modelo de objeto, além de dezenas de outras novas funcionalidades.

2.15 BOOTSTRAP

Em 2011, o Bootstrap foi criado como uma solução interna para resolver inconsistências de desenvolvimento dentro da equipe de engenharia da empresa Twitter. Basicamente, não havia uma definição de estrutura de código na forma escolhida pelos engenheiros do Twitter para desenvolver a plataforma.

O Bootstrap foi uma ferramenta desenvolvida originalmente por Mark Otto e Jacob Thornton, então engenheiros do Twitter, como uma tentativa de incentivar o uso de uma única estrutura pela equipe de engenharia do Twitter, reduzindo essas inconsistências. A iniciativa certamente foi bem-sucedida no Twitter, permitindo que toda a equipe trabalhasse com maior rapidez e eficiência e menos inconsistências.

Embora inicialmente uma solução interna no Twitter, Mark e Jacob rapidamente perceberam que tinham descoberto algum muito maior. Em agosto de 2011, a estrutura Bootstrap foi lançada como um projeto de software livre.

O Bootstrap é um *framework* composto de uma coleção de vários elementos e funções personalizáveis para projetos da web, empacotados previamente em uma única ferramenta. Ao projetar um site com o Bootstrap, os desenvolvedores podem escolher quais elementos querem usar. E, o mais importante, podem ter a certeza de que os elementos escolhidos não conflitarão entre si. É como um quebra-cabeças, exceto que cada peça se encaixa perfeitamente com as outras, quaisquer que sejam as peças escolhidas.

2.16 GOOGLE MAPS API

Dentre os vários produtos da Google está o Google Maps, que permite aos usuários visualizarem o globo terrestre por mapas de satélite ou por mapas das ruas, visualizar locais

em 3D, navegação com GPS entre outros. Permite ainda visualizar clima, encontrar pontos importantes como estabelecimentos comerciais, hospitais e afins como visto na IMAGEM 07, criar rotas entre pontos no mapa entre outras funcionalidades.

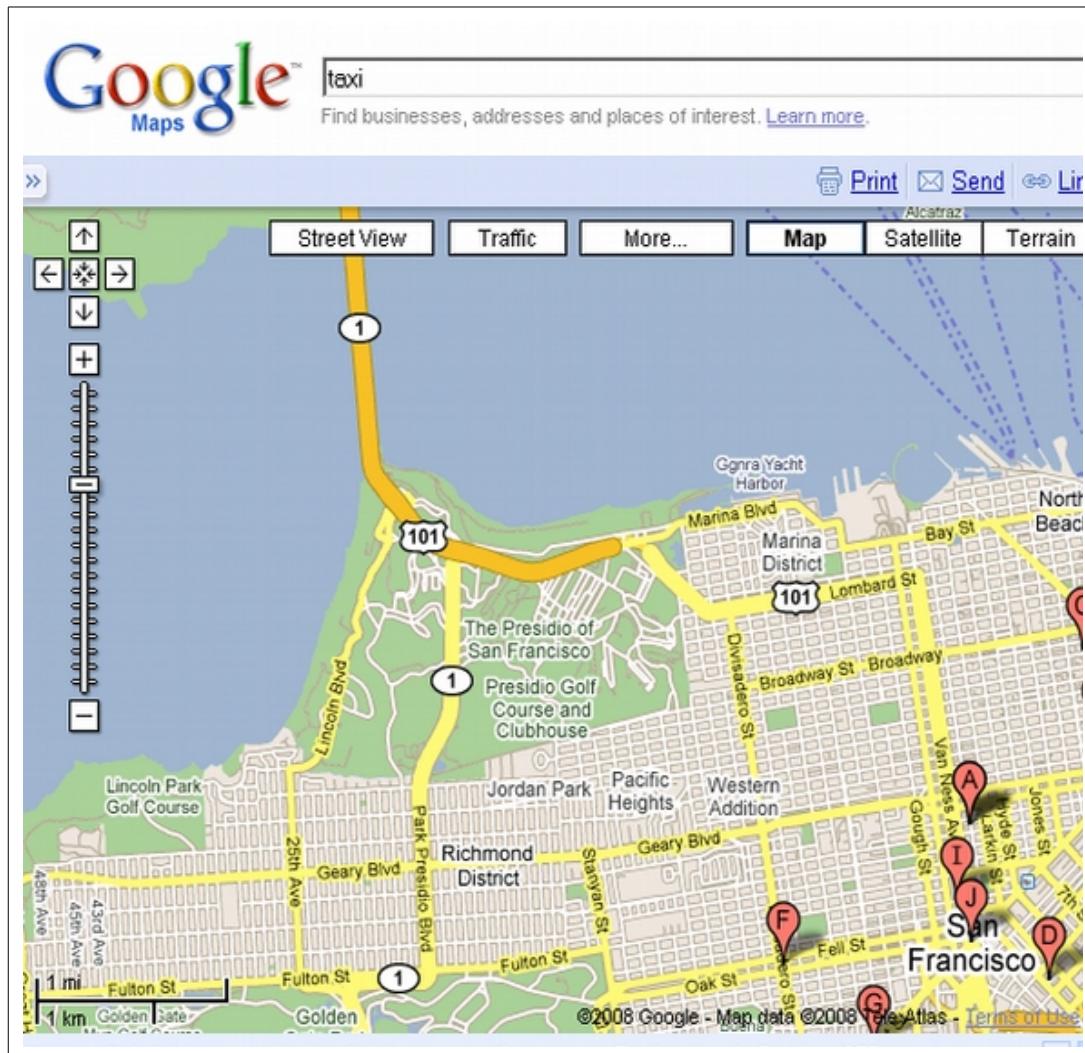


IMAGEM 07: Busca de táxi em determinada região do globo.

A primeira prévia desse produto surgiu em 2004 com o Google Local, que oferecia listagens, mapas e orientação para empresas locais. Mais tarde, em 2005, o Google Local foi combinado com o Google Maps, que foi primeiramente lançado na Europa. Com o passar dos anos foi expandindo os mapas, e hoje possui um dos maiores, se não o maior banco de mapas da internet.

O Google Maps ainda proporciona um API (Interface de programação de aplicações) para que desenvolvedores possam, por exemplo, incorporar os mapas em quaisquer WEB sites, criar aplicativos baseados em localização, criar mapas para aplicativos móveis, entre outros.

Outros serviços a serem destacados da API do Google Maps são: Google Distance Matrix API, que calcula as distâncias entre determinadas coordenadas geográfica; Google Directions API, que cria rotas entre coordenadas geográficas; Google Elevation API, que determina a elevação de uma coordenada geográfica baseado no nível do mar; e Google Geocoding API, que converte coordenadas geográficas em endereços, ou endereços em coordenadas.

2.17 GOOGLE CHARTS API

O Google Charts API é uma ferramenta muito útil e interessante para quem desenvolve aplicações web e quer gerar gráficos de uma forma dinâmica. Esta API caracteriza-se pela facilidade de utilização e implementação, não sendo necessária a instalação de qualquer software ou *frameworks*. Para a sua utilização, basta o URL da API no qual serão referenciados os dados e características necessários para gerar o gráfico pretendido.

Ela é vista por muitos como a melhor disponível hoje, por ser muito clara, ter uma curva de aprendizagem pequena e uma documentação muito boa.

Com ela podemos desenhar os principais tipos de gráficos: torta (pizza), barras, linhas, tabelas entre outros, como visto na IMAGEM 08.

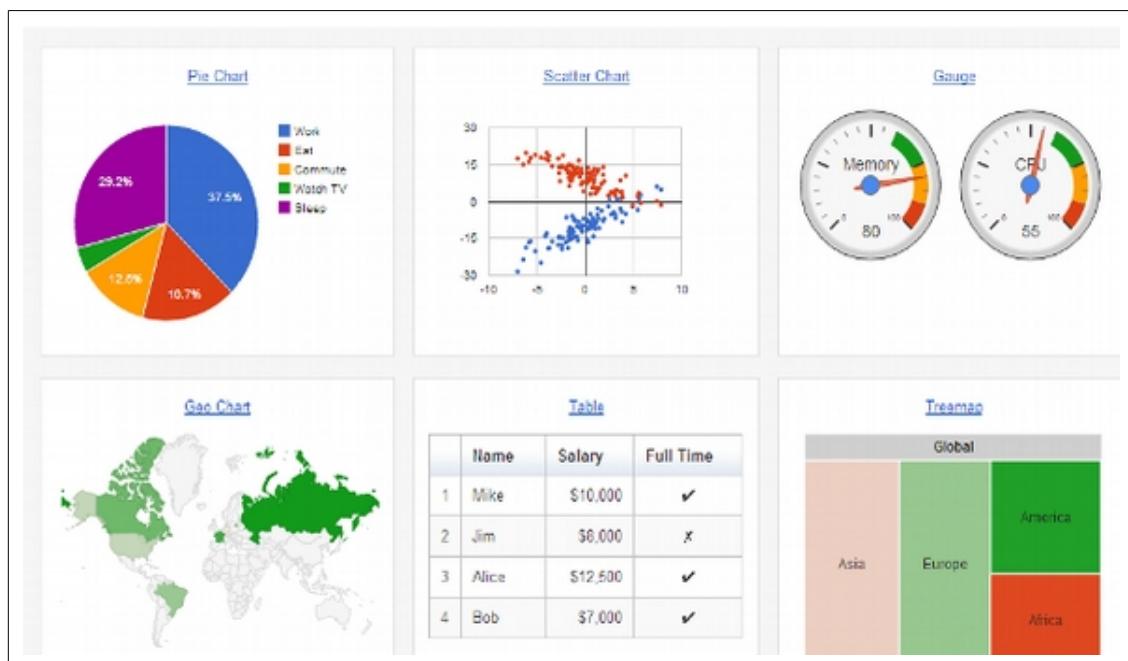


IMAGEM 08: Exemplos de gráficos criados com a Google Charts API

2.18 FACEBOOK DEVELOPERS

A empresa Facebook Inc. fornece a qualquer desenvolvedor uma API de integração e criação de aplicativos ou jogos para sua rede social Facebook.

Essa API permite adicionar funções como o compartilhamento de conteúdos, fotos, conectar com a conta Facebook, ou adicionar o botão Curtir em páginas ou aplicativos de terceiros, através de códigos e documentações fornecidas pela empresa. Para os aplicativos criados para a rede social, fornece o uso de propagandas, comércio eletrônico, entre outros.

2.19 JAVA

Java é uma linguagem de programação orientada a objetos que começou a ser criada em 1991, na Sun Microsystems. Teve início com o Green Project, no qual os mentores foram Patrick Naughton, Mike Sheridan, e James Gosling. Este projeto não tinha intenção de criar uma linguagem de programação, mas sim de antecipar a “próxima onda” que aconteceria na área da informática e programação. Os idealizadores do projeto acreditavam que em pouco tempo os aparelhos domésticos e os computadores teriam uma ligação.

A tecnologia Java teve uma enorme utilização, e logo grandes empresas como a IBM, anunciaram que dariam suporte ao Java, ou seja, os produtos destas empresas rodariam aplicativos feitos em Java. Estimativas apontam que a tecnologia Java foi a mais rapidamente incorporada na história da informática. Em 2003 o Java já tinha mais de 4 milhões de desenvolvedores. A ideia inicial do Green Project começou a se concretizar. A linguagem deles passou a ser utilizada em dezenas de produtos diferentes. Computadores, celulares, *palmtops*, e a maioria dos produtos da Apple.

Java é uma tecnologia, pois envolve várias técnicas e ferramentas para o desenvolvimento de aplicações. O “tecnologia”, na verdade, seria uma definição mais ampla do que é o Java.

Java é uma linguagem de programação, pois possui sintaxe e semântica próprios para a criação de programas. A linguagem de programação, no caso de Java, é um dos componentes que compõem a tecnologia Java.

Java é uma plataforma de desenvolvimento, pois oferece um ambiente computacional que disponibiliza os recursos necessários para a criação e execução dos programas escritos com a linguagem java. (COSTA, 2008, p. 8)

Em 2009 a Oracle realizou a compra da Sun Microsystems por US\$7,4 bilhões, tornando-se a responsável pelo desenvolvimento da linguagem.

2.19.1 Características

- Suporte à orientação a objetos;
- Portabilidade;
- Segurança;
- Linguagem Simples;
- Alta Performance;
- Dinamismo;
- Interpretada (o compilador pode executar os *bytecodes* (forma intermediária de código compilado interpretada pelas Máquinas Virtuais Java - JVMs) do Java diretamente em qualquer máquina);
- Distribuído;
- Independente de plataforma;
- Tipada (detecta os tipos de variáveis quando declaradas);

2.19.2 Java Virtual Machine (JVM)

Dentro das características do Java, o principal item é o fator da “Independência de plataforma”. Hoje a maioria das linguagens sofrem na transferência de plataforma quando o sistema desenvolvido tem que migrar para outra plataforma, pois o compilador transforma o arquivo-fonte em código de máquina específico para a arquitetura do processador e sistema operacional.

Por exemplo, se o programa desenvolvido for compilado em sistemas Macintosh, mais tarde terão problemas quando forem migrar para plataformas Intel, tendo que transferir o código fonte para a plataforma Intel e fazer a compilação novamente para produzir o código de máquina específico para este sistema. Muitas vezes o programador terá que alterar o código fonte antes de efetuar a compilação para a nova plataforma, esse motivo acontece por possuírem arquiteturas de processador diferenciadas (IMAGEM 09).

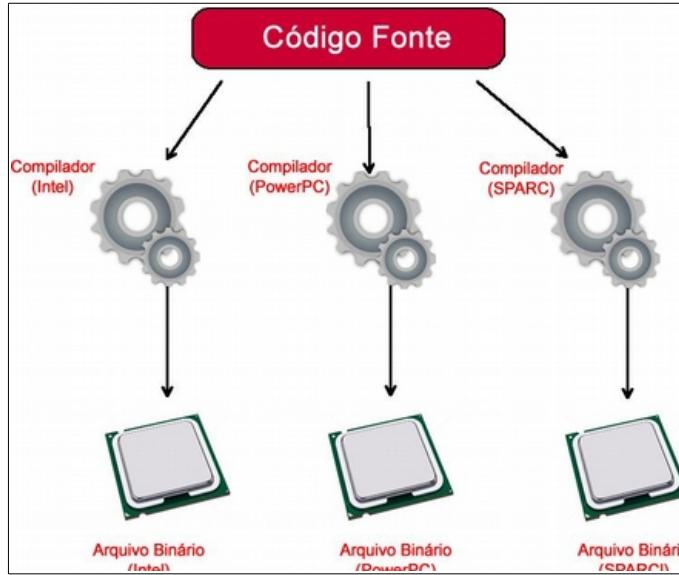


IMAGEM 09: Compilador padrão

Já os programas em Java possuem uma característica fundamental que permite desenvolver sem se preocupar com o tipo de sistema ou plataforma que precisa ser desenvolvida e preparada.

A independência de plataforma já fala por si, pois possibilita o programa ser executado em diferentes plataformas e sistemas operacionais, através de um emulador conhecido como a Máquina Virtual Java ou JVM (Java Virtual Machine) que ajuda rodar os sistemas baseados em Java. Pode-se também se denominar como uma máquina virtual baseada em software que é executada dentro dos aparelhos eletrônicos onde irá ler e executar os *bytecodes* do Java (IMAGEM 10).

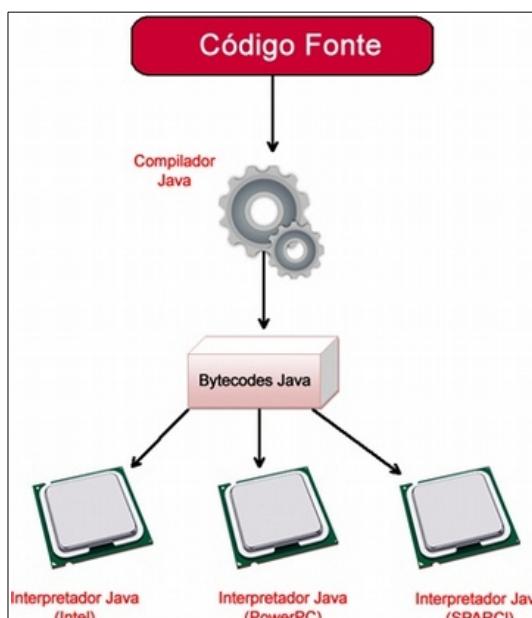


IMAGEM 10: Compilador Java

2.20 LAMP

LAMP é uma combinação de softwares livres e de código aberto. O acrônimo LAMP refere-se as primeiras letras de:

1. Linux (sistema operacional),
2. Apache (servidor web),
3. MariaDB ou MySQL (software de banco de dados) e
4. PHP, Perl ou Python (linguagens de programação).

A combinação exata dos softwares incluídos em um pacote LAMP pode variar, especialmente com respeito ao software de *script* web, uma vez que Perl ou Python as vezes são tirados do conjunto. Apesar de os autores originais desses programas não terem desenvolvido-os para trabalhar especificamente um com o outro, a filosofia e o conjunto de ferramentas de desenvolvimento são compartilhados e foram desenvolvidos em conjunção próxima. Essa combinação de software tornou-se popular principalmente por possuírem código aberto, livres de custo, e assim de fácil adaptação. Quando usadas juntas, suportam servidores de aplicações web.

O termo LAMP sofre variações de acordo com o sistema operacional alvo, porém mantém suas funcionalidades e normalmente seus softwares.

GNU/Linux – **LAMP**

Microsoft Windows – **WAMP**

Apple MAC OS – **MAMP**

2.21 XAMPP

XAMPP é um servidor com as mesmas funções do LAMP, porém independente de plataforma. Consiste principalmente na base de dados MySQL, o servidor web Apache e os interpretadores para linguagens de *script* PHP e Perl. O programa está liberado sob a licença GNU e atua como um servidor web livre, fácil de usar e capaz de interpretar páginas dinâmicas. Atualmente XAMPP está disponível para Microsoft Windows, GNU/Linux, Solaris, e MacOS X.

2.22 DAEMON (SERVIÇOS)

Em sistemas operacionais (S.O.) multitarefas, como Unix ou Linux, um Daemon (*Disk And Execution Monitor* – Monitor de Execução e de Disco) também conhecido por “serviço”, no Microsoft Windows, é um programa de computador que roda de forma independente no plano de fundo (*background*), em vez de ser controlado diretamente por um usuário. Tipicamente, daemons têm nomes que terminam com a letra “d”; por exemplo, *syslogd*: daemon que gerencia o *log* do sistema (“*system log*”).

Os dameons são os programas que ficam em execução em segundo plano para servir outros programas, em outras palavras é um software que não estabelece relação direta com o usuário, ele cria uma interface com as aplicações que interagem. O daemon inicia, reinicia ou para a execução de um serviço provido pelo sistema e também é responsável, em muitos casos, de recarregar as configurações do programa. O daemon do Linux é muito semelhante aos serviços do Microsoft Windows XP e Microsoft Windows 2003. (BRITO, 2008, p. 19)

De forma geral, os sistemas operacionais iniciam daemons durante o processo de *boot* (termo em inglês para o processo de iniciação do computador que carrega o sistema operacional), e o processo pai (primeiro processo) normalmente é o *init* (PID 1). Os daemons então lidam com requisições diversas de serviços, como requisições de rede, atividades de *hardware*, dentre outros. Também podem executar tarefas em horários pré-determinados ou realizar configurações de *hardware*.

2.22.1 Origem do Termo

A palavra daemon é uma forma alternativa de escrever *demon* (demônio em inglês). O termo foi trazido para a computação pelos programadores do projeto MAC do MIT. A ideia partiu do demônio de Maxwell, um ser imaginário de um experimento teórico muito famoso, no qual um demônio trabalharia constantemente em segundo plano, selecionando as partículas de uma caixa.

De qualquer forma, os sistemas Unix herdaram essa nomenclatura e, conforme eles se espalharam, o termo é disseminado por toda parte.

Alguns nomes alternativos para *daemons* são serviços (Windows), subsistema (IBM z/OS), servidor virtual (IBM VM) e tarefa fantasma (XDS UTS).

