UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM ENGENHARIA DE SOFTWARE

EDYWALDO RHONAN CARNEIRO SILVA

INTERFACE WEB DE MONITORAMENTO DE TRÁFEGO AÉREO

QUIXADÁ
2014

EDYWALDO RHONAN CARNEIRO SILVA

INTERFACE WEB DE MONITORAMENTO DE TRÁFEGO AÉREO

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientador Prof. Arthur de Castro Callado

QUIXADÁ
2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S581i Silva, Edywaldo Rhonan Carneiro
Interface Web de monitoramento de tráfego aéreo / Edywaldo Rhonan Carneiro Silva. – 2014.
50 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Bacharelado em Engenharia de Software, Quixadá, 2014.
Orientação: Prof. Dr. Arthur de Castro Callado
Área de concentração: Computação

1. Simuladores de controle de tráfego aéreo 2. Interface de usuário baseada na Web 3. Software - desenvolvimento I. Título.

CDD 004.6

EDYWALDO RHONAN CARNEIRO SILVA

INTERFACE WEB DE MONITORAMENTO DE TRÁFEGO AÉREO

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: _____ / junho / 2014.

BANCA EXAMINADORA

_____
Prof. Dr. Arthur de Castro Callado (Orientador)
Universidade Federal do Ceará-UFC

_____
Profª. Drª. Paulyne Matthews Jucá
Universidade Federal do Ceará-UFC

_____
Prof. MSc. Marcio Espíndola Freire Maia
Universidade Federal do Ceará-UFC

À minha mãe por tudo que ela fez, vem
fazendo e ainda fará por mim.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, em especial a minha mãe, e a minha família por me darem o suporte necessário para que a realização deste sonho fosse possível. Também gostaria de agradecer a todos os colegas de faculdade, não só os da minha turma, e a todas as pessoas que fazem a UFC – Campus Quixadá, pois todos fizeram parte desta realização. Ao professor Arthur de Castro Callado que durante quase 3 anos foi meu orientador na bolsa, me guiando em meus estudos, mostrando caminhos e mostrando que eu era capaz. Por último e não menos importante gostaria de agradecer a todos os amigos que conquistei em Quixadá durante minha estadia, que estiveram ao meu lado em momentos de alegria e em momentos de dificuldade.

"O futuro pertence àqueles que acreditam
na beleza de seus sonhos."
(Elleanor Roosevelt)

RESUMO

Transparência é cada vez mais uma característica buscada em qualquer serviço. Este trabalho busca trazer mais transparência sobre o uso do espaço aéreo brasileiro através de uma interface web de monitoramento de tráfego aéreo que esteja ao alcance de qualquer cidadão. Foi feito um levantamento bibliográfico sobre as tecnologias necessárias para o desenvolvimento da interface web e para o software que irá coletar as informações das aeronaves utilizando antenas compatíveis com a tecnologia *Automatic Dependent Surveillance - Broadcast* (ADS-B). Após este levantamento é apresentado o processo utilizado e explicado algumas peculiaridades sobre o desenvolvimento dos softwares.

Palavras chave: Tráfego aéreo. Interface web. Transparência.

ABSTRACT

Transparency is increasingly becoming a characteristic sought in any service. This paper seeks to bring more transparency to the Brazilian air space usage through a web interface for monitoring air traffic available for every citizen. A literature review has been made about the technologies necessary for the development of the web interface and for the software that will collect the data from the airships using antennas compatibles with Automatic Dependent Surveillance - Broadcast (ADS-B) technology. After this review the process used is shown and some peculiarities about the development of the softwares are explained.

Keywords: Air traffic. Web interface. Transparency.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O tráfego aéreo mundial está em forte ascensão em todo o mundo. Na América Latina, a previsão de crescimento de 2007 até 2025 é de 239% (STANISCIA, 2007). O sistema Automatic Dependent Surveillance (ADS) é um dos diversos sistemas usados para controle de tráfego aéreo. Os sistemas mais antigos, ainda em operação no país, permitem um período de atualização, tempo de espera entre uma mensagem e outra, de cerca 30 segundos enquanto o sistema ADS tem um período configurável entre 0,5 e 2 segundos. A diferença entre eles é muito grande e para os dias de hoje, 30 segundos para uma atualização sobre o posicionamento de uma aeronave é considerado muito preocupante. O modo mais comum de uso do sistema ADS é o broadcast (ADS-B). Neste modo, a aeronave periodicamente envia dados sobre identificação, altitude, velocidade, posição (GPS) e rota. O alcance de um receptor ADS-B é tipicamente de 370 km (linha de visada), o que pode ser aumentado com o uso de uma torre. No trabalho de Baud, Honore e Taupin (2006), eles executam a fusão de dados ADS-B obtidos por uma estação ADS-B e concluem que a utilização do ADS-B traz uma grande melhoria ao monitoramento aéreo devido a alta taxa de atualização e possuir um atraso muito baixo.

O trabalho que trata da decodificação de mensagens ADS-B provenientes do monitoramento do tráfego aéreo e persistência destas mensagens feito por Carvalho (2012), deixou como trabalho futuro a parte de exibição dos dados que foram obtidos e tratados em seu trabalho. Neste projeto, pretende-se fazer a exibição destes dados de forma hábil e de fácil entendimento para o usuário, que poderá ser utilizada por desde funcionários de aeroportos a qualquer cidadão que deseje obter alguma informação sobre algum voo.

Atualmente não existe no Brasil qualquer serviço deste tipo ao alcance da população, mas existe para aeroportos de grande porte devido ao custo de equipamento e software. Existe um site chamado FlightRadar24 que faz parte do que é pretendido a ser feito, porém ele não é brasileiro e não possui uma boa cobertura em nosso país. Com este trabalho, além de disponibilizar informações sobre voos de forma fácil e prática para qualquer pessoa que tenha interesse, será possível ajudar na administração de pequenos aeroportos do nosso país, e os sistemas desenvolvidos neste trabalho estarão disponíveis para qualquer pessoa que queira contribuir com uma antena ADS-B em sua própria casa.

Com este projeto pretende-se dar uma contribuição para a transparência no espaço aéreo brasileiro, deixando ao alcance de qualquer cidadão informações sobre voos através de um portal nacional com informações no idioma de nosso país. Atualmente, não temos essas informações de todos os pontos de nosso país e onde conseguir essas informações também não é de conhecimento da grande massa da nossa população, devido o site já citado (FlightRadar24) que faz este serviço ser um projeto de fora do nosso país, que não tem como foco a transparência aérea do Brasil.

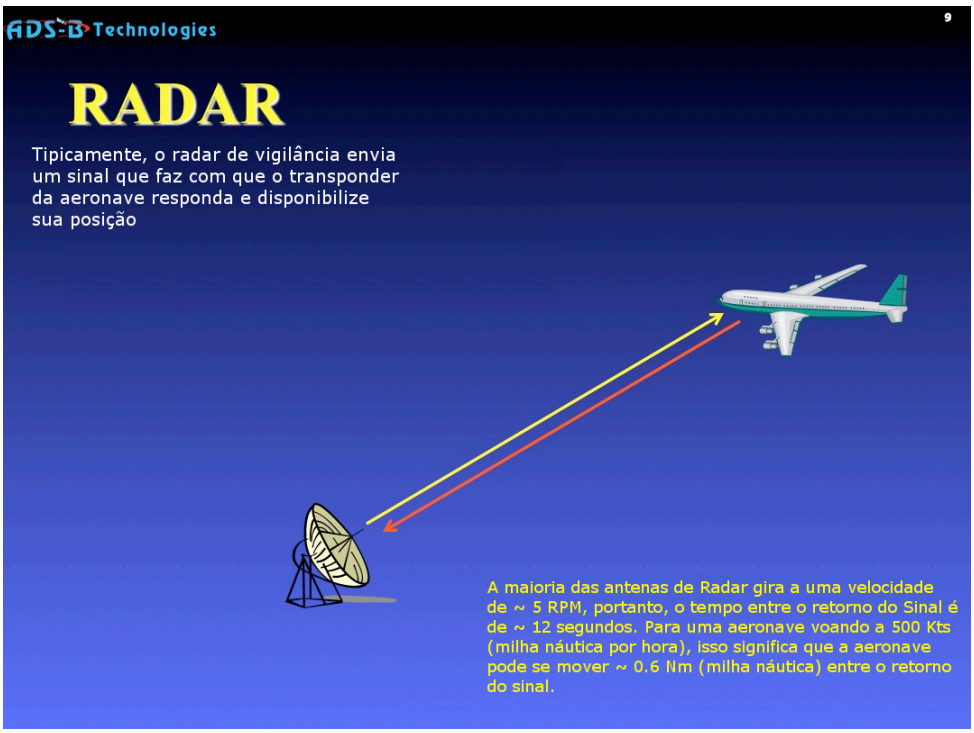
Este trabalho tem como objetivo a criação de uma interface web para monitoramento de tráfego aéreo, e para alcançar este objetivo maior o trabalho foi dividido em alguns objetivos específicos menores que são: utilização de uma API de mapas para apresentação dos dados obtidos pelo monitoramento de uma forma mais dinâmica para o usuário; e desenvolvimento de um serviço de acompanhamento de voos para que qualquer cidadão tenha acesso a informações sobre o tráfego aéreo brasileiro, desta forma contribuindo para a transparência do espaço aéreo brasileiro.