2.22.2 Java Service Wrapper

Java Service Wrapper (Tanuki Software, 2014) é uma biblioteca que disponibiliza um conjunto de arquivos binários e *scripts* para diferentes arquiteturas e sistemas operacionais para executar aplicações Java como *daemon*.

Existem alternativas para o propósito apresentado, como a biblioteca JSVC (<http://commons.apache.org/proper/commons-daemon/jsvc.html>), porém o Java Service Wrapper é multiplataforma e possui melhor curva de aprendizado.

A biblioteca Java Service Wrapper é distribuída sob três licenças:

1. Licença para servidores: Esta opção provê a utilização da ferramenta em apenas um servidor ou estação de trabalho.
2. Licença para desenvolvedores: Esta opção permite o empacotamento e redistribuição da biblioteca com um ou mais sistemas comerciais.
3. Licença comunitária: Esta opção é baseada na popular licença de software livre GPL v2 (*General Public License* versão 2). Para projetos de código aberto esta é a melhor escolha, pois permite a utilização e distribuição da biblioteca sem custos.

2.23 SOCKET

Os computadores na Internet são conectados entre si pelo protocolo TCP/IP. Na década de 1980, a ARPA (*Advanced Research Projects Agency*) do governo norte-americano forneceu recursos financeiros à Universidade da Califórnia em Berkeley com a finalidade de oferecer uma implementação UNIX do pacote de protocolos TCP/IP. O que foi desenvolvido então ficou conhecido como interface de *sockets*.

Um *socket* essencialmente é uma conexão de dados transparentes entre dois computadores.

Um *socket* não é nada além de uma abstração conveniente. Ele representa um ponto de conexão para uma rede TCP/IP. Quando dois computadores querem manter uma conversação, cada um deles utiliza um *socket*. Um computador é chamado servidor, ele abre um *socket* e presta atenção às conexões. O outro computador denomina-se cliente, ele chama o *socket* servidor para iniciar a conexão. Para estabelecer uma conexão, é necessário apenas um endereço de destino e um número de porta. (HOPSON, 1997, p. 335)

Os *sockets* têm dois modos principais de operação: o modo baseado em conexões (TCP) e o modo sem conexão (UDP). Nos baseados em conexões, tudo o que for enviado chega na mesma ordem em que foi transmitido. Nos sem conexão os diferentes itens da correspondência podem chegar em uma ordem diferente daquela em que foram enviados.

2.24 WEB SERVICES

Os *Web Services* surgiram com a grande demanda de utilização de Internet e a necessidade da criação de aplicações e disponibilizá-las em grande escala. Inicialmente as páginas da internet eram estáticas, mas com o passar dos anos, devido a evolução tecnológica, as mesmas começaram a interagir de forma a acionar programas produtores de informação dinâmica, proveniente de bancos de dados ou outras fontes.

Nessa fase de evolução, iniciou-se o modelo de aplicações distribuídas para Internet. Esse fato proporcionou o surgimento de várias soluções para a criação de aplicações que são capazes de extrair dados de várias fontes e disponibilizá-las aos usuários. Criou-se então, uma demanda de serviços que poderiam ser realizados na WEB: A necessidade das empresas de todos os tamanhos de trocar informações, entre sistemas diferentes que pudessem se comunicar através de padrões simples e públicos, sendo assim suprida com a utilização de *Web Services* e a linguagem XML, que pode ser considerada onipresente, simples e genérica.

Do ponto de vista técnico, Web Services constituem-se em software de baixo acoplamento, reutilizáveis, com componentes feitos para serem facilmente acessados pela internet. Na ótica conceitual, o emprego de Web Services representa um modo para integrar tarefas que compõem um processo de negócios através da internet, em uma cadeia de valor na qual procedimentos estão interligados e são interdependentes para atingir um resultado concreto final. (ABINADER; LINS, 2006, p. 10).

2.25 AXIS 2

Com o aumento da busca por soluções em *Web Services* pelas empresas, novas necessidades foram encontradas e novos padrões estão sendo definidos. As empresas procuram confiabilidade, segurança e desempenho. Até então o Axis 1 supria algumas dessas

necessidades, mas com o aumento de demanda nesse tipo de serviço, ele começou a se tornar ultrapassado, apesar de seu motor de *Web Service* ser estável. Então a *Apache Foundation*, empresa que o mantém, decidiu que a melhor opção era não atualizá-lo e formular uma nova arquitetura, lançando outra ferramenta, então surgiu o Axis 2.

No Axis2, o motor principal é um mecanismo de puro processamento SOAP (Protocolo Simples de Acesso a Objetos). Cada mensagem que entra no sistema, deve ser transformada em uma mensagem SOAP, antes de ser entregue ao núcleo do motor. Uma mensagem de entrada pode ser uma mensagem SOAP ou não-SOAP(REST JSON ou JMX). Mas ao nível de transportes, ele será convertido em uma mensagem SOAP.

O Apache Axis2 foi lançado em 2006 sendo considerado a terceira geração de frameworks para desenvolvimento de *Web Services* da Apache. Seus antecessores são Apache SOAP e Apache Axis 1.0. O Axis2 foi criado para atender os novos padrões de *Web Services*, pois não era viável alterar a arquitetura do Axis 1.0. (JÚNIOR, 2013, p. 23)

O Axis 2 gera um WSDL (*Web Service Description Language*), que funciona como um contrato entre as duas ou mais partes. Ele é um documento escrito em XML, que além de descrever o serviço, especifica como acessá-lo e quais as operações e métodos disponíveis. É utilizado em combinação com SOAP e XML para fornecer *Web Services*. O solicitante do serviço que se conecta ao *Web Service*, pode ler esse documento para determinar quais as funções disponíveis para utilização.

2.25.1 Simple Object Access Protocol (SOAP)

O SOAP (*Simple Object Access Protocol*) fornece um mecanismo simples e leve para troca de informações estruturadas de forma descentralizada em um ambiente distribuído utilizando o formato XML para o envio de suas mensagens. Um exemplo pode ser visto no QUADRO 10.

O protocolo SOAP pode formar a camada base de uma pilha de protocolos de *Web Services*, fornecendo um *framework* de mensagens básico sob o qual os serviços web podem ser construídos. Esse protocolo baseado em XML consiste de três partes: um envelope, que define o que está na mensagem e como processá-la, um conjunto de regras codificadas para expressar instâncias do tipos de dados definidos na aplicação e uma convenção para representar chamadas de procedimentos e respostas.

```

POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:GetLastTradePrice xmlns:m="Some-URI">
<symbol>DIS</symbol>
</m:GetLastTradePrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

QUADRO 10: Exemplo de Mensagem SOAP

2.26 APACHE TOMCAT

O Apache Tomcat é um servidor de aplicações Java para WEB que implementa as tecnologias *JavaServlets* e *JavaServer Pages*. Foi desenvolvido pela Apache Software Foundation e é distribuído como software livre. Foi inicialmente criado como um subprojeto do Apache Jakarta, mas devido a sua alta popularidade, acabou sendo separado do projeto inicial e passou a ser mantido por voluntários da comunidade de código aberto do Java. Ele permite a execução de aplicações para WEB.

Os *Servlets* são classes Java, desenvolvidas de acordo com uma estrutura bem definida (como mostrado no QUADRO 11) que quando instaladas e configuradas em um servidor que implemente um *Servlet Container*, podem tratar requisições recebidas de clientes WEB. Ao receber a requisição, o *Servlet* captura os parâmetros dessa requisição, executa o processamento necessário e retorna uma página HTML.

```

public class RemoteIPServlet extends HttpServlet {
    public void doGet( HttpServletRequest p_request, HttpServletResponse p_response )
        throws IOException {
        PrintWriter l_pw = p_response.getWriter();
        l_pw.println("<HTML> <BODY>");
        l_pw.println("O seu endereço IP é \\" + "
            p_req.getHeader("getRemoteAddr()") + "\\"");
        l_pw.println("< /BODY> < / HTML >");
        l_pw.flush ();
    }
}

```

QUADRO 11: Exemplo de Servlet

As *Java Server Pages*, foram criadas para melhorar o desenvolvimento com *Servlets*. Se em uma aplicação *Servlet*, a página HTML resultante se mistura com a lógica da aplicação, que dificulta a alteração dessa formatação. Já em uma página JSP(*Java Server Pages*), a formatação fica separada da programação, podendo ser modificada sem afetar o restante da aplicação. Assim, um JSP consiste de uma página HTML com alguns elementos especiais, que conferem o caráter dinâmico da página. A QUADRO 12 a seguir exibe um exemplo de página JSP.

```
<!--Página JSP Simples que imprime endereço IP da máquina que está fazendo o
acesso a esta página -->
<HTML>
    <BODY>
        O seu endereço IP é "<%= request.getRemoteAddr() %>"
    </BODY>
</HTML>
```

QUADRO 12: Exemplo de Pagina JSP

2.27 APACHE WEB SERVER

Para podermos acessar qualquer site, é necessário que haja um servidor por trás daquele endereço responsável por disponibilizar as páginas e os demais recursos que podemos acessar. Assim, quando enviamos um e-mail através de um formulário, colocamos uma mensagem em um fórum de discussão, realizamos uma compra on-line e afins, um servidor WEB (ou um conjunto de servidores) é responsável por processar todas essas informações.

Como dito em Marcelo (2005), Falar do servidor WEB Apache é a mesma coisa que falar sobre um dos softwares mais utilizados nos dias de hoje na Internet. O Apache é sem sombra de dúvidas, um dos mais robustos e seguros programas desenvolvidos para ambiente TCP/IP e que mantém em operação mais de 60% da *homepage/sites* disponíveis no mundo.

O servidor Apache (ou *Apache Server*) surgiu no NCSA (*National Center of Supercomputing Applications*) através do trabalho de Rob McCool.

O servidor Apache é capaz de executar códigos em PHP, Perl, *Shell Script* e até em ASP e pode atuar como servidor FTP, HTTP, entre outros. Sua utilização mais conhecida é a que combina o Apache com a linguagem PHP e o banco de dados MySQL.

2.28 ARDUINO

O Arduino é uma plataforma eletrônica para prototipagem baseada em *hardware* e *software* livre. Foi criada em meados de 2005 em Ivrea na Italia, por Mássimo Banzi, no *Interaction Design Institute Ivrea* (Instituto Ivrea de Design de Interação) com o intuito de criar uma plataforma de prototipagem eletrônica mais moderna do que as disponíveis no mercado, mais barata, mais simples e mais fácil de usar.

O Arduino possui um microcontrolador ATMega, que é um computador em um chip que contem microprocessador, memória RAM e dispositivos de entrada e saída. Esse microcontrolador possibilita que Arduino possa captar e enviar dados, para qualquer que seja sua finalidade. Seu principal objetivo é facilitar a automatização de projetos, sem que seja necessário um conhecimento profundo em eletrônica. Os projetos que podem ser desenvolvidos com essa tecnologia são os mais variados: é possível construir um quadricóptero, uma estação meteorológica, realizar automação residencial, entre muitas outras possibilidades que o desenvolvedor imaginar.

Uma vez que o Arduino é baseado em um microcontrolador e, portanto, é programável, torna-se possível criar diversas aplicações diferentes com uma certa facilidade. Além disso, o próprio equipamento pode ser reutilizado, através de uma nova programação. Por sua vez, a sua programação é simplificada pela existência de diversas funções que controlam o dispositivo, com uma sintaxe similar à de linguagens de programação comumente utilizadas (C e C++). Assim sendo, em um ambiente profissional, as características do Arduino fazem dele uma boa ferramenta de prototipação rápida e de projeto simplificado. Por outro lado, em um ambiente acadêmico, ele pode ser perfeitamente utilizado como ferramenta educacional, uma vez que não requer do usuário conhecimentos profundos de eletrônica digital nem da programação de dispositivos digitais específicos. (RENNÁ et al., 2013, p. 4)

2.28.1 Arduino Uno

O Arduino Uno (IMAGEM 11) é uma placa microcontroladora baseada no microcontrolador ATmega328. Ele possui 14 pinos de entrada/saída (onde 6 deles podem ser usados como saídas de energia), 6 entradas analógicas, um resonador de cerâmica de 16MHz, uma conexão USB, uma entrada de energia, cabeçalho ICSP e um botão de reiniciar. Possui tudo o que é necessário para suportar o microcontrolador, e suas funções são estendidas com a conexão de dispositivos de entrada e saída de daods dos mais diversos tipos.



IMAGEM 11: Arduino Uno

2.28.1.1 Especificações

Microcontrolador	ATmega328
Voltagem	5 V
Entrada de Voltagem (recomendado)	7-12V
Entrada de Voltagem (limites)	6-20V
Pinos de Entrada/Saída Digitais	14(onde 6 proveem saída de energia)
Pinos de Entrada Analógica	6
Corrente Continua por Pino de Entrada/Saída	40 mA
Corrente Continua por Pino de 3.3V	50 mA
Memória Flash	32 KB onde 0,5 KB é utilizado pelo bootloader
SRAM	1 KB
EEPROM	1 KB
Clock	16 MHz

QUADRO 13: Especificações do Arduino UNO

O suprimento de energia pode ser realizado via conexão USB ou por meio de alimentação externa, onde o método para suprir a energia é selecionado automaticamente. Todas as especificações podem ser visualizadas no QUADRO 13.

Cada um dos 14 pinos digitais podem ser usado como entrada ou saída, usando as funções *pinMode()*, *digitalWrite()*, e *digitalRead()*. Eles operam em 5 V. Cada pino pode receber no máximo 40 mA e existem alguns pinos com funções especiais:

- Serial: 0 (RX) e 1(TX). Usado para receber (RX) e transmitir(TX) dados seriais TTL.
- PWM: 3, 2, 6, 9, 10 e 11. Fornecem uma saída analógica PWM de 8 bits com a função *analogWrite()*.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Esses pinos suportam comunicação SPI.
- LED: 13. Existe um LED já conectado e montado no pino 13.

O Uno possui 6 entradas analógicas, marcadas de A0 a A5, cada uma delas com 10 bits de resolução. Por padrão elas medem até 5 volts e alguns pinos têm funções especializadas.

Em resumo, o Arduino é um kit de desenvolvimento, que pode ser visto como uma unidade de processamento capaz de mensurar variáveis do ambiente externo, transformadas em um sinal elétrico correspondente, através de sensores ligados aos seus terminais de entrada. De posse da informação, ele pode processá-la computacionalmente. Por fim, ele pode ainda atuar no controle ou no acionamento de algum outro elemento eletroeletrônico conectado ao terminal de saída. (RENNNA et al., 2013, p.3)

Existem outros modelos de Arduino: Arduino Leonardo, Arduino Due, Arduino Yún, Arduino Micro, etc. Cada um deles possui suas particularidades, como modelos diferentes de microcontroladores, quantidade de pinos e alguns deles têm aplicações específicas, esse é o caso do Arduino Robot, que é o primeiro Arduino com rodas e é voltado ao desenvolvimento de robôs.

2.28.2 Arduino IDE

O ambiente de desenvolvimento Arduino é o ArduinoIDE, como mostra a IMAGEM 12 disponibilizado no site do próprio site do fabricante. A IDE foi desenvolvida em JAVA e possui várias funções que auxiliam no desenvolvimento do software como o realce de sintaxe e com apenas um clique o usuário pode enviar o código desenvolvido para o Arduino.

A linguagem de desenvolvimento tem origem em *Wiring* (plataforma de prototipagem eletrônica de hardware livre composta por uma linguagem de programação, um ambiente de desenvolvimento integrado (IDE) e um microcontrolador de placa única). Assim como citado por Renna et al. (2013): “Os programas para o Arduino são implementados tendo como

referência a linguagem C++. Preservando sua sintaxe clássica na declaração de variáveis, nos operadores, nos ponteiros, nos vetores, nas estruturas e em muitas outras características da linguagem”

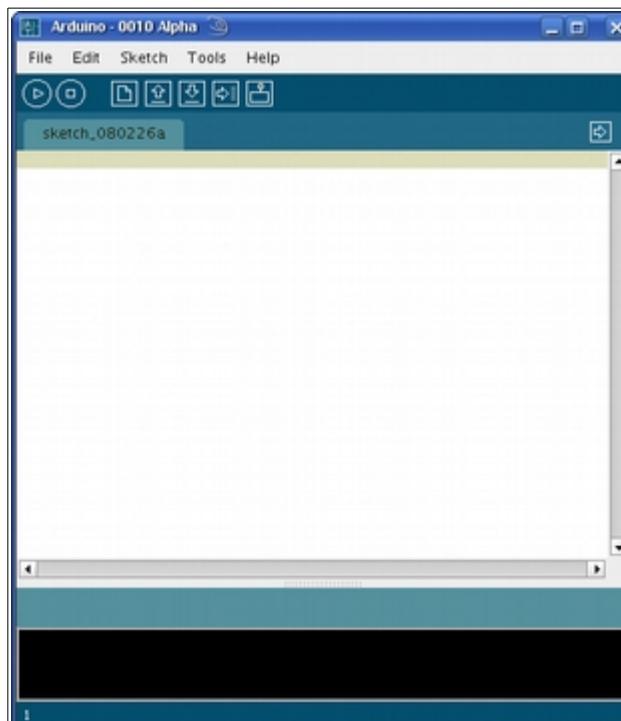


Imagen 12: Arduino IDE

A linguagem Arduino possui duas funções que essencialmente são implementadas e fazem parte da estrutura código, como mostra o QUADRO 14:

```
void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

QUADRO 14: Exemplo de Código Arduino

- Setup() - A função *setup()* é chamada quando o código inicia. É utilizada para inicializar variáveis, modos de pinagem, iniciar o uso de bibliotecas, etc. A função *setup()* apenas executa uma vez, depois que a placa Arduino é ligada ou reiniciada.

- Loop() - A função *loop()* é executada depois que a função *setup()* termina de executar. A função *loop()* executa constantemente, fazendo com que o programa se modifique e responda enquanto executa.

2.28.3 Ethernet Shield

O *Ethernet Shield* (como mostra a IMAGEM 13) é uma extensão do Arduino, que funciona como uma placa de rede, permitindo que o mesmo se conecte à Internet. Dessa forma as funcionalidades do Arduino aumentam. É possível controlá-lo remotamente, hospedar páginas WEB, trabalhar com *sockets*, etc.

O *Ethernet Shield* é baseado no *Wiznet W5100 Ethernet Chip*. O *Wiznet W5100* fornece conexão com a internet através de protocolos TCP e UDP. Ele suporta até quatro *sockets* conectados simultaneamente. O Ethernet Shield possui uma entrada padrão RJ45.

O *shield* possui também uma conexão para cartão SD que pode ser usado como servidor de arquivos. O Arduino se comunica com o *shield* e com o cartão SD através de SPI bus (através do cabeçalho ICSP).



IMAGEM 13: Ethernet Shield

2.28.4 Sensor Ultrassônico HC SR-04

O sensor ultrassônico (IMAGEM 14) funciona como um detector de objetos próximos. Esse sensor tem dois componentes: um emissor de ultrassons, que envia pulsos ultrassônicos de curta duração, que refletem no objeto a ser detectado e retorna um ou mais pulsos de reflexão, sendo captados por um microfone.



IMAGEM 14: Sensor Ultrassônico HC SR-04

O cálculo para medir a distância percorrida pelo som é: o tempo entre a emissão do som pelo componente emissor e a a captura do mesmo, pelo componente microfone ultrassônico, multiplicado pela velocidade do som ($331 + 0,6 * \text{temperatura ambiente em } ^\circ\text{C}$), dividido por 2, isso porque o som percorre a distância duas vezes.

2.28.5 Sensor de Chuva YL-83

O sensor de chuva Yl-83, como exibido na IMAGEM 15, possui trilhas metálicas em sua superfície que quando estão secas, a resistência entre elas é muito grande, resultando o sinal próximo a 0 V. Ao cair água nas trilhas a resistência aumenta, fazendo com que o sinal aumente gradualmente de acordo com a intensidade de água sobre as trilhas, até o valor máximo 5 V.

O sensor funciona de duas formas:

- Analógica – Retorna um valor entre 0 e 1023. Esses valores se alteram ascendentemente, de acordo com a quantidade de chuva.
- Digital – Retorna 0 ou 1. Se estiver chovendo retorna 0, senão 1.



IMAGEM 15: Sensor de Chuva Yl-83

2.29 PHONEGAP

O PhoneGap é um *framework open source*, para desenvolvimento de aplicações móveis multiplataforma nativas, que utiliza tecnologias web como linguagem de desenvolvimento HTML5 e JavaScript. O projeto iniciou em 2008, no iPhoneDevCamp, pela empresa Nytohi, com o intuito de simplificar o desenvolvimento de aplicações multiplataforma. Em 2011, o código fonte foi doado a Apache Software Foundation, onde foi batizado de Apache Cordova. O PhoneGap é uma distribuição do Apache Cordova, com algumas ferramentas adicionais.

O PhoneGap permite que o desenvolvedor acesse funções nativas do dispositivo a partir do JavaScript. Isso permite que os aplicativos desenvolvidos sobre ele utilizem apenas tecnologias web.

Para compilar os aplicativos, ele utiliza os SDK's padrões de cada plataforma. Suporta as seguintes plataformas: iOS, Android, Blackberry OS 6+, Blackberry 10, Windows Phone 8, Ubuntu e Firefox OS.

Como está disposto na imagem a seguir (IMAGEM 16), o Android é o sistema operacional mais utilizado em dispositivos portáteis, com mais de 1 bilhão de usuários ativos, seguido pelo IOS que tem cerca de 800 milhões de usuários ativos e o Windows Phone com 60 milhões de usuários ativos.

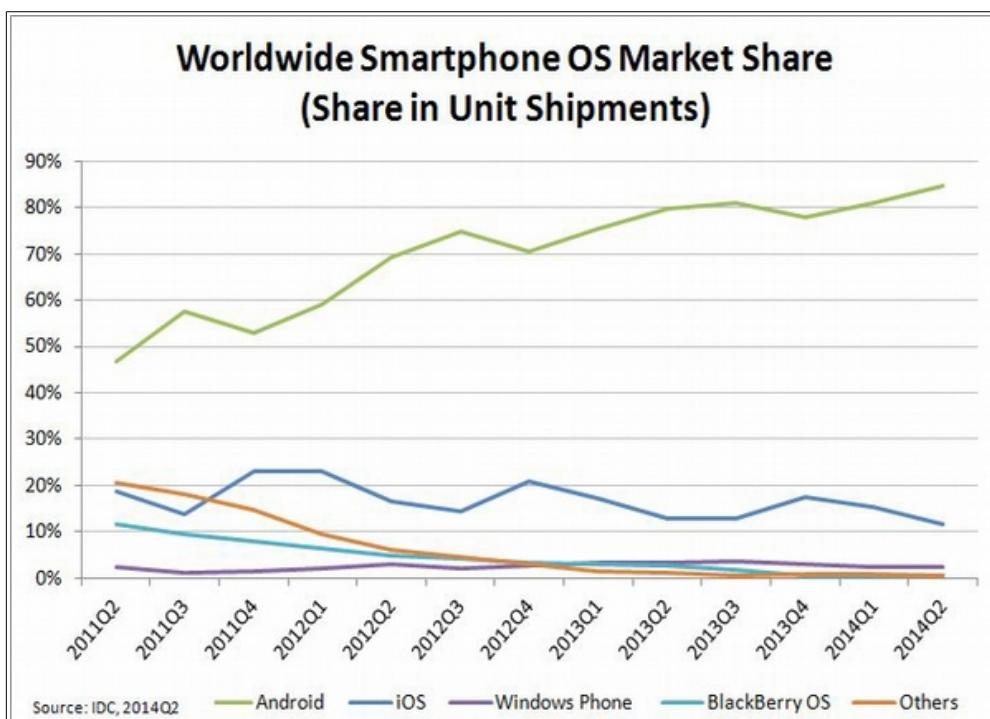


IMAGEM 16: Gráfico de utilização de S.O. em smartphones. Fonte: Hamann (2014)

2.30 GENYMOTION

Genymotion é um emulador gratuito de sistema operacional Android, capaz de proporcionar a seu usuário a possibilidade de experimentar versões e aplicativos desse ambiente do Google sem a necessidade de obter dispositivos novos ou realizar alterações em seus *tablets* ou *smartphones*.

2.31 ECLIPSE

Eclipse é um IDE (*Integrated Development Environment* – Ambiente de desenvolvimento integrado) multiplataforma para desenvolvimento com suporte a muitas linguagens, incluindo C/C++, PHP, ColdFusion, Python, Scala e Android. É desenvolvido em Java e distribuído sob a licença EPL (Eclipse Public License), pela Eclipse Foundation. Atualmente faz parte do *kit* de desenvolvimento de software recomendado para desenvolvedores Android.

IDE, do inglês *Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar esse processo. [...] O Eclipse tornou-se famoso não apenas por ser uma IDE de primeira qualidade, desenvolvida com a extensibilidade em mente, mas também porque ela tinha uma interface gráfica levíssima e que era igual à do sistema operacional onde ele estivesse sendo executado. (SERSON, 2008, p. 7)

2.32 APTANA STUDIO

Aptana Studio é um IDE (*Integrated Development Environment* – Ambiente de desenvolvimento integrado) para desenvolvimento WEB escrito em Java que suporta as linguagens PHP, Python, Ruby, Ruby on Rails, CSS3, HTML5, JavaScript, ScriptDoc, XML e texto comum, embora também seja possível configurá-lo para suportar Adobe AIR e Bibliotecas AJAX. É baseado no Eclipse, programa similar que por sua vez tem foco no desenvolvimento de linguagens de programação.

Principais funcionalidades:

- Assistente de codificação;
- Funções de autocompletar;
- Destacamento de sintaxe;
- Suporte completo a funções e objetos de todas linguagens citadas;
- Suporte a informações de compatibilidade com navegadores;
- Suporte ao protocolos FTP (File Transfer Protocol – Protocolo de Transferência de Arquivos) e SFTP (SSH File Transfer Protocol – Protocolo de Transferência de Arquivo Criptografado);
- Depurador embutido;

2.33 ROBOMONGO

Robomongo (IMAGEM 17) é um ambiente multiplataforma de código aberto para administração do banco de dados não relacional MongoDB. A ferramenta incorpora o mesmo motor JavaScript utilizado no *shell* do banco de dados. Tudo que você pode escrever no *shell* do MongoDB, você pode escrever no Robomongo.

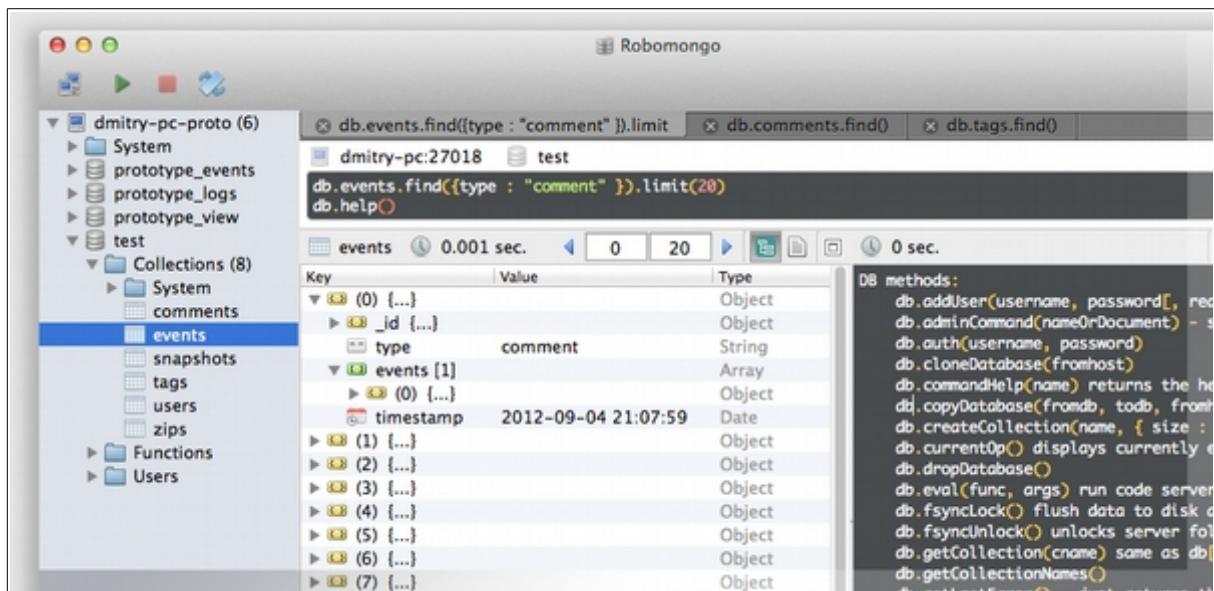


IMAGEM 17: Interface principal do Robomongo

Principais funções:

- Destacamento de sintaxe;

- Funções de autocompletar;
- Vários modos de visualização de documentos;
- Múltiplas áreas de comando;

2.34 STARUML

É uma ferramenta de código aberto utilizada para desenvolvimento de diagramas UML, distribuído sob a licença GPL. Inicialmente desenvolvido em 1996, utilizando a linguagem de programação Delphi. O objetivo do projeto era substituir ferramentas pagas. Ele permite que o usuário faça a modelagem UML de seus projetos de forma simples e prática e também oferece algumas funções como engenharia reversa e gerador de códigos.

2.34.1 Unified Modeling Language (UML)

A UML (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma linguagem para a elaboração da estrutura de projetos de software.

Em 1993 existiam três métodos para modelagem de software que estavam crescendo no mercado: Booch'93 de Grady Booch, OMT 2 de James Rumbaugh e OOSE de Ivar Jacobson. Os três se uniram para unificar os métodos e em 1996 foi lançada a primeira versão do UML, que unificou os três métodos.

A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir sistemas de informações corporativas a serem distribuídos a aplicações baseadas em Web e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas. Apesar de sua expressividade, não é difícil compreender e usar a UML. Aprender a aplicar a UML de maneira efetiva tem início com a formação de um modelo conceitual da linguagem, o que pressupõe o entendimento de três principais elementos: os blocos básicos de construção da UML, as regras que determinam como esses blocos de construção deverão ser combinados e alguns mecanismos básicos que se aplicam a toda linguagem. (BOOCH; RUMBAUGH; JACOBSON, 2006, p.13)

A UML é destinada a visualizar, especificar, construir e documentar os artefatos de um sistema de um software.

2.35 PENCIL PROJECT

Pencil Project (ou apenas Pencil, IMAGEM 18) é um programa gratuito multiplataforma para prototipação de interfaces gráficas. É fácil de instalar e usar, provendo modelos para compatibilidade com as interfaces mais famosas.

O sistema é compatível com os sistemas operacionais Mac OSX, Linux e Windows, mas também pode ser instalado como extensão do navegador de internet Mozilla Firefox, tornando-o portável para um grande número de plataformas.

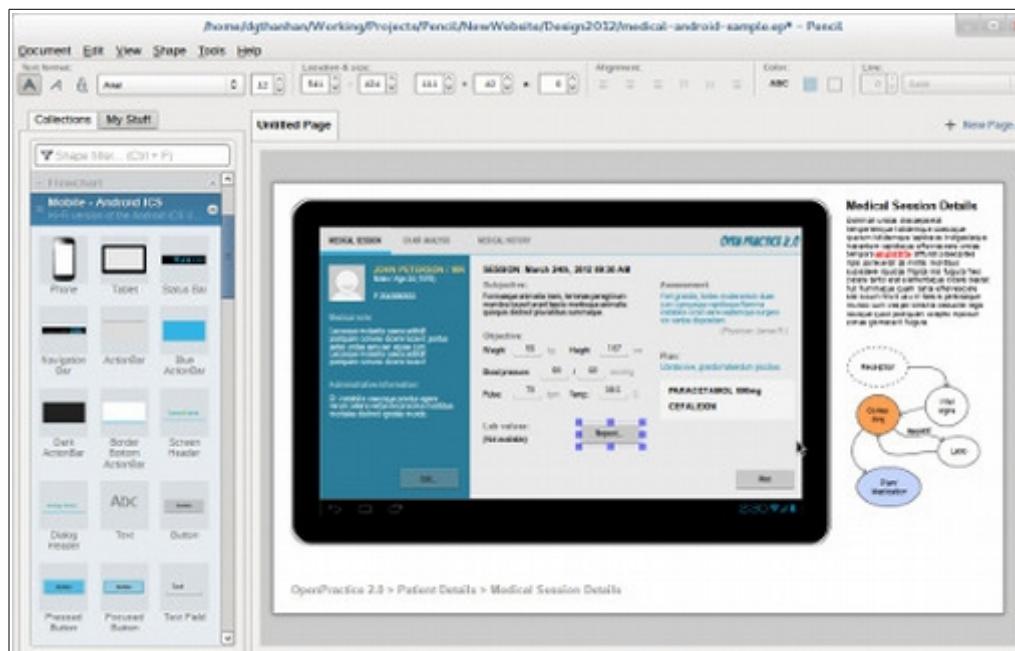


IMAGEM 18: Interface principal do Pencil

Pencil é mantido pela empresa Evolus Software e distribuído sobre a licença GPL v2 (*GNU Public License* versão 2), entregando à comunidade a liberdade de usar, modificar e redistribuir a ferramenta.

2.36 FRITZING

Fritzing é uma ferramenta *open source* (software livre) utilizada para o desenvolvimento da prototipação e documentação de projetos eletrônicos, desenvolvido na *University of Applied Sciences of Potsdam*(Universidade de Ciências Aplicadas de Potsdam)

na Alemanha. Ele permite que o usuário modele o protótipo de forma simples e intuitiva, apenas arrastando os componentes que são exibidos em uma barra lateral, permite que sejam criados circuitos eletrônicos, como no mundo real, conectando componentes, *protoboards* e placas controladoras. É possível adicionar componentes que não estão disponíveis nativamente na ferramenta e também é possível desenhar seus próprios componentes.

2.37 GIT

Criado em 2005, o Git é um sistema de controle de versão distribuído e um sistema de gerenciamento de código fonte (SCM), com ênfase em velocidade. O Git foi inicialmente projetado e desenvolvido por Linus Torvalds para o desenvolvimento do Kernel Linux, mas foi adotado por muitos outros projetos.

O Git é diferente dos outros SCMs, pois é um sistema de gerenciamento de código-fonte distribuído. Isso significa que, em vez de ter um único repositório em seu servidor, que todos seus colegas utilizam para efetuar o *checkout* de cópias locais(SCM centralizado, cliente-servidor) cada membro da equipe tem seu próprio repositório, além de uma cópia local, e todos fazem o *push* de uma cópia desse repositório para um repositório remoto. (CARNEIRO JUNIOR, 2011, p. 373)

Cada diretório de trabalho do Git é um repositório com um histórico completo e habilidade total de acompanhamento das revisões, independente do acesso a um servidor.

2.38 GITHUB

O GitHub é um Serviço de hospedagem e compartilhamento para projetos que usam o controle de versionamento Git. Lançado em 2008, o GitHub foi desenvolvido em Ruby on Rails pelos desenvolvedores da *Logical Awesome* (Chris Wanstrath, PJ Hyett e Tom Preston – Wernder). Possui planos comerciais e gratuitos para projetos de código aberto.

Existem outros sites onde podemos colocar projetos, mas o que torna o GitHub interessante são alguns recursos de redes sociais. Nele podemos seguir projetos de nosso interesse ou mesmo outros desenvolvedores.

2.39 LIBREOFFICE

LibreOffice é uma suíte de aplicativos para escritório multiplataforma. Ela utiliza o formato OpenDocument (ODF) — formato homologado como ISO/IEC 26300 e NBR ISO/IEC 26300 — e é compatível com demais formatos, incluindo as extensões da suíte Microsoft Office.

O objetivo do projeto é eliminar o abismo digital e dar poderes a todos os cidadãos, apoiando a preservação dos idiomas nativos e evitando que os usuários fiquem presos a softwares fechados e a formatos de arquivos proprietários.

No dia 28 de setembro de 2010, os antigos desenvolvedores do projeto OpenOffice.org decidiram sair da empresa detentora e lançar sua própria suíte de aplicativos para escritório. Junto com o projeto do LibreOffice, que inicialmente pensava-se em ser apenas um nome provisório para o novo projeto, nasceu a The Document Foundation, fundada em 28 de setembro de 2010, que é a atual mantedora do projeto. A bifurcação deu-se na versão 3.3 do OpenOffice.org, número da primeira versão lançada do LibreOffice.

A comunidade LibreOffice não só desenvolve o *software*, como oferece suporte gratuito, com base em trabalho voluntário. Os utilizadores do LibreOffice podem obter ajuda on-line da comunidade através de listas de discussão e fóruns. Existem também outros sites geridos pelos utilizadores que oferecem dicas e tutoriais gratuitos. (AFONSO et al., 2014, p. 7)

O LibreOffice está licenciado sob os termos da GPLv3 (ou posteriores) e pela MPL - Mozilla Public License (ou posteriores). Significa que o usuário está livre para usá-lo para fins pessoais e comerciais, copiar e distribuir o *software* e modificar ou reestruturar o código-fonte e criar obras derivadas.

A suíte LibreOffice é composta das seguintes ferramentas:

- Writer – Editor de texto
- Calc – Planilha eletrônica
- Impress – Editor de apresentações
- Draw – Editor de desenhos
- Math – Editor de fórmulas
- Base – Banco de dados

3 DESENVOLVIMENTO

Neste capítulo serão apresentados os passos e conceitos utilizados no desenvolvimento do protótipo, como requisitos, casos de uso, diagrama de atividades e exemplos de codificação.

Para facilitar o levantamento dos requisitos e desenvolvimento do código fonte foi utilizada a metodologia de prototipação junto a programação extrema (*extreme programming*), numa evolução constante de todas as partes que compõe o trabalho.

3.1 REQUISITOS

A seguir serão apresentados os requisitos funcionais e não funcionais que moldam a base do protótipo. Para levantamento desses dados foi utilizada uma técnica de pesquisa por questionário (vide ANEXO A), que era composto de perguntas de múltipla escolha sobre as principais funcionalidades da ferramenta, como preferência de plataforma, forma de acesso, disponibilidade e viabilidade.

3.1.1 Requisitos Funcionais

Os requisitos funcionais (QUADRO 15) definem as funções de um sistema. Uma função é descrita como um conjunto de entradas, comportamentos e saídas. Os requisitos funcionais podem ser cálculos, detalhes técnicos e outras funcionalidades específicas que definem o que um sistema, idealmente, deverá de realizar.