2 REVISÃO BIBLIOGRÁFICA

2.1 Monitoramento de tráfego aéreo com ADS-B

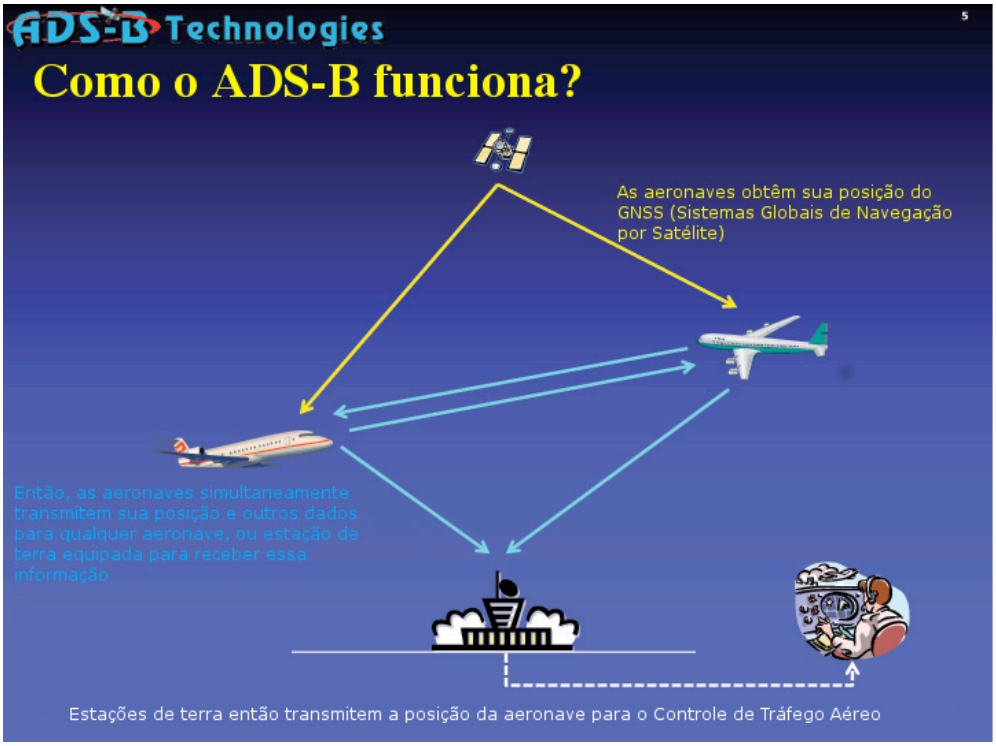
ADS-B (Automatic Dependent Surveillance - Broadcast) é a versão de ADS na qual o sistema de bordo das aeronaves transmite as informações do sistema de navegação (latitude, longitude, velocidade, etc) de forma automática, de tal forma que podem ser captadas por qualquer aeronave equipada ou por outro receptor no raio de alcance da transmissão. A frequência dos informes pode chegar a uma ou mais de uma mensagem ADS-B por segundo. ADS-B se destina a complementar e, eventualmente, substituir com vantagem a vigilância por meio de radares secundários. É através deste protocolo que serão obtidos os dados que irão ser exibidos na interface web de monitoramento de tráfego aéreo, pois como já foi dito anteriormente, o sistema mais utilizado atualmente, o SSR (Secondary Surveillance Radar), possui uma grande demora em sua atualização. A Figura 1, retirada do site corporativo ADS-B Technologies, explica o funcionamento de um radar convencional, tipicamente o radar envia um sinal para o transponder (abreviação de transmitter/responder) da aeronave que responde com sua posição. Na Figura 2, também retirada do site corporativo ADS-B Technologies, é explicado como o ADS-B funciona, a aeronave com ADS-B está sempre enviando sua localização (que é obtida através de GPS) para as estações e outras aeronaves ao alcance compatíveis com a tecnologia ADS-B.