RF01 – Manter leituras do ponto de medição:	UC02, UC05
<ul style="list-style-type: none"> ○ Data de medição; ○ Nível do Rio (em metros); ○ Estado de chuva: <ul style="list-style-type: none"> ■ Nula; 	

<ul style="list-style-type: none"> ▪ Moderada; ▪ Forte; 	
RF02 – Manter pontos analisados de geo elevação: <ul style="list-style-type: none"> ◦ Elevação (metros); ◦ Latitude; ◦ Longitude; 	UC01
RF03 – Possuir um mapa interativo que exiba as áreas inundadas através de uma camada sobreposta na cor azul;	UC01
RF04 – Exibir a medição atualizada do nível do rio (em metros);	UC02, UC03
RF05 – Exibir o estado atual de chuva: <ul style="list-style-type: none"> ◦ Nula; ◦ Moderada; ◦ Forte; 	UC02, UC04
RF06 – Exibir histórico do nível de rio e estado chuva através de gráficos;	UC05
RF07 – Exibir um painel de alerta de enchentes e inundações com os estados: <ul style="list-style-type: none"> ◦ Normal; ◦ Alerta; ◦ Inundação; 	UC06
RF08 – Manter históricos de enchentes: <ul style="list-style-type: none"> ◦ Data; ◦ Nível máximo do rio; 	UC13
RF09 – Fornecer uma galeria colaborativa integrada ao Facebook com imagens de inundações: <ul style="list-style-type: none"> ◦ Qualquer pessoa pode visualizar as imagens disponíveis; ◦ Para enviar uma imagem o usuário deve realizar <i>login</i> com uma conta do Facebook; ◦ Manter uma descrição para imagem; 	UC10, UC11
RF10 – Disponibilizar hiperlinks para sites de utilidade pública: <ul style="list-style-type: none"> ◦ CEOPS – Furb; ◦ Defesa Civil de Santa Catarina; ◦ Prefeitura de Timbó; ◦ EPAGRI/CIRAM; 	UC08

<ul style="list-style-type: none"> ◦ SAMAE; <p>RF11 – Disponibilizar no aplicativo móvel um sistema de alerta através de notificações:</p> <ul style="list-style-type: none"> ◦ Permitir o cadastro de múltiplos pontos de verificação; ◦ O cadastro do ponto pode ser feito por digitação de endereço ou localidade atual por GPS; ◦ Realizar uma verificação de alerta a cada dez minutos; ◦ Deve ser emitida uma notificação se a diferença entre a altura dos locais cadastrados e o nível atual do rio for menor ou igual a dois metros. 	UC12
<p>RF12 – Permitir acesso ao histórico de leituras através de um servidor WebService:</p> <ul style="list-style-type: none"> ◦ Para utilização de método deve ser informado via parâmetro: <ul style="list-style-type: none"> ▪ Data e hora da leitura; ▪ Quantidade de leituras para retorno; ◦ Retornar os dados em formato JSON; 	UC14
<p>RF13 – Disponibilizar um simulador de inundação que exiba as informações através de um mapa interativo;</p>	UC09
<p>RF14 – Exibir a previsão do tempo para os próximos dias através de ferramentas de terceiros;</p> <ul style="list-style-type: none"> ◦ Mapa meteorológico; ◦ Imagem de satélite; ◦ Previsão do tempo para os próximos dias; 	UC07
<p>RF15 – Disponibilizar no aplicativo móvel uma ferramenta para verificar incidências de enchente no local atual;</p>	UC13
<p>RF16 – No acesso móvel através do navegador de internet as funções devem ser limitadas, direcionando à instalação do aplicativo:</p> <ul style="list-style-type: none"> ◦ Não disponibilizar acesso ao simulador de inundações; ◦ Não disponibilizar imagens de satélite; ◦ Não disponibilizar galeria de fotos; 	UC07, UC09, UC10, UC11
<p>RF17 – Manter imagens da galeria colaborativa:</p> <ul style="list-style-type: none"> ◦ Imagem; ◦ Cidade; ◦ Bairro; 	UC10, UC11

<ul style="list-style-type: none"> ◦ Rua; ◦ Data; ◦ Hora. 	
--	--

QUADRO 15: Requisitos funcionais

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais (QUADRO 16) estão relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Não é preciso o cliente dizer sobre eles, pois normalmente são características especificadas pelo desenvolvedor do software.

RNF01 – A persistência das informações deve ser feita através do banco de dados não relacional MongoDB;
RNF02 – A régua automatizada de medição de nível de rio deve utilizar a plataforma de prototipação de hardware livre Arduino;
RNF03 – O sistema deve conter um serviço (no Linux: <i>Daemon</i>) implementado em Java 7 para realizar a leitura da régua de medição e armazenar as informações no banco de dados.
RNF04 – A conexão entre o serviço e a régua de medição deve ser realizada através de conexão <i>Socket</i> de rede TCP/IP;
RNF05 – A linguagem de programação utilizada na implementação da régua automatizada deve ser C;
RNF06 – A interface WEB deve ser implementada em HTML5 (HTML5, CSS3, JavaScript), PHP5.5 e Twitter BootStrap;
RNF07 – O site deve ser hospedado num servidor Apache;
RNF08 – O servidor Apache, MongoDB e serviços devem estar hospedados numa instância EC2 da Amazon AWS;
RNF09 – A interface WEB deve ser compatível com os navegadores Mozilla Firefox 3.6, Google Chrome, Opera, Safari 4 e Internet Explorer 9, ou superior;
RNF10 – O aplicativo móvel deve ser compatível com os sistemas operacionais Firefox OS, Android, Iphone e Windows Phone;
RNF11 – O desenvolvimento multiplataforma móvel deve ser realizado através do <i>framework</i> PhoneGap, linguagem HTML5 e Ionic Framework;

RNF12 – O sistema deve utilizar o serviço de mapas interativos Google Maps API e de gráficos Google Charts;
RNF13 – Devem ser utilizadas as seguintes bibliotecas com o PHP:
<ul style="list-style-type: none"> ◦ Facebook; ◦ MongoDB;
RNF14 – O WebService para disponibilização de dados deve ser construído através da ferramenta Apache Axis2;
RNF15 – O Apache Axis2 deve ser executado sobre o servidor de aplicação Apache Tomcat 8;
RNF16 – Disponibilizar o acesso ao sistema apenas com conexão à internet;
RNF17 – A interface WEB deve possuir design responsivo para <i>desktop</i> e móvel;
RNF18 – O tempo de atualização das leituras deve variar conforme o nível do rio:
<ul style="list-style-type: none"> ◦ Nível normal: 2 h; ◦ Nível de alerta: 30 min; ◦ Nível critico: 15 min;

QUADRO 16: Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Em sequência serão apresentados diagramas e especificações de casos de uso e atividades, com o objetivo e detalhar e clarificar as funcionalidades e forma de operação do sistema.

3.2.1 Casos de Uso

O diagrama de casos de uso (IMAGEM 19) descreve as funcionalidades propostas para um novo sistema que será projetado e é uma excelente ferramenta para o levantamento dos requisitos funcionais. Um caso de uso representa uma unidade de interação entre um usuário (humano ou máquina) e o sistema. Sua especificação (QUADRO 17) detalhada descreve os requisitos atendidos, atores envolvidos, cenário principal e alternativos, aprofundando o entendimento do caso.

3.2.1.1 Diagrama de Casos de Uso

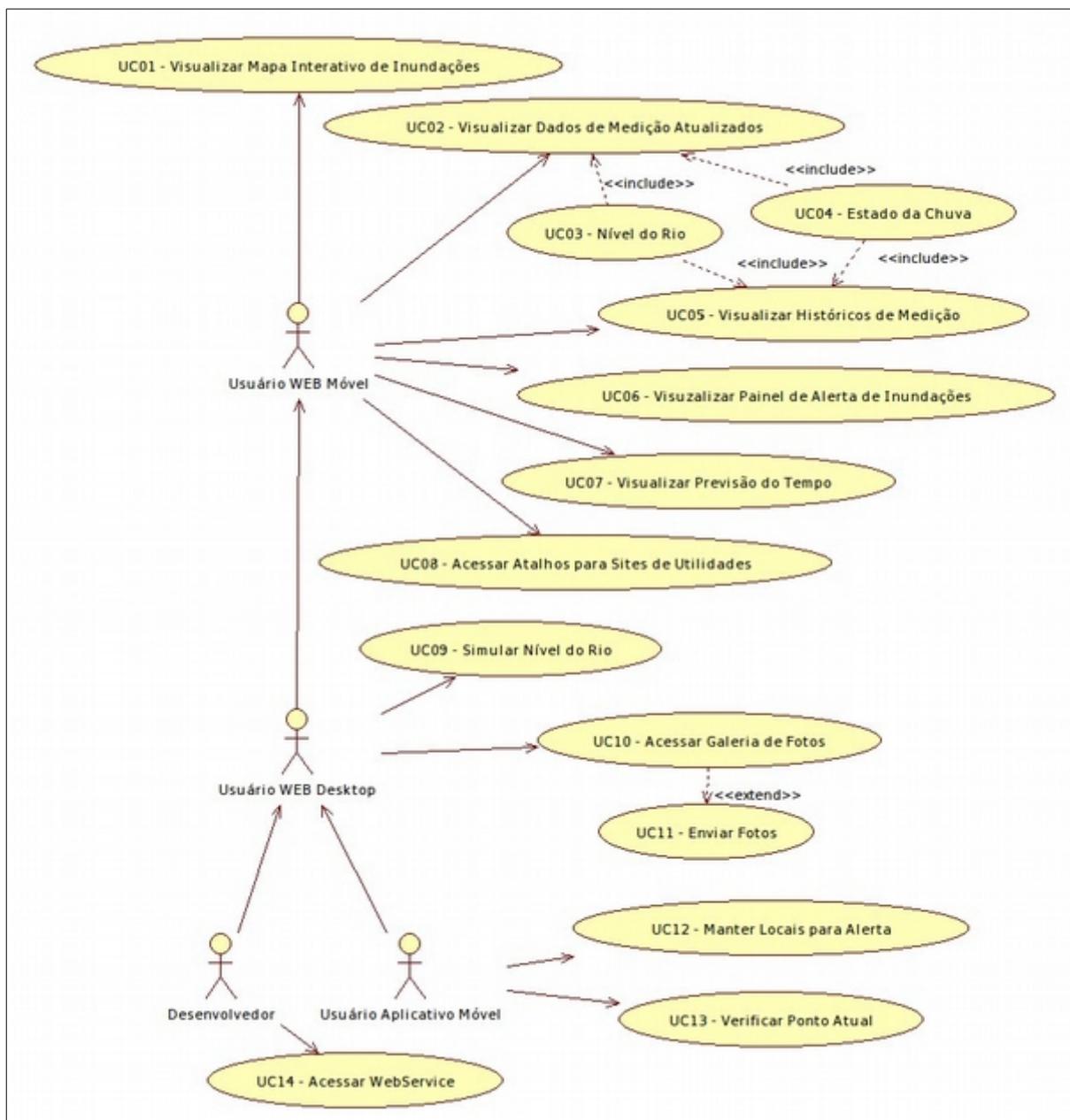


IMAGEM 19: Diagrama de casos de uso

3.2.1.2 Especificação dos Casos de Uso

UC01 – VISUALIZAR MAPA INTERATIVO DE INUNDAÇÕES	
REQUISITOS ATENTIDOS	RF02, RF03.
ATORES	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo

ENVOLVIDOS	Móvel, Desenvolvedor.
CENÁRIO PRINCIPAL	<p>Ao acessar o sistema os usuários visualizam um mapa interativo das ruas de Timbó. Este mapa exibe, com uma coloração azulada, as áreas inundadas.</p> <p>Essas áreas inundadas são definidas pelo servidor, que faz os cálculos necessários baseados nos dados de elevação das coordenadas geográficas salvas no banco.</p>

UC02 – VISUALIZAR DADOS DE MEDIÇÃO ATUALIZADOS	
REQUISITOS ATENTIDOS	RF01, RF04, RF05.
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
CENÁRIO PRINCIPAL	O usuário terá acesso a um quadro com as últimas medições do nível do rio e de estados de chuva baseados no banco de dados.

UC03 – NIVEL DO RIO	
REQUISITOS ATENTIDOS	RF04.
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
CENÁRIO PRINCIPAL	O usuário visualizará a medição atual do rio onde a régua está posicionada, sendo a medição feita em metros.

UC04 – ESTADO DA CHUVA	
REQUISITOS ATENTIDOS	RF05.
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
CENÁRIO PRINCIPAL	O usuário visualizará o estado da chuva do local onde está posicionado a régua de medição, sendo exibido como: sem chuva, chuva moderada ou chuva forte.

UC05 – VISUALIZAR HISTÓRICOS DE MEDIÇÃO	
REQUISITOS ATENTIDOS	RF01, RF06.
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Acessar o menu históricos de medição
CENÁRIO PRINCIPAL	O usuário encontrará dois gráficos, um de nível do rio e outro de estado da chuva, que exibem as medições das últimas 24 horas.

UC06 – VISUALIZAR PAINEL DE ALERTA DE INUNDAÇÕES	
REQUISITOS ATENTIDOS	RF07
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
CENÁRIO PRINCIPAL	O usuário visualiza na tela principal do sistema um painel que exibe o estado atual do rio.

CENARIO ALTERNATIVO 01	Caso haja alguma falha no serviço, o painel a ser visualizado informa o estado do rio referente à última medição.
------------------------	---

UC07 – VISUALIZAR PREVISÃO DO TEMPO	
REQUISITOS ATENTIDOS	RF14, RF16
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Acessar menu de previsão do tempo.
CENÁRIO PRINCIPAL	O usuário visualiza a previsão do tempo através de um mapa meteorológico, imagem de satélite e quadro de temperaturas e precipitações.

UC08 – ACESSAR ATALHOS PARA SITES DE UTILIDADES PÚBLICAS	
REQUISITOS ATENTIDOS	RF10
ATORES ENVOLVIDOS	Usuário WEB Móvel, Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Acessar menu de utilidades públicas.
CENÁRIO PRINCIPAL	O usuário visualiza uma lista de sites de utilidade pública, com hiperlinks para acesso.

UC09 – SIMULAR NÍVEL DE RIO	
REQUISITOS ATENTIDOS	RF13, RF16
ATORES ENVOLVIDOS	Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Acessar menu de simulação de rio.
CENÁRIO PRINCIPAL	O usuário altera o nível do rio através de um controle, sendo exibido no mapa o resultado da inundação.

UC10 – ACESSAR GALERIAS DE FOTOS	
REQUISITOS ATENTIDOS	RF09, RF16, RF17
ATORES ENVOLVIDOS	Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Acessar menu de galeria.
CENÁRIO PRINCIPAL	Visualizar uma galeria colaborativa de fotos de enchentes e inundações enviadas pelos usuários, com descrição, local e data.

UC11 – ENVIAR FOTOS	
REQUISITOS ATENTIDOS	RF09, RF16, RF17
ATORES ENVOLVIDOS	Usuário WEB Desktop, Usuário Aplicativo Móvel, Desenvolvedor.
PRÉ-CONDIÇÃO	Realizar login com conta do Facebook.
CENÁRIO PRINCIPAL	Clicar no botão “Enviar Fotos”, selecionando a imagem e preenchendo as informações:

	<ul style="list-style-type: none"> ○ Cidade; ○ Bairro; ○ Rua; ○ Data; ○ Hora.
CENARIO ALTERNATIVO 01	Se não for adicionada nenhuma imagem, apresentar a mensagem “É necessário selecionar pelo menos uma imagem”.
CENARIO ALTERNATIVO 02	Se um dos campos não for preenchido, exibir a mensagem “Todas as informações são necessárias!”.
CENARIO ALTERNATIVO 03	Se houver algum erro ao enviar as fotos, exibir a mensagem “Não foi possível enviar as imagens. Tente novamente.”

UC12 – MANTER LOCAIS PARA ALERTA	
REQUISITOS ATENTIDOS	RF11
ATORES ENVOLVIDOS	Usuário Aplicativo Móvel.
PRÉ-CONDIÇÃO	Clicar em “Sistema de Alerta”.
CENÁRIO PRINCIPAL	Cadastrar / Remover localidade para alerta de inundação. Ao cadastrar: selecionar local via digitação ou localidade atual por GPS. Ao remover: Selecionar um local e clicar em remover.

UC13 – VERIFICAR PONTO ATUAL	
REQUISITOS ATENTIDOS	RF15
ATORES ENVOLVIDOS	Usuário Aplicativo Móvel
CENÁRIO PRINCIPAL	O usuário verifica a incidência de inundações no ponto atual, adquirido via GPS.
CENARIO ALTERNATIVO 01	Se não houve nenhuma enchente no local, exibir a mensagem: “Nenhuma inundação encontrada para este local”

UC14 – ACESSAR WEB SERVICE	
REQUISITOS ATENTIDOS	RF12
ATORES ENVOLVIDOS	Desenvolvedor.
CENÁRIO PRINCIPAL	Acessar WebService para resgatar no formato JSON dados de leitura, informando data e hora e quantidade de registros para retornar.
CENARIO ALTERNATIVO 01	Se a quantidade de leituras para retorno for maior que 15, retornar: {“erro”: true, “descricao”: “numero max de leituras = 15”}.
CENARIO ALTERNATIVO 02	Se não houver dados para a hora informada no parâmetro retornar medição anterior mais próxima.
CENARIO ALTERNATIVO 03	Se não houver dados para informar, retornar: {“erro”: true, “descricao”: “nenhum registro encontrado”}.

QUADRO 17: Especificação dos casos de uso

3.2.2 Diagrama de Atividades

Um diagrama de atividades (IMAGEM 20, IMAGEM 21 e IMAGEM 22) é essencialmente um gráfico de fluxo, mostrando a direção de uma atividade para outra e é empregado para fazer a modelagem de aspectos dinâmicos do sistema. Na maior parte, isso envolve a modelagem das etapas sequenciais em um processo computacional.

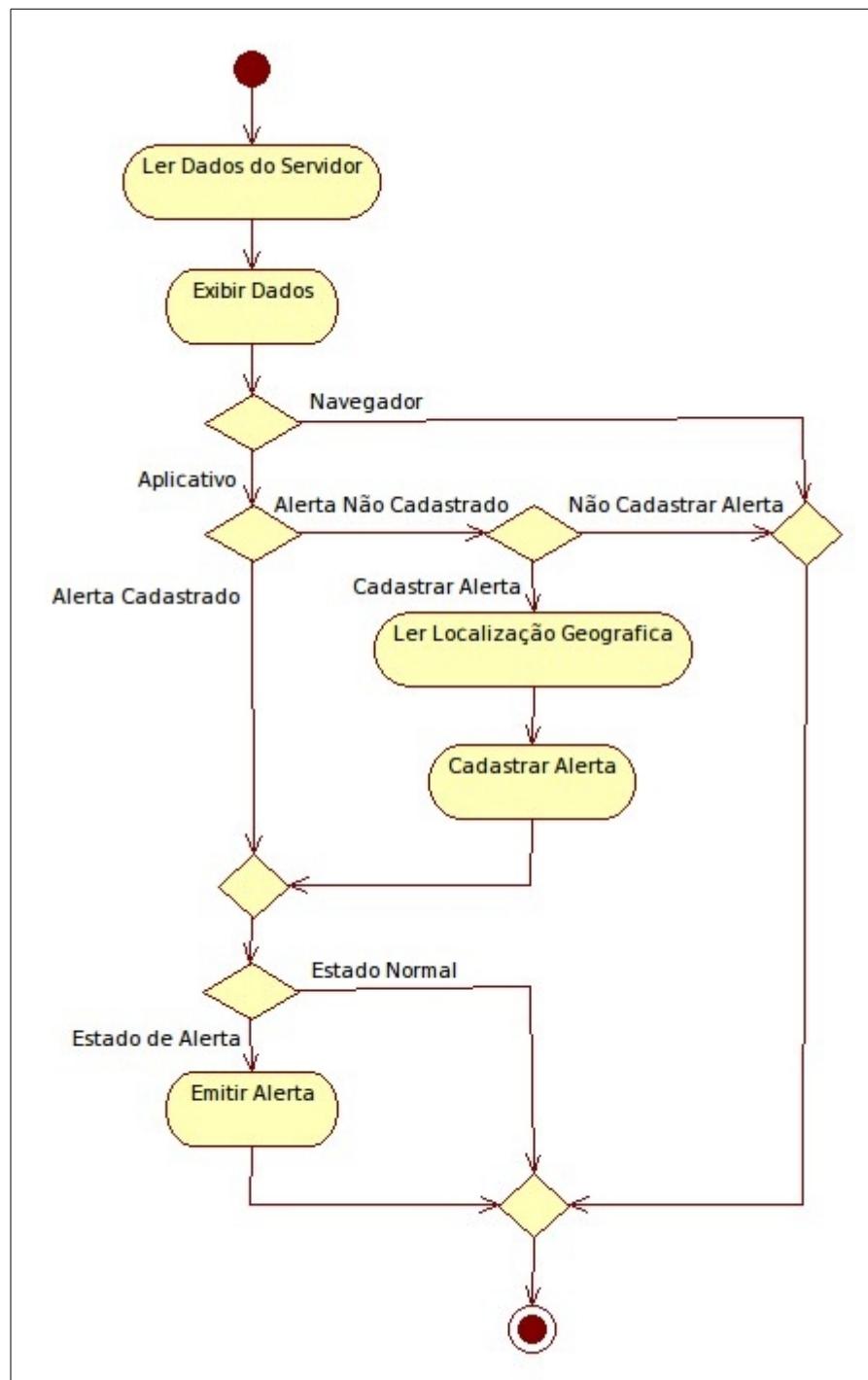


IMAGEM 20: Diagrama de atividades do cliente



IMAGEM 21: Diagrama de atividades do servidor

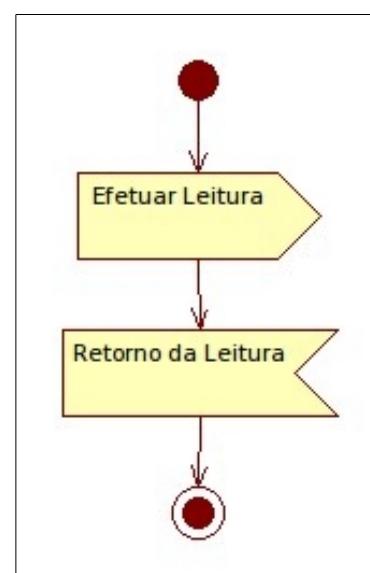


IMAGEM 22: Diagrama de atividades do ponto de medição

3.3 IMPLEMENTAÇÃO

Neste tópico serão apresentados os detalhes do desenvolvimento do código fonte da ferramenta, montagem eletrônica da régua de medição e configurações de servidores para hospedagem dos serviços necessários. Serão expostos quadros com exemplos de algoritmos e protótipos de interface gráfica.

3.3.1 Prototipação do software

Após o levantamento das especificações, como requisitos, casos de uso e fluxos de informação, foram definidas as formas e ambientes pelos quais os usuários têm acesso ao sistema. Para auxiliar essa tarefa foi utilizada a ferramenta Pencil Project, junto à imagens dos componentes de frameworks presentes no projeto, como Ionic Framework e Twitter BootStrap.

Na IMAGEM 23 é possível ver a interface gráfica definida para o aplicativo de dispositivos móveis e, na IMAGEM 24 e IMAGEM 25, a estrutura básica de interface para o site *desktop* e móvel, respectivamente.



IMAGEM 23: Protótipo de aplicativo móvel

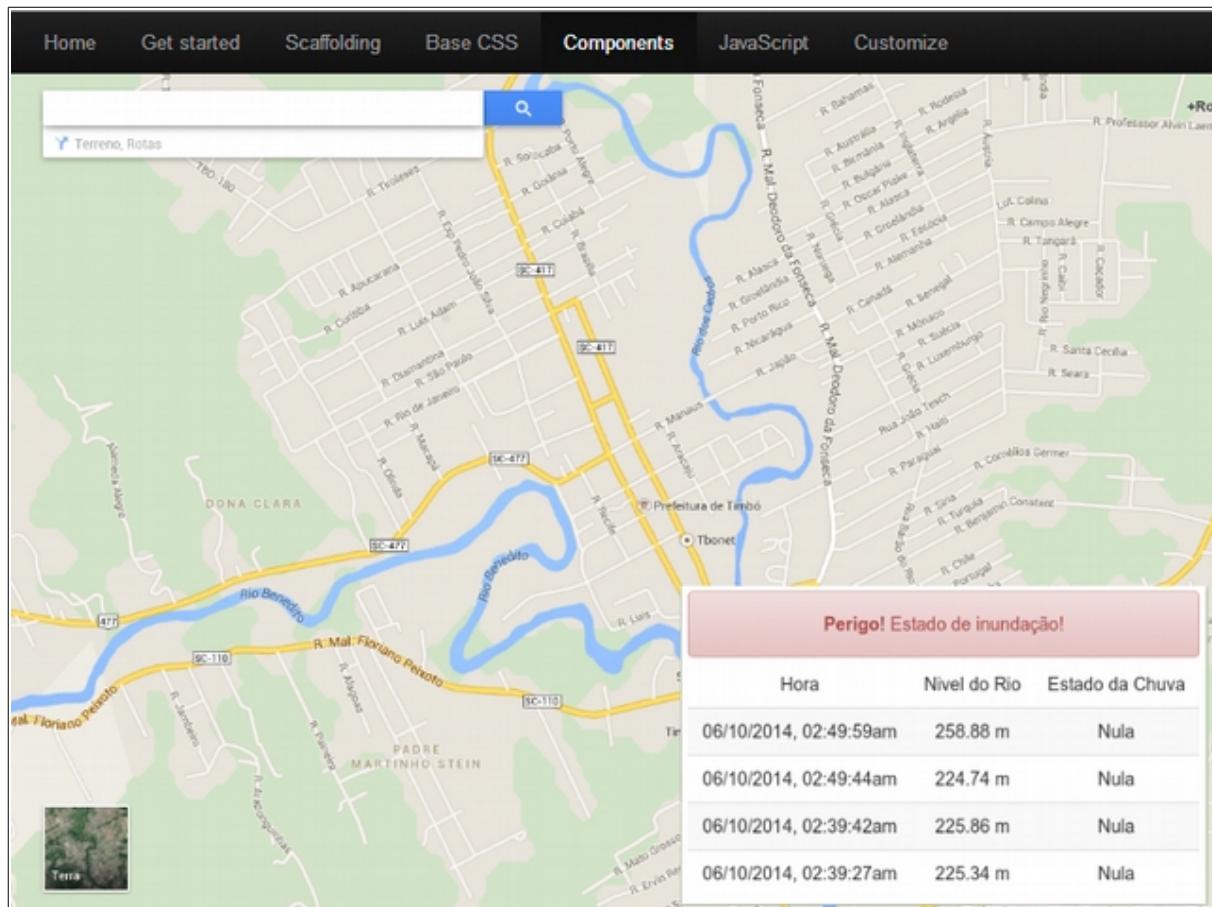
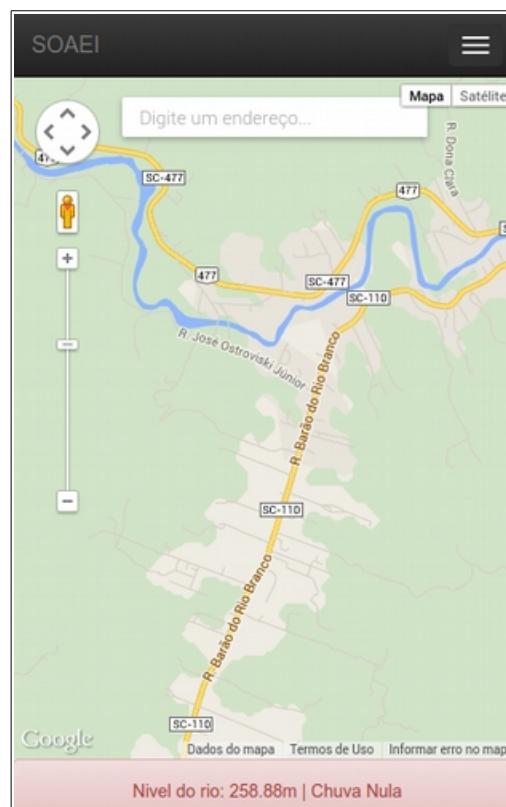
IMAGEM 24: Protótipo de site *desktop*

IMAGEM 25: Protótipo de site móvel

3.3.2 Programação Extrema

Normalmente no desenvolvimento de um software por uma equipe pequena são focadas etapas como levantamento de requisitos, implementação do código fonte e testes, mantendo sempre grande entrosamento e troca de informações entre os membros, facilitando a obtenção de um código e limpo e padronizado, como também uma ferramenta coesa e funções interligadas e de manutenção facilitada.

No desenvolvimento desse protótipo foram utilizadas as técnicas da programação extrema (em inglês: *extreme programming*), que possui as propriedades descritas acima como algumas de suas principais características. Foi utilizado o software Pencil Project para prototipação, Git como gerenciador de versão de código, junto ao GitHub. Optou-se pelo desenvolvimento de pequenas tarefas, dividindo o sistema por funções básicas, promovendo *commits* (envio de alterações de código ao repositório) curtos e constantes, o que evita conflitos de alterações.

Apesar de o desenvolvimento seguir uma sequência pré-determinada – pesquisa de mercado, levantamento de requisitos, especificação, desenvolvimento e testes –, em várias etapas do projeto foi necessário rever alguns pontos já concluídos, como requisitos ou protótipos de interface gráfica, por não serem realmente viáveis ou existir uma forma melhor de atender os objetivos. Tais alterações são comumente utilizadas na prototipação e desenvolvimento com programação extrema.

3.3.3 Ponto de Medição Automatizado

Para a medição do nível do rio e intensidade de chuva, foi desenvolvida uma régua automatizada, que se comunicará com um serviço na nuvem. A régua enviará as leituras cada vez que for feita uma solicitação.

A régua foi desenvolvida utilizando uma placa Arduino Uno, um sensor de distância ultrassônico HC SR-04, um *Ethernet Shield*, um sensor de chuva YL-83 e um led de cor vermelha para indicar quando é efetuada a leitura. O desenvolvimento do código da placa Arduino Uno foi feita na interface de desenvolvimento Arduino IDE.

O protótipo de arquitetura eletrônica da régua pode ser visto na IMAGEM 26, construído com a ferramenta Fritzing.

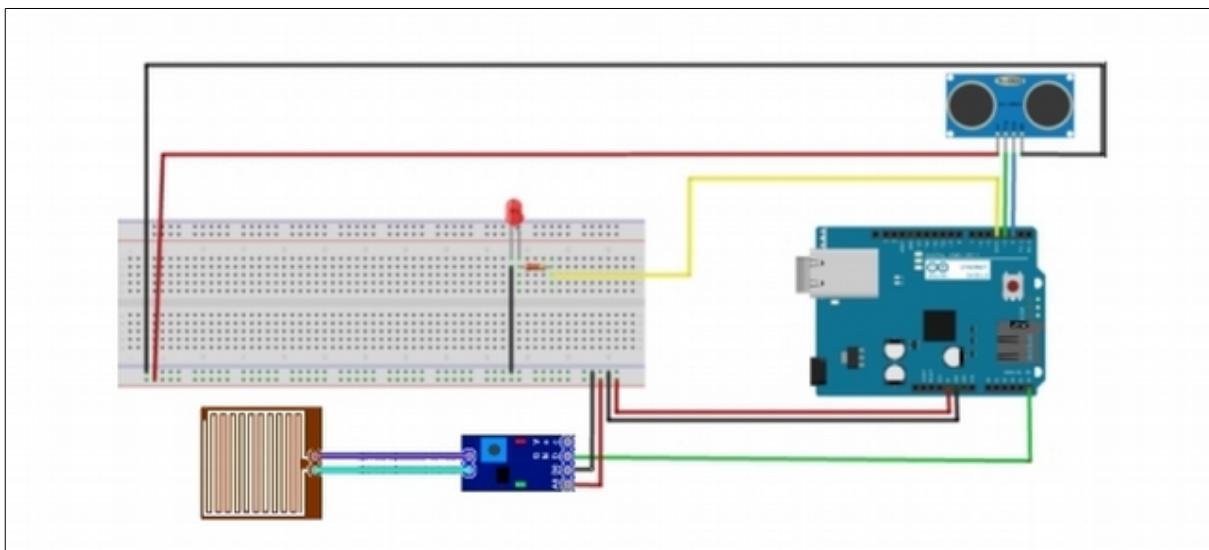


IMAGEM 26: Prototipação da régua de medição

O sensor de distância ficará posto sobre a água, para captar a distância entre os dois, assim calculando o nível em que o rio se encontra. No QUADRO 18 é demonstrado um pedaço do código que calcula a distância entre o sensor e a água.

```
String readUltrasonicSensor(){
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);

    // O sensor calcula o tempo gasto entre o envio e o recebimento
    // do sinal e retorna um pulso com esta duração
    long duration = pulseIn(echoPin, HIGH);

    // Converte o tempo para distância em centímetros
    float cm = microsecondsToCentimeters(duration);
    char leitura[100];
    dtostrf(cm,2,2,leitura);

    return leitura;
}
```

QUADRO 18: Código Arduino de Leitura do Sensor Ultrassônico

O sensor de chuva ficará posicionado em algum lugar livre para que a chuva caia sobre o mesmo. Ele capta um valor entre 0 e 1023, onde definimos valores para a intensidade da chuva: valores entre 900 e 1023 indicam intensidade 0 (chuva nula), entre 450 e 900

indicam intensidade 1 (chuva moderada) e entre 0 e 450 indicam intensidade 2 (chuva intensa). No QUADRO 19 é demonstrado um pedaço do código do sensor de chuva.

```
String leituraChuva (){
    int valorLeitura = analogRead(pino_d);
    int valorRetorno;
    if(valorLeitura>900 && valorLeitura<1023){
        valorRetorno = 0;//chuva nula
    } else if (valorLeitura>450 && valorLeitura<900){
        valorRetorno = 1; //chuva moderada
    } else if (valorLeitura>0 && valorLeitura<450){
        valorRetorno = 2;//chuva intensa
    }
    return String(valorRetorno);
}
```

QUADRO 19: Código Arduino de Leitura do Sensor de Chuva

A comunicação com o serviço que efetua a leitura é feita através de *socket* de rede com o *Ethernet Shield* plugado ao Arduino Uno, possibilitando que seja hospedado um server socket dentro do Arduino. Ele ficará esperando uma solicitação do serviço que efetua a leitura e retornará um texto no formato JSON com as leituras atualizadas. No QUADRO 20 é demonstrado um pedaço do código do *server socket*.

```
EthernetClient client = server.available();
if (client) {
    if (client.available()) {
        client.println(JSONLeituras());
        piscaLed();
    }
}
```

QUADRO 20: Código Arduino de Server Socket

3.3.4 Serviço de Controle de Leituras

Para que o site ou aplicativo móvel possam exibir seus dados sobre nível do rio e estado de chuva corretamente é necessário um software que realize a comunicação e controle entre a régua automatizada e o servidor. Como os dados e serviços foram construídos com o objetivo de serem hospedados num servidor na nuvem com sistema operacional Linux, optou-se pelo desenvolvimento de um *daemon* (a partir de agora chamado serviço) em Java, em conjunto ao *framework* Java Service Wrapper como solução para problema descrito acima.

O funcionamento resumido desse serviço é realizar a leitura dos sensores da régua automatizada, armazenar a informação e se necessário gerar uma nova imagem de inundação. A leitura da régua é feita pela internet via conexão *Socket*, que a cada intervalo pré-determinado de tempo é aberta para captura de dados, que é retornada num formato padrão JSON. A codificação da conexão pode ser vista no QUADRO 21 a seguir.

```
public Leitura efetuarLeitura() throws IOException {
    Propriedades prop = new Propriedades();
    Socket socketRegua = null;
    Leitura leitura = null;
    try {
        socketRegua = new Socket(prop.getProp("reguaHost"),
        Integer.parseInt(prop.getProp("reguaPorta")));
        BufferedReader in;
        DataOutputStream out;
        //Envia dados para iniciar leitura
        out = new DataOutputStream(socketRegua.getOutputStream());
        out.writeBytes(".");
        //Recebe leitura
        in = new BufferedReader(new
InputStreamReader(socketRegua.getInputStream()));
        String sLeitura = in.readLine();
        JSONObject jsonLeitura = new JSONObject(sLeitura);
        leitura = new Leitura();
        leitura.setDataHora(new Date());
        leitura.setNivelChuva(jsonLeitura.getInt("nivelChuva"));
        leitura.setNivelRio(jsonLeitura.getDouble("nivelRio"));
    } finally {
        socketRegua.close();
    }
    return leitura;
}
```

QUADRO 21: Conexão de leitura com a régua automatizada

As informações coletadas são armazenas no banco de dados não relacional MongoDB, pois possui grande velocidade de leitura e gravação e o relacionamento entre tabelas não é necessário. Se o serviço identificar que houve alteração no nível do rio maior que a esperada, inicia-se a geração de uma nova imagem de inundação (QUADRO 22) transparente no formato PNG, que é utilizada como sobreposição no mapa pela API Google Maps. A geração

é realizada utilizando a biblioteca ImageIO do JavaFX, onde são analisados os pontos geográficos presentes no banco de dados e, se ele estiver abaixo do nível atual do rio, será marcado de azul, simulando a presença de água.

```
public void gerarImagemMongoDB(Leitura leitura) throws UnknownHostException {
    int x, y = 0;

    //Tamanho dos blocos a serem preenchidos
    int tamBloco = 1;

    [...]

    //Prepara imagem de saida
    BufferedImage image = new BufferedImage(tamBloco * X, tamBloco * Y,
        BufferedImage.TYPE_INT_ARGB);
    Graphics g = image.getGraphics();

    //Varre o array de elevacoes preenchendo o mapa
    cont = 0;
    for(int i =0; i < Y; i++){
        for(int j =0; j < X; j++){
            x = i * tamBloco;
            y = j * tamBloco;

            //Define cor do bloco
            if (elevacoes[cont] > leitura.getNivelRio()) {
                g.setColor(new Color(0, 0, 0, 0));
            } else {
                g.setColor(new Color(0, 0, 1, opacidadeBloco));
            }

            //Pinta bloco
            g.fillRect(y, x, tamBloco, tamBloco);

            cont++;
        }
    }
    g.dispose();
    salvarImagem(image);
}
```

QUADRO 22: Geração de imagem de inundação

Todas as variáveis de configurações podem ser alteradas num arquivo de texto na pasta de usuário, que é salvo no formato .cfg. A estrutura básica do arquivo pode ser vista no QUADRO 23. Para realizar sua leitura é utilizada a biblioteca Properties, do pacote Utils do Java.

```
#Sun Aug 31 17:11:49 BRT 2014
reguaHost=201.67.130.36
reguaPorta=5661
mongoHost=localhost
mongoPorta=27017
mongoDB=mydb
dirImg=/home/roberto/teste.png
diffGerador=5
```

QUADRO 23: Estrutura do arquivo de configurações do serviço

Para transformar o arquivo .jar executável do java num *daemon* foi utilizado o Java Service Wrapper, onde é necessário alterar alguns arquivos de configuração e posicionar o executável num diretório específico. É possível ver a hierarquia de diretórios e arquivos do serviço no QUADRO 24. O *framework* possui arquivos de *batch* que realizam a instalação e controle do serviço. Como ele rodará em *background*, nenhuma interação com o usuário é possível e o software deve possuir um aprimorado tratamento de erros.

```
.
├── bin
│   ├── aplicativo-daemon
│   ├── aplicativo.jar
│   ├── lib
│   │   └── libwrapper.so
│   ├── wrapper
│   └── wrapper.jar
└── conf
    └── wrapper.conf
    └── sobre.txt
    └── start-service.sh
    └── stop-service.sh
```

QUADRO 24: Hierarquia de diretórios do Java Service Wrapper

3.3.5 Banco de Dados

Como pode ser visto nos requisitos funcionais, o sistema não armazenará dados complexos, porém mesmo que básicos, necessitam ser acessados e gravados com muita velocidade e em grande quantidade. Para suprir tal necessidade optou-se por um banco de dados não relacional, o MongoDB, que apresenta estrutura simples e cumpre o objetivo exposto.

As partes do protótipo que realizam comunicação com o banco de dados são o serviço de controle de leituras, WebService – ambos em Java –, aplicativo e site – através do PHP –.