Figura 1 – Funcionamento dos radares convencionais.



Fonte: ADS-B Technologies, 2013

Figura 2 – Funcionamento dos radares ADS-B.



Fonte: ADS-B Technologies, 2013

De acordo com Carvalho (2012) a tecnologia ADS-B é usada em todo o mundo como uma alternativa para o SSR. Implementar a tecnologia ADS-B provê vários benefícios para os Air Navigation Service Providers (ANSPs) e para a aviação comercial. Seus Custos de aquisição e manutenção são baixíssimos quando comparado aos radares convencionais (BARSHESHAT, 2011). Os ANSPs são as agências reguladoras do espaço aéreo, geralmente cada país possui a sua, mas pode haver casos de um país possuir mais de uma, onde cada conjunto de estados possui a sua agência reguladora. No Brasil a agência reguladora é o Departamento de Controle do Espaço Aéreo (DECEA).

2.2 Interface web

Para a criação da interface web de monitoramento de tráfego aéreo será utilizado um Web Map Service, que, conforme Wang et al (2004), "é um serviço que fornece informações espaciais para os seus usuários via imagens de mapa". O Web Map Service que utilizaremos será o Google Maps.

A API Javascript do Google Maps permite incorporar o Google Maps em suas páginas da Web. A API oferece diversos utilitários para manipulação de mapas e para a adição de conteúdo ao mapa por meio de diversos serviços, o que permite criar robustos aplicativos de mapas em seu website.

A utilização da API do Google Maps requer conhecimento em JavaScript que "é o nome real de uma instância de uma especificação de scripting mais ampla, a ECMAScript" (POWERS, 2010, p. 24) e é atualmente a principal linguagem para programação do lado cliente em navegadores web. Um motivo pelo qual JavaScript é tão popular é ser relativamente fácil de ser acrescentado a uma página web, tudo que precisa-se fazer é um incluir pelo menos um elemento HTML script na página, especificar "text/javascript" para o atributo type e adicionar qualquer JavaScript desejado. Para ajudar no desenvolvimento da parte JavaScript será utilizada a biblioteca JQuery que possui como foco principal a simplicidade. Algumas das funcionalidades que chamaram atenção para o uso do JQuery foram as seguintes: buscar informações no servidor sem necessidade de recarregar a página e simplificar tarefas específicas de JavaScript.

Devido à variedade de tamanho dos dispositivos atualmente, no desenvolvimento desta interface foi buscado um desenvolvimento responsivo, utilizando plugins e frameworks de interface web responsivos, que segundo Mohorovicic (2013), "é uma nova abordagem de

design web que permite a flexibilidade de um website se adaptar a qualquer dispositivo". Atentando para este detalhe, foi utilizado um plugin do jQuery, que é conhecido como Sidr[1], para criar menus laterais (como os implementados pelo Facebook em sua plataforma móvel) de uma forma fácil e responsiva. Para as demais partes da interface web (botões, menus) que foram necessários, foram utilizados os componentes do framework Bootstrap[2], que é um framework de *front-end* bastante popular na web e de código aberto.

2.3 Banco de dados

No desenvolvimento da aplicação será utilizado o SGBD (Sistemas de Gerência de Banco de Dados) PostgreSQL. Um dos motivos que nos levaram a esta escolha foi este possuir a licença BSD, pode ser modificado e distribuído por qualquer pessoa para qualquer finalidade, seja particular, comercial ou acadêmica, livre de encargos. Além de ser Software Livre, o PostgreSQL possui outras vantagens como ser multiplataforma e escalável. Comparando a outros bancos de dados do mercado ele também possui a seu favor o fato de possuir integridade referencial e suporte nativo a transações. O PostgreSQL será utilizado para guardar informações sobre as capturas de dados das aeronaves e suas rotas.