O MongoDB possui bibliotecas preparadas para ambas linguagens, bem documentadas e completas. No QUADRO 25 é possível analisar um exemplo de operação de consulta ao banco em Java e no QUADRO 26 em PHP.

```
public void setNovaLeitura(Leitura leitura) throws UnknownHostException {
    // Conecta ao banco de dados
    MongoClient mongoClient = new MongoClient(prop.getProp("mongoHost"),
        Integer.parseInt(prop.getProp("mongoPorta")));

    DB db = mongoClient.getDB(prop.getProp("mongoDB"));
    DBCollection coll = db.getCollection("leituras");

    //Cria objeto para gravação
    BasicDBObject obj = new BasicDBObject("dataHora", leitura.getDataHora()).
        append("nivelRio", leitura.getNivelRio()).
        append("nivelChuva", leitura.getNivelChuva());

    //Grava objeto
    WriteResult wResult = coll.insert(obj);

    //Verifica resultado da gravação
    if (wResult.getField("ok").toString().equals("1.0")) {
        System.out.println("Nova leitura gravada com sucesso!");
    } else {
        throw new RuntimeException(
            "Erro ao gravar nova leitura no banco de dados.");
    }
}
```

QUADRO 25: Consulta no MongoDB por Java

```
function getLeituras($qtdLeituras, $leiturasValidas = true) {
    #Conecta ao MongoDB
    $m = new MongoClient();
    $db = $m -> mydb;
    $collectionLeituras = $db -> leituras;

    #Busca ultimas leituras
    if ($leiturasValidas) {
        $query = array('nivelRio' => array('$ne' => 'null'));
        $cursor = $collectionLeituras -> find($query);
    } else {
        $cursor = $collectionLeituras -> find();
    }
    $cursor -> sort(array('dataHora' => -1));
    $cursor -> limit($qtdLeituras);

    #Insere leituras no array
    $leituras = array();
    $i = 0;
    foreach ($cursor as $document) {
        $Hora = date(DATE_ISO8601, $document["dataHora"] -> sec);
```

```

    $leituras[$i][0] = date("d/m/Y", strtotime($Hora)) . " " . date("h:i:sa",
strtotime($Hora));
    $leituras[$i][1] = $document["nivelRio"];

    [...]
}

return $leituras;
}

```

QUADRO 26: Consulta no MongoDB por PHP

3.3.6 Web Service

Será disponibilizado um Web Service para que os usuários que tenham interesse em acessar a base de dados, tenham acesso às leituras realizadas do nível de rio e intensidade de chuva. Os usuários devem acessar o método leituras(), informando a quantidade de leituras que devem ser retornadas e a data e hora no formato ISODate em que as leituras foram feitas. O usuário terá acesso ao Web Service através de um link disponibilizado no site, que o encaminhará para a página WSDL utilizada na invocação do método do Web Service. A seguir no QUADRO 27, é exibido um exemplo de uma página WSDL.

```

<xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://ws.apache.org/axis2">
<xs:elementname="leituras">
<xs:complexType>
<xs:sequence>
<xs:elementminOccurs="0" name="quantidadeRegistros" type="xs:int"/>
<xs:element minOccurs="0" name="dataHora" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:elementname="leiturasResponse">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="return" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

QUADRO 27: Pagina WSDL com Metodo leituras()

O Web Service foi desenvolvido em Java, utilizando o *framework* Axis2 e as bibliotecas Java JSON, para manipular arquivos no formato JSON e Java MongoDB Driver

para acessar o banco de dados MongoDB. O algoritmo desenvolvido, como exibido no QUADRO 28, será hospedado junto ao Axis2 em um servidor de aplicação Apache TomCat.

```
public String leituras(int quantidadeRegistros, String dataHora) {
    [...]

    mongoClient = new MongoClient("localhost", 27017);
    DB database = mongoClient.getDB("mydb");
    DBCollection collection = database.getCollection("leituras");

    Date data = dateFormater.parse(dataHora);
    BasicDBObject range = new BasicDBObject("dataHora",
        new BasicDBObject("$lte", data));

   DBObject ordem = new BasicDBObject();
    ordem.put("dataHora", -1);

    DBCursor cursor = collection.find(range).sort(ordem).limit(quantidadeRegistros);

    [...]
    while (cursor.hasNext()) {
       DBObject tempObj = cursor.next();
        if ((!tempObj.get("nivelRio").equals("null")) ||
            (tempObj.get("nivelChuva").equals("null"))) {
            arrayRio.put(tempObj.get("nivelRio"));
            arrayChuva.put(tempObj.get("nivelChuva"));

        arrayDataHora.put(dateFormater.format(tempObj.get("dataHora")));
    }
}

jsonFinal.put("dataHora", arrayDataHora);
jsonFinal.put("nivelChuva", arrayChuva);
jsonFinal.put("nivelRio", arrayRio);

[...]

return jsonFinal.toString();
}
```

Quadro 28: Exemplo de Código Web Service

O cliente para o Web Service pode ser desenvolvido em qualquer linguagem que suporte o protocolo SOAP. O algoritmo no quadro a seguir (QUADRO 29) é um exemplo de cliente desenvolvido na linguagem PHP.

```
<?php
$client = new SoapClient('http://localhost:8080/axis2/services/WebServiceLeituras?
wsdl');

$function = 'leituras';

$args = array('ConvertTemp' => array(
```

```

        'quantidadeRegistros' => 15,
        'dataHora'      => '2014-09-05T22:16:23.862Z'
    ));

$result = $client->__soapCall($function, $arguments);
?>

```

QUADRO 29: Implementação de um cliente WebService em PHP

3.3.7 Site

A interação entre os usuários e os dados das enchentes armazenados no banco de dados, dá-se através de um site de Internet. O mesmo será desenvolvido utilizando-se as linguagens de marcação HTML5, CSS3, JQUERY(JavaScript) e a linguagem de programação PHP. Também será projetado de forma a melhor se aptar aos navegadores Google Chrome e Mozilla Firefox.

A fim de trazer as informações da melhor forma possível e que os usuários tenham uma ótima experiência de uso, o site terá uma aparência semelhante à de um programa de computador, sem rolagens e com um menu fixado na barra superior.

O sistema contará com o uso do *framework* Bootstrap, que permite o uso de um design pronto, padronizado e bonito para componentes visuais como botões, menus e tabelas, gerando assim uma aparência mais limpa e agradável para o usuário e mais simples de construir para o desenvolvedor, como visto na IMAGEM 27.

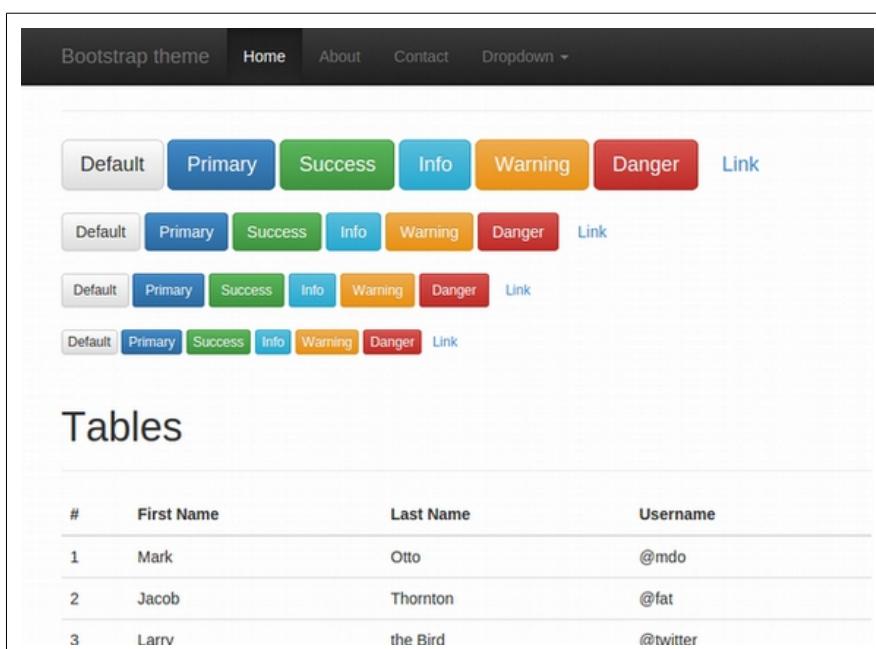


IMAGEM 27: Design padronizado do Bootstrap

Esse *framework* permite através do uso de seu sistema de grade, que os elementos do site sejam responsivos, ou seja, se ajustem nos mais diversos tamanhos de telas, sejam de computadores, *tablets*, *smartphones* ou *notebooks*.

Esse sistema de grades divide a tela do navegador em linhas que possuem até 12 colunas. Para isso utilizam-se conjuntos de *tags* <div> que criam uma divisão no documento e herdam classes como *.row*, usadas para definir linhas e classes *.col*, para definir colunas. Um exemplo pode ser visto no QUADRO 30.

```
<div class="row">          <!--Adiciona uma linha-->
    <div class="col-md-8">.col-md-8</div>      <!--2 divisões, uma com 8 colunas-->
    <div class="col-md-4">.col-md-4</div>      <!--e uma com 4-->
</div>
<div class="row">          <!--Adiciona uma linha-->
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4">.col-md-4</div>      <!--3 divisões com 4 colunas cada-->
    <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">          <!--Adiciona uma linha-->
    <div class="col-md-6">.col-md-6</div>      <!--2 divisões com 6 colunas cada-->
    <div class="col-md-6">.col-md-6</div>
</div>
```

QUADRO 30: Exemplo de uso do sistema de grades do Bootstrap.

Outra opção que o Bootstrap possibilita é o uso da classe *.visible*, que permite ocultar ou exibir elementos em determinadas resoluções de telas. Nesse contexto, em telas menores como em *smartphones*, algumas funções do site podem ser escondidas, levando o usuário a utilizar um aplicativo instalado para ter total acesso as funcionalidades, e não acessar o sistema diretamente pelo navegador.

Como conteúdo, o site terá um mapa das inundações em tempo real, um simulador de inundações, tabelas e gráficos do nível do rio e estado da chuva, galeria de imagens integrada com o Facebook, acesso facilitado a outros sites relacionados, e históricos de enchentes.

3.3.8 Aplicativo Móvel

O aplicativo móvel será desenvolvido, assim como o site, utilizando HTML5, CSS3, JQuery e PHP. A diferença está no uso o *framework* Phonegap, pois ele permite a integração de um mesmo aplicativo com as mais variadas plataformas móveis, tornando possível o aplicativo rodar em Android, IOs, Firefox OS ou Windows Phone.

O Phonegap possui métodos JavaScript que fazem a ponte entre a página HTML(site) e os componentes do dispositivo móvel. Todas as funções que fazem uso de recursos dos *smartphones* ou *tablets* como câmera, geolocalização, vibração, entre outras, são obtidas através do uso de *plug-ins* do Phonegap instalados separadamente. Seu uso é simplificado, através de alguns comandos, como: *phonegap create nomeProjeto*, que cria uma pasta com o projeto Phonegap dentro; *phonegap install plataforma*, que adiciona uma das plataformas disponíveis como Android, IOS, Windows Phone; *phonegap build plataforma*, que converte a página HTML no aplicativo da plataforma selecionada; e, por fim, *phonegap run plataforma*, que executa o aplicativo, da plataforma selecionada, diretamente em um *smartphone* conectado ao computador, ou em um emulador como o Genymotion.

O aplicativo usará como interface gráfica o *framework* Ionic, muito semelhante ao *Bootstrap*, que permite que o site do aplicativo seja responsivo e disponibiliza componentes visuais prontos. Um exemplo de sua utilização pode ser visto na IMAGEM 28.

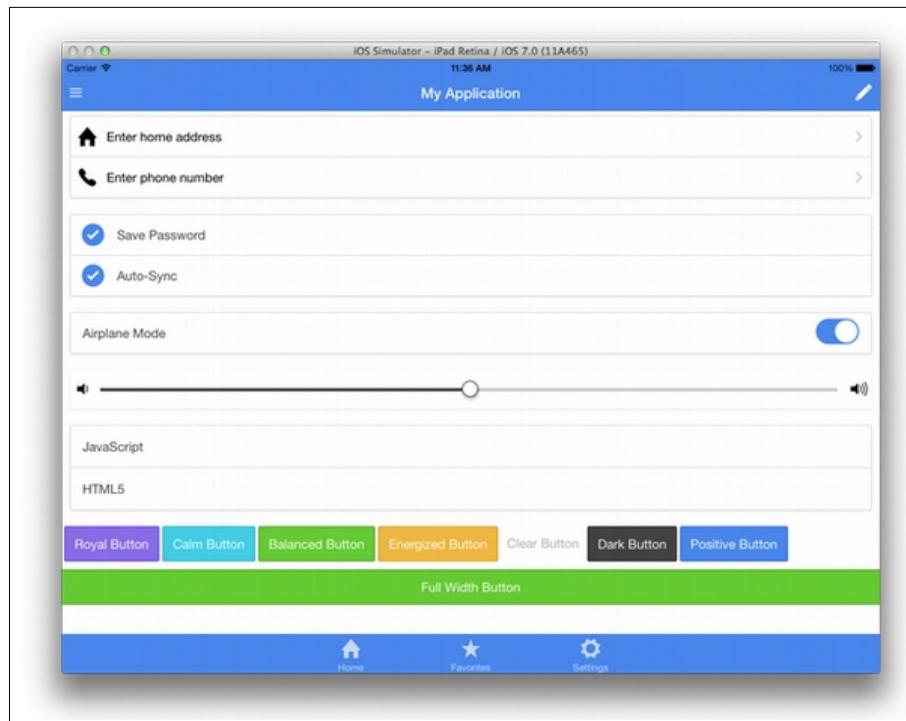


IMAGEM 28: Design do *framework* Ionic

3.3.9 Mapa de Inundações

O mapa de enchentes e inundações fará parte da tela inicial do aplicativo e do site. Ele é exibido sobre a página toda, abaixo dos menus superiores, e quando o site ou o

aplicativo são abertos o mapa será posicionado no centro da cidade. Assim, é possível saber o estado da enchente ou inundação no momento em que é aberto o site ou aplicativo.

Para introduzir o mapa na página, são utilizados métodos da API do Google Maps. A utilização da API baseia-se em: adiciona-se um `<div id='mapa'>` e um código JavaScript que exibe o mapa. Nesse código é inicializado o mapa através do uso de duas variáveis, `mapOption` que define os parâmetros do mapa, como coordenadas iniciais tipo, zoom, entre outras; e `google.maps.Map()`, que transforma o `<div>`, aplicando as configurações definidas em `mapOption`. Um exemplo simples pode ser observado no QUADRO 31.

```
<script type="text/javascript">
function initialize() {
    var mapOptions = {
        center: new google.maps.LatLng(-34.397, 150.644), //coordenadas
centro do mapa
        zoom: 8,
        mapTypeId: google.maps.MapTypeId.ROADMAP //tipo de mapa
    };
    var map = new google.maps.Map(document.getElementById("mapa"),
        mapOptions);
}
</script>
<div id="mapa"></div>
```

QUADRO 31: Exemplo HTML da utilização da API Google Maps.

Nesse mapa são exibidos, em tempo real, o estado de inundação no local. Para isso é inserida uma imagem azul com transparência, como visto na IMAGEM 29. Isso é possível através do uso do método *google.maps.GroundOverlay()* da API, que permite sobrepor uma imagem no formato PNG sobre o mapa, fixando-a em determinada posição geográfica.

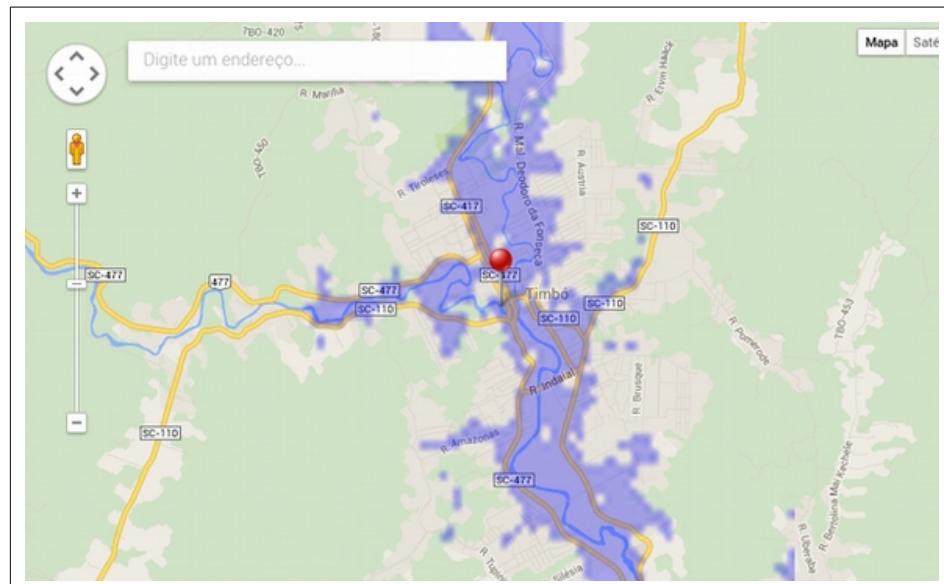


IMAGEM 29: Mapa com a imagem Azul simulando a inundação.

Nesse mapa ainda há uma barra de pesquisa de endereços, para que os usuários possam facilmente encontrar locais. Essa barra é posicionada no mapa através do método `map.controls[parametro].push(elementoHtml)`, que recebe como parâmetro, entre as chaves, a posição em que será posicionado o elemento, sendo usado, por exemplo, o formato `google.maps.ControlPosition.TOP_LEFT`. Nos parênteses são informados a referência do elemento HTML a ser posicionado no mapa.

A variável `google.maps.places.SearchBox()` transforma o elemento HTML na barra de pesquisa funcional. O método `google.maps.event.addListener(searchBox, funçãoTratamento)` permite fazer todo o tratamento com os dados do local pesquisado, como posicionar o mapa.

3.3.9.1 Simulação de Inundações

Ainda nessa tela é possível realizar uma simulação de inundação, que é ativada pelo menu “simular”, exibindo uma janela com controles para aumentar ou diminuir o nível do rio, com intervalo de meio metro. A simulação dá-se através da alteração entre varias imagens azuis que sobrepõem o mapa, estas previamente geradas e salvas no servidor. Quando o usuário aumenta ou diminui o nível, é chamado o método JavaScript `similar()`, que carrega novamente o mapa, passando como parâmetro a imagem cujo nome corresponde ao nível.

3.3.10 Níveis Atualizados e Alertas

O site e aplicativo móvel do sistema exibem as últimas leituras atualizadas e informações de possíveis situações de alerta. Para as leituras, o PHP simplesmente realiza um acesso ao banco de dados MongoDB. Para o sistema de alerta, existe uma regra que considera a última leitura válida e dados presentes no site da Defesa Civil de Santa Catarina.

O site da Defesa Civil exibe informações de alerta e precaução para diversas cidades. Através da biblioteca `PHP Simple HTML DOM Parser` (<http://simplehtmldom.sourceforge.net>), é carregado o código HTML da página e feito uma varredura em busca da área (`div`) que contém os avisos. Por um filtro de palavras, se for encontrado chaves como “Timbó”, “Blumenau” ou “Médio Vale”, o sistema considera que estamos em estado de alerta, e cria um hiperlink na mensagem de aviso que encaminha o

usuário à página da Defesa Civil. A seguir (QUADRO 32) é possível visualizar parte do algoritmo PHP que efetua essa verificação.

```
function verificarEstadoDefesaCivil() {
    try {
        #Palavras chave de pesquisa
        $palavrasChave = array('Timbó', 'Blumenau');

        #Busca quadr de informações no site da defesa civil
        include "lib/simple_html_dom.php";
        $html = file_get_html('http://www.defesacivil.sc.gov.br/');

        #Filtrar dados
        $html = $html -> getElementById("user5");
        $html = $html -> find("p");

        #Varre dados em busca de palavras chave
        $cont = 0;
        $achouSemAviso = false;
        $achouPalavra = false;
        foreach ($html as $key => $value) {
            if ($cont != 0) {

                #Procura por "Sem aviso."
                if (strpos($value,'Sem aviso.') !== false) {
                    $achouSemAviso = true;
                }

                #Procura por palavras Chave
                foreach ($palavrasChave as $key => $palavraChave) {
                    if (strpos($value, $palavraChave) !== false) {
                        $achouPalavra = true;
                    }
                }
            }
            $cont++;
        }

        #Se não achou "Sem aviso" e achou palavra chave
        if (!$achouSemAviso & $achouPalavra) {
            return true;
        } else {
            return false;
        }
    } catch (Exception $e) {
        #echo 'Cuaght exception: ', $e -> getMessage(), "\n";
        return false;
    }
}
```

QUADRO 32: Exemplo de algoritmo PHP para varredura do site da Defesa Civil de SC

A regra de alerta do sistema considera a seguinte situação: se o rio estiver em condições normais e não houver avisos no site da Defesa Civil de Santa Catarina, o estado é

NORMAL; se o rio estiver elevado – porém não transbordado – ou houver avisos no site da Defesa Civil, o estado é de ALERTA; e finalmente, se o rio estiver em inundação, o estado é de INUNDAÇÃO.

3.3.11 Histórico de Medições

Históricos de medições é uma área do sistema que mostra ao usuário os registros das últimas medições de nível do rio e estado da chuva. Todos os dados que o aplicativo necessita, armazenados no servidor por pelo MongoDB, são adquiridos através do uso de requisições AJAX (vide QUADRO 33), que o servidor interpreta com PHP retornando os dados no formato JSON. Esses dados são dispostos em uma tabela contendo todos os dados das últimas 24 horas e, ainda em dois gráficos, as alterações do nível do rio e estado da chuva.

```
function getEstadoAlerta(){
    $.ajax({
        url: "192.168.0.1/Projeto%20Web/Comum/php/funcoes.php?
getEstadoAlerta=?",
        dataType: 'jsonp',
        crossDomain: true,
        success: function(data){
            setEstado(data);
        }
    });
}
```

QUADRO 33 – Requisição AJAX que obtêm o estado de alerta no servidor

Os gráficos são gerados usando a API Google Charts, que disponibiliza uma grande variedade de opções. Para utilizá-la é necessário o uso de um código JavaScript como visto no QUADRO 34.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
    google.load("visualization", "1", {packages:["corechart"]});
    google.setOnLoadCallback(drawChart);
    function drawChart() {

        var data = google.visualization.arrayToDataTable([
            ['Year', 'Sales', 'Expenses'],
            ['2004', 1000, 400],
            ['2005', 1170, 460],
            ['2006', 660, 1120],
        ]);
    }
}</script>
```

```

        ['2007', 1030, 540]
    ]);

    var options = {
        title: 'Company Performance',
        hAxis: {title: 'Year', titleTextStyle: {color: 'red'}}
    };

    var chart = new google.visualization.ColumnChart(document.
getElementById('chart_div'));

    chart.draw(data, options);
}
</script>
<body>
    <div id="chart_div" style="width: 900px; height: 500px;"></div>
</body>

```

QUADRO 34: Exemplo HTML do uso do Google Charts.

Nesse código a variável *google.visualization.arrayToDataTable* recebe os dados que serão exibidos no gráfico. A variável *options* recebe as definições do gráfico, como título e cores. O *google.visualization.LineChart* define o gráfico em cima do *<div>* relacionado. Então o método *Draw* junta as informações e desenha o gráfico na tela.

3.3.12 Previsão do Tempo

A seção Previsão do Tempo possui um mapa com informações meteorológicas. Este não possui a imagem azul sobreposta, é utilizado apenas o serviço Weather Layer para exibir informações meteorológicas em tempo real como visto na IMAGEM 30. Isso é possível definindo no mapa a variável *google.maps.weather.WeatherLayer*, que exibe dados de temperatura atual e condições de tempo como chuvoso ou ensolarado, e *google.maps.weather.CloudLayer*, que exibe a imagem de satélite das nuvens em tempo real.



IMAGEM 30: Mapa da Google com Weather Layer

Junto a esse mapa é incorporado um quadro, fornecido pelo CPTEC (tempo.cptec.inpe.br), com a previsão do tempo da cidade. O quadro pode ser visto na IMAGEM 31.



IMAGEM 31: Previsão do tempo incorporada fornecida por CPTEC

A tela ainda conta com uma imagem GIF encontrada no site do Clima Tempo (<http://www.climatempo.com.br>) que exibe uma imagem de satélite com os avanços do tempo, como visto na IMAGEM 32.

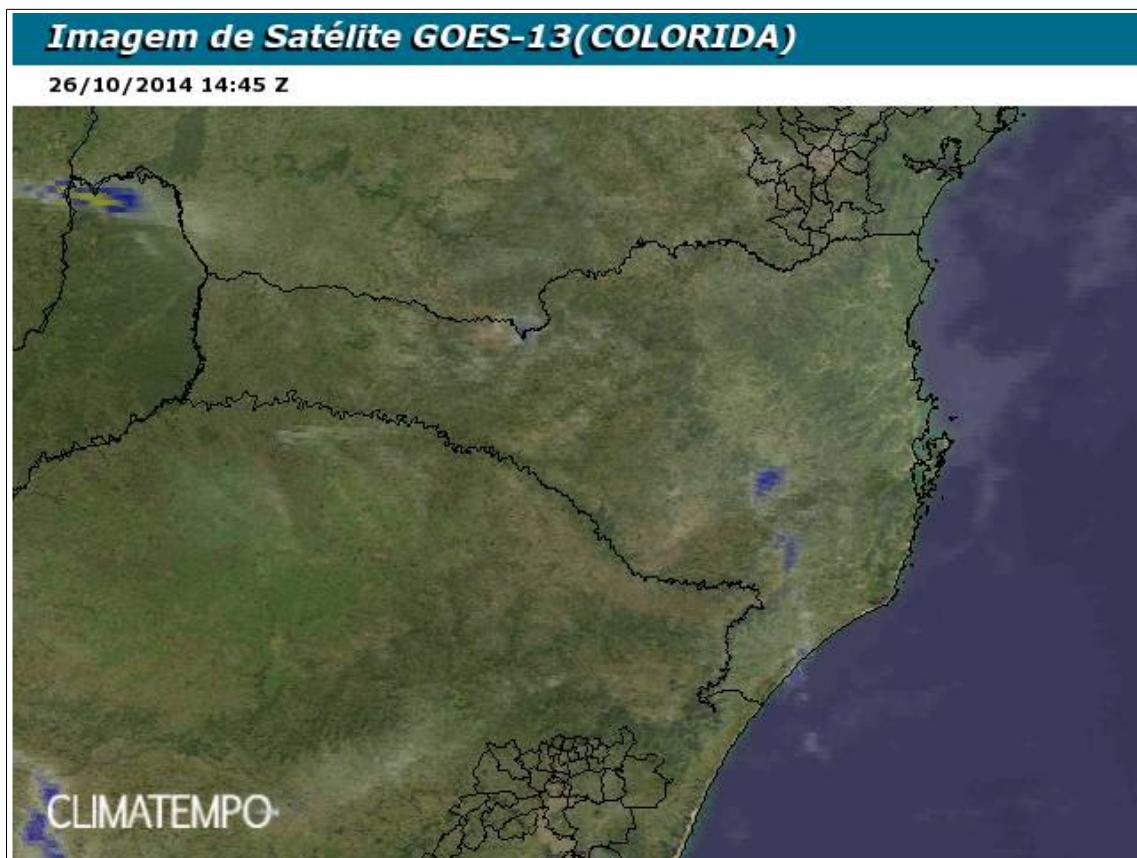


IMAGEM 32: Imagem de satélite d e Santa Catarina

3.3.13 Galeria Colaborativa

O sistema disponibiliza no site de versão *desktop* e aplicativos móveis instalados uma galeria colaborativa onde os usuários podem enviar, pesquisar e visualizar fotos de enchentes e inundações. A galeria é composta em sua maioria por imagens enviadas pelos usuários, e por isso, é utilizado um sistema de login da API da rede social Facebook.

Para enviar uma foto, o usuário necessita obrigatoriamente realizar a autenticação com uma conta do Facebook. Ao postar uma imagem e preencher algumas descrições básicas, a imagem é enviada ao site e automaticamente publicada no mural do usuário na rede social. Isso gera maior segurança nos conteúdos enviados, desencorajando a publicação de imagens fora do tema.

Para a exibição da galeria é utilizado um sistema de grades com o *framework* de desenvolvimento WEB BootStrap. Ao enviar uma imagem ela é carregada para o perfil do usuário no Facebook e ao servidor que hospeda o site, pois caso seja removida da rede social, continuará disponível no sistema.

3.3.14 Verificador de Vulnerabilidade de Locais

A verificação de vulnerabilidade de locais é uma área do sistema, apenas disponível para aplicativo móvel, onde o usuário terá acesso à incidência de inundações do local atual, através do sistema global de posicionamento (GPS).

O verificador de vulnerabilidade de locais foi desenvolvido utilizando o *plug-in* Geolocation do PhoneGap e os serviços Geocoding e Elevation da API Google Maps.

O *plug-in* Geolocation é utilizado para capturar as coordenadas de latitude e longitude, através do sistema GPS. A seguir é exposto no QUADRO 35 um exemplo do código em JavaScript da utilização do *plug-in*.

```
function coordenadas() {
    var onSuccess = function(position) {
        latitude = position.coords.latitude
        longitude = position.coords.longitude
    };
    function onError(error) {
```

```

        alert ('code: ' + error.code + '\n' + 'message: ' + error.message + '\n');
    }

var options = {
    maximumAge : 3,
    timeout : 60000,
    enableHighAccuracy : true
};

navigator.geolocation.watchPosition(onSuccess, onError, options);

}

```

QUADRO 35: Código JavaScript do *plug-in* Geolocation

O serviço Geocoding da API Google Maps é utilizado em uma função de geocodificação reversa, onde após passadas as coordenadas de latitude e longitude, a função retorna o endereço respectivo às coordenadas. No QUADRO 36, é demonstrado um pedaço do código em JavaScript da utilização do serviço.

```

function geocodificacaoReversa() {
    var geocoder;
    geocoder = new google.maps.Geocoder();
    var latlng = new google.maps.LatLng(latitude, longitude);
    geocoder.geocode({
        'latLng' : latlng
    }, function(results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
            if (results[0]) {
                alert(results[0].formatted_address);
            } else {
                alert("Geocoder failed due to: " + status);
            }
        }
    });
}

```

QUADRO 36: Exemplo de código JavaScript do serviço Geocoding

O serviço Elevation é utilizado em uma função que através das coordenadas de latitude e longitude, retorna a altura do local especificado pelas coordenadas. No QUADRO 37 é demonstrado um exemplo do código JavaScript da utilização do serviço.

```

function getAltura() {
    locations = [];

    elevator = new google.maps.ElevationService();
    var latlng = new google.maps.LatLng(latitude, longitude);

```

```

locations.push(latlng);
var positionalRequest = {
    'locations' : locations
};

elevator.getElevationForLocations(positionalRequest, function(results, status) {
    if (status == google.maps.ElevationStatus.OK) {
        if (results[0]) {
            alert('altura = ' + (results[0].elevation));
        } else {
            alert('No results found');
        }
    } else {
        alert('Elevation service failed due to: ' + status);
    }
});
}

```

QUADRO 37: Exemplo de código JavaScript do serviço Elevation

Após coletar todos os dados, o sistema buscará no banco de dados informações sobre enchentes anteriores e exibirá informações sobre o local pesquisado, como dados geográficos e históricos de inundações.

3.3.15 Sistema de Alerta

O Sistema de Alerta é uma funcionalidade presente apenas no aplicativo móvel. Seu objetivo é permitir aos usuários receberem alertas em casos de enchentes, a qualquer momento em seus dispositivos móveis, através da utilização de serviços de segundo plano, que ficam constantemente em execução.

Os usuários podem pesquisar no mapa ou obter os locais pelo GPS. Esses locais são armazenados no celular junto a sua elevação, obtida a partir do serviço Elevation da API Google Maps. Esse armazenamento é feito através do uso do *plug-in org.apache.cordova.file*, que possui métodos que permitem criar e armazenar arquivos no diretório de instalação do dispositivo móvel. Um exemplo da utilização desse algoritmo pode ser visto no QUADRO 38.

```

//evento carregado quando o dispositivo estiver pronto
function onDeviceReady() {
    window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, gotFS, falha);
}

function gotFS(fileSystem) {
    fileSystem.root.getFile("readme.txt", {create: true, exclusive: false},

```

```

gotFileEntry, falha);
}

function gotFileEntry(fileEntry) {
    fileEntry.createWriter(gotFileWriter, falha);
}

function gotFileWriter(writer) {
    writer.onwriteend = function(evt) {
        writer.truncate(11);
        writer.onwriteend = function(evt) {
            writer.seek(4);
            writer.write(" texto diferente");
        };
    };
    writer.write("texto texto");
}

//exibe um log de erros
function falha(error) {
    console.log(error.code);
}

```

QUADRO 38: Métodos de controle de arquivos com PhoneGap

A partir disto, quando ativado por um controle de usuário, é chamado um método JavaScript que a cada quinze minutos compara a elevação dos locais com o nível do rio. Se o mesmo estiver a menos de dois metros de diferença, é enviado ao usuário uma notificação, gerada pelo *plug-in org.apache.cordova.notifications* do Phonegap. O código do QUADRO 39 representa seu funcionamento.

```

/**
 * Ativa ou desativa os alertas
 */
function ativarAlerta(ativo){
    if(ativo){ //se ativado o alerta
        //chama alerta a cada 15min
        timer = setInterval(function(){alerta()}, 1000*5*15);

        //referencia para o estado on/off do botão de ativação
        alertaAtivo = true;
    }
    else{
        //destroi o alerta
        clearInterval(timer);
        alertaAtivo = false;
    }
}

// chamado quando clicado no botão "Abrir App" do alerta
function alertaAbrirPrograma() {

```

```

        application.show; exibe o programa
    }

//parametros do alerta
function alerta() {
    navigator.notification.confirm(
        'Atenção! Água a caminho!',
        alertaAbrirPrograma,
        'SAEmóvel Alerta',
        ['Abrir App','Ignorar']
    );
}

```

QUADRO 39: Código JavaScript do funcionamento do alerta

3.3.16 Servidor em Nuvem

Conforme exibido nos requisitos deste trabalho, várias partes do sistema – como servidor HTTP, *WebService* e serviço de controle de leituras (*Daemon*) – necessitam rodar sobre um servidor com conexão interrupta à internet. Como solução para o problema, concluiu-se que a melhor opção na data de elaboração desse projeto são os serviços de computação em nuvem da Amazon, Amazon Web Services.

O serviço *Amazon Elastic Compter 2* (EC2) fornece um ambiente virtualizado disponível vinte e quatro horas. É possível escolher o sistema operacional do servidor a ser utilizado, optando entre múltiplas versões de Windows Server e distribuições Linux, como Ubuntu Server, Amazon Linux, RedHat e Oracle Linux.

Para a execução de nossos aplicativos foi escolhida a distribuição Linux Ubuntu Server, por sua popularidade, documentação e estabilidade. Todas operações são realizadas num ambiente *shell*, via conexão SSH. O comando de conexão pode ser visualizado no QUADRO 40, e a IMAGEM 33 exibe o ambiente em operação.

```
ssh -i ~/<nome da chave>.pem ubuntu@<ip do servidor>
```

QUADRO 40: Comando de conexão à máquina virtual na Amazon Web Services

Especificamente, serão executados nesse ambiente o servidor de HTTP e PHP (Apache Web Server), serviço de WebService (Apache AXIS 2), contêiner de aplicações WEB Java (Apache TOMCAT 8), serviço de controle de leituras (*Daemon*, em Java) e banco de dados não relacional MongoDB.

```
ubuntu@ip-172-31-10-21: ~
roberto@robertodebarba-Aspire-5552:~$ !112
ssh -i ~/Roberto.pem ubuntu@54.232.207.63
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/

 System information as of Tue Oct 14 22:44:39 UTC 2014

 System load:  0.0          Processes:           82
 Usage of /:   38.5% of 7.75GB  Users logged in:    0
 Memory usage: 29%           IP address for eth0: 172.31.10.21
 Swap usage:   0%

 Graph this data and manage this system at:
   https://landscape.canonical.com/

 Get cloud support with Ubuntu Advantage Cloud Guest:
   http://www.ubuntu.com/business/services/cloud

82 packages can be updated.
41 updates are security updates.

*** /dev/xvda1 should be checked for errors ***

Last login: Tue Oct 14 22:44:40 2014 from 187.5.137.153
ubuntu@ip-172-31-10-21:~$ █
```

IMAGEM 33: Ambiente de operação de uma máquina na Amazon Web Services

4 CONCLUSÕES

O produto final adquirido com este trabalho mostra-se de grande utilidade para municípios ou regiões afetadas por enchentes e inundações. Seu desenvolvimento teve como frutos a obtenção de uma grande quantidade de informações, resultando em conhecimento útil a seus autores e possíveis leitores, tanto no quesito tecnológico quanto socioambiental.

As tecnologias empregadas mostram-se, em maioria, inovadoras para o mercado e em determinados casos ainda pouco estudadas. Sua utilização promove uma forma de conhecimento e incentivo a inovações, como a implantação de sistemas de medição de rio automatizados, que hoje ainda são realizados de forma arcaica na região do Médio Vale do Itajaí.

As dificuldades encontradas na implementação de um sistema com tais recursos tornou o trabalho desafiador, porém aumentou o incentivo pelo alcance de seus objetivos, que também foi impulsionado pelas opiniões das pessoas consultadas, que desde o levantamento de requisitos até a apresentação do produto final, se mostraram muito positivas.

A possibilidade de prever, acompanhar e alertar a população sobre possíveis enchentes, de forma automática e de fácil acesso, despertou grande interesse e, considerando o baixo custo de implantação, possui grandes chances de chegar ao mercado. Seus benefícios são claros e partem da economia de recursos financeiros à tranquilização das pessoas, traumatizadas pelos eventos ocorridos recentemente.

Como limitação principal encontrada durante o desenvolvimento deste trabalho, está a aplicabilidade do sistema em cidades com geografia e desenvolvimento adversos. Áreas com grandes variações de elevação de terreno, onde o rio possui grandes quedas, não possibilitam a implantação do produto, já que a análise de área ocorre num ponto específico e a inundação de locais varia entre os diversos níveis do rio. Outro ponto que pode dificultar sua implantação é o desenvolvimento das estruturas de rede de dados da região, considerando-se que o sistema necessita nos pontos de medição conexão interrupta à internet. Para contornar tais problemas, a utilização de vários pontos de medição e a substituição de conexões de Internet por sistemas via rádio mostram-se viáveis.

4.1 EXTENSÕES

- Implantar sistemas de alerta via mensagens de texto SMS;
- Substituir a conexão dos pontos de medição automatizados por sinais via rádio;
- Implantar uma ferramente de disparo de notificações editáveis;
- Desenvolver um sistema de ponto de medição com hardware específico, de menor custo e maior durabilidade;
- Desenvolver um sistema de navegação GPS que evita caminhos alagados.

REFERÊNCIAS BIBLIOGRÁFICAS

ABINADER, Jorge Abílio; LINS, Rafael Dueire. Conceitos Fundamentais em Web Services. In: ABINADER, Jorge Abílio; LINS, Rafael Dueire. **Web Services em Java**. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2006. p. 10-11.

ADOBE SYSTEMS INC.. **PhoneGap Documentation**. Disponível em: <<http://docs.phonegap.com/en/3.5.0/index.html>>. Acesso em: 07 ago. 2014.

ALECRIM, Emerson. **O que é cloud computing?** 2013. Disponível em: <<http://www.infowester.com/cloudcomputing.php>>. Acesso em: 14 set. 2014.

AMARO, Gustavo. **Criando um Web Service em Java utilizando Apache Axis2 e Eclipse**. 2012. Disponível em: <<http://spigandoeaprendendo.wordpress.com/2012/12/19/criando-um-web-service-em-java-utilizando-apache-axis2-e-eclipse/>>. Acesso em: 18 out. 2014.

AMAZON INC. (Org.). **Amazon Web Services**: Produtos e Serviços. Disponível em: <<http://aws.amazon.com/pt/products/>>. Acesso em: 14 set. 2014.

ARDUINO (Org.). **Arduino Uno: Overview**. Disponível em: <<http://arduino.cc/en/Main/ArduinoBoardUno>>. Acesso em: 06 set. 2014.

ARDUINO (Org.). **DHCP Chat Server**. Disponível em: <<http://arduino.cc/en/Tutorial/DhcpChatServer>>. Acesso em: 12 jul. 2014.

ARDUINO E CIA. **Ligando um sensor de chuva ao Arduino**. 2014. Disponível em: <<http://www.arduinoecia.com.br/2014/06/sensor-de-chuva-arduino.html>>. Acesso em: 18 out. 2014.

BARRETO, Maurício Vivas de Souza. **Curso de Linguagem PHP**. 2000. Departamento de Ciência da Computação e Estatística, Universidade Federal de Sergipe, Aracaju, 2000.

BECK, Kent. **Programação Extrema (XP) Explicada**. São Paulo: Artmed, 2004. 182 p.

BEGOLI, Edmon. **How to generate JPEG images from Java.** 2010. Disponível em: <<http://it.toolbox.com/blogs/lm/how-to-generate-jpeg-images-from-java-41449>>. Acesso em: 24 ago. 2014.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário.** 2. ed. Rio dos Janeiro: Elsevier Editora Ltda, 2006.

BRITO, Eduardo Ferreira. **Sistema operacional GNU/Linux:** um estudo sobre economia, estabilidade e segurança para tratamento das informações de microempresas. 2008. 46 f. Dissertação – Curso de Sistemas de Informação, Faculdade de Minas – Faminas-BH, Belo Horizonte, 2008. Disponível em: <http://www.ricardoterra.com.br/publications_files/students/2008_faminas_brito.pdf>. Acesso em: 14 set. 2014.

CAMPOS, Sylvia et al. **Introdução ao Eclipse.** Disponível em: <<http://www.cin.ufpe.br/~phmb/ip/MaterialDeEnsino/IntroducaoAoEclipse/IntroducaoAoEclipse.htm>>. Acesso em: 14 set. 2014.

CARNEIRO JUNIOR, Cloves; BARAZI, Rida Al. **Rails 3:** Básico. São Paulo: Novatec Editora, 2011. 400 p.

CEOPS – FURB. **Histórico de Enchentes:** Vale do Itajaí. 2010. Disponível em: <<http://ceops.furb.br/index.php/institucional/historico>>. Acesso em: 20 set. 2014.

CORDERO, Ademar. **Estudo da precipitação máxima diária para Blumenau-SC e o evento de novembro de 2008.** 2012. 8 f. Dissertação - Universidade Regional de Blumenau – FURB, Blumenau, 2012.

COSTA, Rodrigo Gonçalves Porto. **UNIVERSO JAVA:** Domine os recursos oferecidos por essa linguagem de programação. São Paulo: Universo dos Livros, 2008. 272 p.

DAZS, Ragen. **Entendendo um pouco sobre os daemons.** Disponível em: <<http://www.vivaolinux.com.br/artigo/Entendendo-um-pouco-sobre-os-daemons>>. Acesso em: 14 set. 2014.

DEFESA CIVIL DE SÃO BERNARDO DO CAMPO (São Paulo). **Enchente, Inundação, Alagamento ou Enxurrada?** 2011. Disponível em: <<http://dcsbcsp.blogspot.com.br/2011/06/enchente-inundacao-ou-alagamento.html>>. Acesso em: 20 set. 2014.

DOHMS, Rafael. **Google Maps API**: Um exemplo prático e comentado. Disponível em: <<http://blog.doh.ms/2006/12/06/google-maps-api-um-exemplo-pratico-e-comentado/>>. Acesso em: 23 jul. 2014.

EIS, Diego; FERREIRA, Elcio. **HTML5 e CSS3**: com farinha e pimenta. São Paulo: Tableless, 2012.

FAUSTINO, Maurício. **AJAX com JQuery**. Disponível em: <<http://www.mauriciofaustino.com/2009/03/ajax-com-jquery/>>. Acesso em: 25 ago. 2014.

FILIPEFLOP. **Como comunicar com o Arduino Ethernet Shield W5100**. 2014. Disponível em: <<http://blog.filipeflop.com/arduino/tutorial-ethernet-shield-w5100.html>>. Acesso em: 12 jul. 2014.

GOOGLE INC.. **API Javascript do Google Maps v3**. Disponível em: <<https://developers.google.com/maps/documentation/javascript/>>. Acesso em: 30 jul. 2014.

GOMES, Carlos Tiago N.. **Introdução a prototipação e apresentação do Axure RP 6.5** Leia mais em: Introdução a prototipação e apresentação do Axure RP 6.5. Disponível em: <<http://www.devmedia.com.br/introducao-a-prototipacao-e-apresentacao-do-axure-rp-6-5/27978>>. Acesso em: 26 set. 2014.

GOMES, Robson Fernando. **Utilizando arquivos de propriedades no Java**. Publicado por DevMedia. Disponível em: <<http://www.devmedia.com.br/utilizando-arquivos-de-propriedades-no-java/25546>>. Acesso em: 24 ago. 2014.

GONÇALVES, Antônio Ricardo. **O que é SaaS, IaaS e PaaS em Cloud Computing?: Conceitos Básicos**. 2013. Disponível em: <<http://antonioricardo.org/2013/03/28/o-que-e-saas-iaas-e-paas-em-cloud-computing-conceitos-basicos/>>. Acesso em: 14 set. 2014.

GONÇALVES, Eduardo Corrêa. **Introdução ao formato JSON.** Disponível em: <<http://www.devmmedia.com.br/introducao-ao-formato-json/25275#ixzz3CgMcBfk>>. Acesso em: 14 set. 2014.

GOOGLE. **Serviço de geocodificação.** 2013. Disponível em: <<https://developers.google.com/maps/documentation/javascript/geocoding?hl=pt-br#ReverseGeocoding>>. Acesso em: 18 out. 2014.

GOOGLE. **Serviço de Google Elevation API.** 2013. Disponível em: <<https://developers.google.com/maps/documentation/javascript/elevation?hl=pt-br>>. Acesso em: 18 out. 2014.

GROTZINGER, John; JORDAN, Tom. **Para Entender a Terra.** 6. ed. São Paulo: Bookman, 2013. 764 p.

HAMANN, Renan. **IOS, Android e Windows Phone:** números dos gigantes comparados. 2014. Disponível em: <<http://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm>>. Acesso em: 27 set. 2014.

HOPSON, K. C e INGRAM, Stephen E. **Desenvolvendo Applets com Java.** São Paulo. Editora Campus, 1997. p. 335 – 351.

IANNI, Vinicius. **Introdução aos bancos de dados NoSQL.** Disponível em: <<http://www.devmmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044#ixzz3Cjy88yuI>>. Acesso em: 14 set. 2014.

KOBIYAMA, Masato et al. **Prevenção de Desastres Naturais:** Conceitos Básicos. Florianopolis: Organic Trading, 2006. 122 p.

MARCELO, Antonio. **Apache:** Configurando o servidor web para Linux. 3. ed. Rio de Janeiro: Brasport, 2005. 111 p.

MEDEIROS, Higor. **Introdução ao MongoDB.** Disponível em: <<http://www.devmmedia.com.br/introducao-ao-mongodb/30792#ixzz3CfiBoc3v>>. Acesso em: 14 set. 2014.

MEYER, Laurid. **Simple Arduino TCP-Client using the Ethernetshield, DHCP and a Java Server.** 2012. Disponível em: <<http://www.lauridmeyer.com/2012/04/simple-arduino-tcp-client-using-the-ethernetshield-dhcp-and-a-java-server/>>. Acesso em: 12 jul. 2014.

MONGODB. **MongoDB:** Documentação API Java. Disponível em: <<https://api.mongodb.org/java/2.12/>>. Acesso em: 24 ago. 2014.

MONGODB. **The MongoDB 2.6 Manual.** Disponível em: <<http://docs.mongodb.org/manual/>>. Acesso em: 24 ago. 2014.

MOREIRA, Anderson. **Padrão de Codificação JAVA.** Disponível em: <http://siep.ifpe.edu.br/anderson/blog/?page_id=950>. Acesso em: 09 ago. 2014.

MOREIRA, Daniela. **Oracle compra Sun por US\$ 7,4 bilhões.** 2009. De INFO Online. Disponível em: <<http://info.abril.com.br/noticias/negocios/oracle-compra-sun-por-us-7-4-bilhoes-20042009-5.shl>>. Acesso em: set. 2014.

MOREIRA, Lucas. **Usar jQuery \$.ajax() e PHP para atualizar div sem refresh.** 2014. Disponível em: <<http://lucasmoreira.com.br/2013/06/23/usuario-jquery-ajax-e-php-para-atualizar-div-sem-refresh/>>. Acesso em: 23 ago. 2014.

MORGADO, Susana. **HTML: Hipertext Markup Language.** 2010. 15 f. Dissertação (Mestrado) - Curso de Tecnologias da Informação e Comunicação, Instituto Superior de Entre Douro e Vouga, Santa Maria da Feira, 2010. Cap. 1. Disponível em: <<http://pt.scribd.com/doc/46524096/Susana-Morgado-HTML>>. Acesso em: 07 set. 2014.

NASCIMENTO, Jean. **3 razões para usar MongoDB.** 2010. Disponível em: <<http://imasters.com.br/artigo/18334/mongodb/3-razoes-para-usar-mongodb/>>. Acesso em: 14 set. 2014.

NASCIMENTO, Jean. **NoSQL: Você realmente sabe do que estamos falando?.** 2010. Disponível em: <<http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/>>. Acesso em: 14 set. 2014.

NASCIMENTO, Thiago. **Desenvolvendo com Bootstrap 3:** um framework front-end que vale a pena! 2013. Disponível em: <<http://thiagonasc.com/desenvolvimento-web/desenvolvendo-com-bootstrap-3-um-framework-front-end-que-vale-a-pena>>. Acesso em: 16 jul. 2014.

OLIVEIRA JÚNIOR, Ismael de Souza. **Comparação Entre Frameworks Java para Desenvolvimento de Web Services:** Axis2 e CXF. 2013. 65 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade Fumec Faculdade de Ciências Empresariais, Belo Horizonte, 2013.

PALMEIRA, Thiago Vinícius Varallo. **Java:** história e principais conceitos. Disponível em: <<http://www.devmedia.com.br/java-historia-e-principais-conceitos/25178#ixzz3Clso68bO>>. Acesso em: 14 set. 2014.

PEREZ FILHO, Archimedes et al. **Monitoramento E Gerenciamento De Bacias Urbanas Associados A Inundação:** Diagnose Da Bacia Do Ribeirão Quilombo Na Região Metropolitana De Campinas Utilizando Geotecnologias. Revista do Departamento de Geografia, Campinas, v. 19, n. 1, p.44-55, jan. 2006.

PINHEIRO, Fernando Krein. **Sensor Ultrassônico HC-SR04.** 2014. Disponível em: <<http://blog.filipeflop.com/sensores/sensor-ultrassonico-hc-sr04.html>>. Acesso em: 18 out. 2014.

POLITOWSKI, Cristiano e MARAN, Vinícius. **Comparação de Performance entre PostgreSQL e MongoDB.** 2014. 10 f. Dissertação – Departamento de Ciências Exatas e Engenharias, Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ), Santa Rosa, 2014. Disponível em: <http://www.researchgate.net/publication/261871960_Comparao_de_Performance_entre_PostgreSQL_e_MongoDB>. Acesso em: 14 set. 2014.

RENNA, Roberto Brauer di et al. **Introdução ao kit de desenvolvimento Arduino.** Niterói: Grupo Pet-tele, 2013. 80 p. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/arduino/Tut_Arduino.pdf>. Acesso em: 06 set. 2014.

SANDAKITH, Lahiru. **Creating Bottom Up Web Service via Apache Axis2**. 2007. Disponível em: <http://www.eclipse.org/webtools/community/tutorials/BottomUpAxis2WebService/bu_tutorial.html>. Acesso em: 18 out. 2014.

SERSON, Roberto Rubinstein. **Programação Orientada a Objetos com Java 6**: Curso Universitário. São Paulo: Brasport, 2008. 492 p.

SILVA, João Paulo Rodrigues Pacheco. **Mapeamento de inundações no Brasil**: Proposta de gestão ambiental através de um sistema de informações geográficas. 2010. 15 f. Dissertação – Universidade Estadual Paulista, Rio Claro, 2010.

SILVA, Maurício Samy. **CSS3**: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das css3. São Paulo: Novatec, 2012. 46 p. Disponível em: <<http://novatec.com.br/livros/css3/capitulo9788575222898.pdf>>. Acesso em: 14 set. 2014.

SILVA, Maurício Samy. **JavaScript**: Guia do programador. São Paulo: Novatec, 2010. 608 p.

SOBCZAK, Maicon. **Trabalhando Requisições AJAX Com jQuery**. 2010. Disponível em: <<http://www.webmaster.pt/requisicoes-ajax-jquery-2216.html>>. Acesso em: 27 ago. 2014.

TAURION, Cesar. **Cloud Computing**: Computação em Nuvem. São Paulo: Brasport, 2009. 228 p.

THE DOCUMENT FOUNDATION. **LibreOffice**: Licenças. Disponível em: <<https://pt-br.libreoffice.org/sobre-nos/licencias/>>. Acesso em: 18 set. 2014.

THE DOCUMENT FOUNDATION. **LibreOffice**: Quem somos?. Disponível em: <<https://pt-br.libreoffice.org/sobre-nos/quem-somos/>>. Acesso em: 18 set. 2014.

TOSI NETO, Jayme. **Linux: Como criar um daemon**. 2012. Disponível em: <<http://legauss.blogspot.com.br/2012/07/linux-como-criar-um-daemon.html>>. Acesso em: 14 set. 2014.

TUCCI, Carlos Eduardo Morelli. **Modelos Hidrológicos**. 2. ed. Porto Alegre, RS: Ufrgs, 2005. 678 p.

UBUNTU. **ApacheMySQLPHP**. Ubuntu documentation. Disponível em: <<https://help.ubuntu.com/community/ApacheMySQLPHP>>. Acesso em: 24 ago. 2014.

VERAS, Manoel. **Arquitetura de Nuvem**: Amazon Web Services. São Paulo: Brasport, 2013. 416 p.

VERAS, Manoel. **Virtualização**. São Paulo: Brasport, 2011. 364 p.

WILSON, Mike. **Construindo Aplicações Node com MongoDB e Backbone**: Prototipação rápida e implantação escalável. São Paulo: Novatec, 2013. 240 p.

YANAGUI, Rafael. **PhoneGap tutorial**: Iniciando um projeto (completo). Disponível em: <<http://www.nanoincub.com.br/blog/tutoriais/tutorial-de-phonegap-iniciando-um-projeto>>. Acesso em: 06 ago. 2014.

ZHANG, Mason. **7 Reasons You Should Use MongoDB over DynamoDB**. 2013. Disponível em: <<http://www.masonzhang.com/2013/08/7-reasons-you-should-use-mongodb-over.html>>. Acesso em: 24 ago. 2014.

ANEXO A – Questionário utilizado no levantamento de requisitos

Prezado.

Estamos realizando uma pesquisa para obter informações sobre a necessidade, melhor forma de desenvolvimento e implantação de um Sistema Online de Acompanhamento de Enchentes, que será apresentado no Centro de Educação Profissional de Timbó (CEDUP) como um dos trabalhos de conclusão do curso Técnico em Informática com Habilitação em Desenvolvimento de Software.

Para realizar a pesquisa e obter resultados válidos, precisamos da colaboração de pessoas que vivem em cidades frequentemente atingidas por alagamentos. Por favor, preencha o questionário abaixo. Não há respostas certas ou erradas, mas é importante que você informe dados reais.

Esse questionário será utilizado para melhoria do sistema proposto. Nenhum nome ou resposta será divulgado.

Apos o preenchimento do questionário, favor devolvê-lo até dia 25 de agosto de 2014.

Desde já agradecemos sua especial atenção e colaboração.

Cordialmente,

Jonathan Eli Suptitz

Luan Carlos Purim

Roberto Luiz Debarba

Alunos técnicos do CEDUP Timbó.

Sistema Online de Acompanhamento de Enchentes e Inundações

1 – Na ferramenta online, você gostaria de poder simular o nível do rio e visualizar num mapa as áreas de alagamento?

() Sim

() Não

2 – Você gostaria que o mapa de alagamento possuísse escala de cores conforme a profundidade do local atingido?

() Sim

() Não

3 – Você gostaria de ter informações atualizadas sobre o status da chuva no local de medição?

() Sim

() Não

4 – Você gostaria de visualizar a previsão do tempo para os próximos dias?

() Sim

() Não

5 – Se o site possuísse dados em forma de gráficos, quais seriam importantes: (se necessário, preencha múltiplas respostas)

() Histórico de chuva.

() Histórico de nível do rio.

() Essas informações não são necessárias.

6 – Você gostaria que o site disponibiliza-se um mapa com imagens de satélite das nuvens sobre a região?

() Sim

() Não

7 – Você gostaria de informações sobre as enchentes passadas, como data, nível do rio, cidades atingidas e imagens?

- () Sim
() Não

8 – Você concordaria com propagandas discretas no site?

- () Sim
() Não

9 – Você gostaria de links para sites de utilidade pública, como prefeitura, defesa civil e previsão do tempo?

- () Sim
() Não

10 – Você gostaria que o site disponibilizasse uma ferramenta para calculo de rotas que evita caminhos alagados?

- () Sim
() Não

11 – Qual navegador de internet você utiliza no seu computador?

- () Google Chrome
() Mozilla Firefox
() Microsoft Internet Explorer
() Opera
() Safari
() Outro. _____

12 – Qual o sistema operacional do seu *smartphone*?

- () Android
() iOS
() WindowsPhone

- () Outro. Qual? _____
- () Não possuo *Smartphone*.

13 – Como você prefere acessar uma ferramenta online pelo seu *smartphone*?

- () Acesso ao site pelo navegador de internet do dispositivo.
- () Aplicativo instalado.

14 – Em estado de possível alagamento, você gostaria de receber notificações de alerta via:
(se necessário, preencha múltiplas respostas)

- () Aplicativo instalado de *smartphone*.
- () Correio eletrônico (e-mail).
- () Mensagem de texto no celular (SMS).
- () Não gostaria de receber notificações de alerta.

15 – Em estado de alagamento, quais tipos de notificações você gostaria de receber: (se necessário, preencha múltiplas respostas)

- () Nível do rio.
- () Previsão de alagamento em minha residência.
- () Profundidade de alagamento em minha residência.
- () Não gostaria de receber notificações.
- () Outro. _____

Agradecemos sua colaboração.

ANEXO B – Picos de enchentes da bacia do rio Itajaí-Açú

Ano	Data	Blumenau	Indaial	Timbó
1852	29/10	16,3m	8,3m	
1855	20/11	13,3m		
1862	20/11	9m		
1864	17/09	10m		
1868	27/11	13,3m		
1870	11/10	10m		
1880	23/09	17,1m	8,7m	
1888	23/09	12,8m		10m
1891	18/06	13,8m		
1898	1/05	12,8m		
1900	2/10	12,8m		
1911	29/05	16,9m	8,5m	9,3m
1911	20/06	9,86m		
1923	14/05	9m		
1925	14/01	10,3m	5,32m	
1926	9/11	9,5m		
1927	18/06	12,3m	6,55m	
1928	15/08	11,76m	6m	
1928	2/05	10,82m		
1931	18/09	10,8m	5,5m	5,3m
1931	25/05	10,8m	5,7m	5,4m
1932	4/10	9,85m		
1933	29/09	11,65m	5,7m	
1935	27/11	11,4m		6,5m
1939	3/08	11,2m	5,8m	4,5m
1943	17/05	10,25m		
1948	17/10	11,8m	5,9m	6,1m
1950	31/10	9,1m	4,72m	3,1m
1953	31/10	9,4m	4,9m	4,8m
1954	19/05	9,3m	4,69m	4,7m
1954	22/10	12,88m	6,1m	5,45m
1955	19/05	10,36m	5,26m	5,3m
1957	21/07	9,1m	4,65m	4,9m
1957	2/08	10,4m	5,22m	3,6m
1957	18/08	12,86m	6,48m	5,4m
1957	16/09	9,24m	4,64m	3,6m
1961	12/09	10,1m	5,08m	3,1m

1961	30/09	9,4m		
1961	1/11	12,18m	5,46m	6,5m
1966	13/02	9,82m	5m	4,05m
1969	6/04	9,89m	5,12m	4,95m
1971	10/06	10,1m	5,32m	3,5m
1972	29/08	11,07m	5,5m	4
1973	25/06	11,05m	5,76m	4,9m
1973	22/07	9,1m	4,74m	5,6m
1973	29/08	12,24m	6,1m	6,28m
1975	4/10	12,4m	6,34m	6,07m
1977	18/08	9,25m	4,96m	2,9m
1978	26/12	11,45m	5,94m	5,5m
1979	10/05	9,75m	5,32m	5,15m
1979	9/10	10,2m	5,64m	4,7m
1980	22/12	13,02m	7,1m	6,46m
1983	4/03	10,35m	5,58m	5,1m
1983	20/05	12,46m	6,72m	6,7m
1983	9/07	15,34m	7,76m	8,7m
1983	24/09	11,5m	6,06m	5,55m
1984	7/08	15,46m	8,04m	7,75m
1990	21/07	8,82m		5,7m
1992	29/05	12,8m	7,12m	9,75m
1992	1/07	10,62m	6,14m	6,16m
1997	1/02	9,44m	5,49m	6,2m
2001	1/10	11,02m	6,54m	7,95m
2008	24/11	11,72m		8,22m
2010	26/05	8,64m		7,11m
2011	31/08	8,7m	5,83m	6,21m
2011	9/09	12,8m	7,6m	9,86m

FONTE: CEOPS (2014)