2.4 Desenvolvimento do lado servidor

A aplicação também será desenvolvida utilizando a linguagem de programação Java que rodará do lado do servidor. Java é uma linguagem de programação e uma plataforma de computação lançada pela primeira vez pela Sun Microsystems em 1995. É a tecnologia que capacita muitos programas como, por exemplo, utilitários, jogos e aplicativos corporativos, entre muitos outros. O Java é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão. Dentro do Java utilizaremos suas tecnologias Servlets e JSP, "teremos os servlets recebendo as requisições dos clientes, executando as partes mais pesadas do processamento da aplicação usando código 100% Java, sendo o responsável por gerar o conteúdo de resposta para o cliente um JSP. Um JSP é um arquivo script (interpretado inicialmente, depois compilado), que contém mesclado ao código Java qualquer outro tipo de conteúdo que deverá

[1] jQuery Sidr: http://www.berriart.com/sidr/
[2] Bootstrap: http://www.getbootstrap.com/

ser gerado dinamicamente, que pode ser XML, WML, HTML, etc" (MOREIRA NETO, 2006).

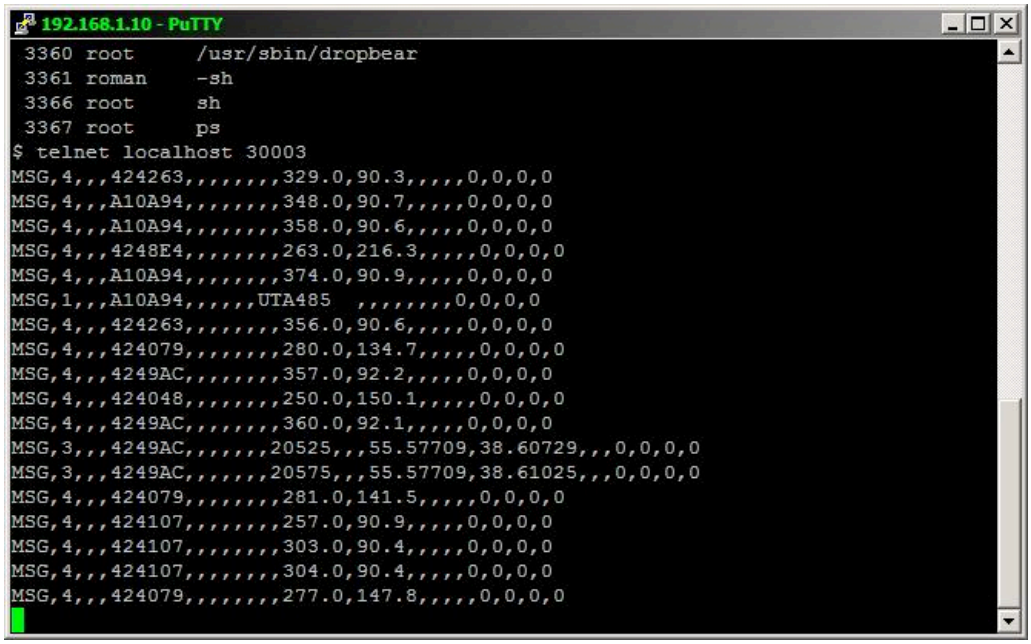
2.5 Coletor

Coletores são computadores que estarão com uma antena ADS-B conectada em uma de suas portas USB, como na Figura 3, e estarão executando dois softwares. Um software de terceiros chamado ADSBOX (http://adsbradar.ru/adsbox), que abrirá um socket na porta 30003 para o envio de mensagens ADS-B como mostrado na Figura 4, e um segundo software que foi parcialmente desenvolvido neste projeto, visto que o foco deste projeto é na interface, para o tratamento, persistência destas mensagens, envio desses dados para o servidor central.

Figura 3 – Antena ADS-B.

Fonte: Transair - Flight Equipment, 2014

Figura 4 – ADSBOX em funcionamento.



Fonte: Adsbradar Radarspotters Team, 2014

2.6 Web services

Web services “são serviços que baseiam-se no protocolo HTTP (*Hypertext Transfer Protocol*)” (CAMPOS, 2013). Está tecnologia será utilizada para o envio de dados do coletor para o servidor central.

Segundo Menéndez (2002 *apud* MORO; DORNELES; REBONATTO, 2011, p. 36) "a necessidade da integração entre aplicações, a utilização unificada de processos encontrados em diferentes sistemas e escritos em diferentes linguagens são alguns exemplos de carências encontradas em empresas nos dias de hoje. A fim de sanar estas questões, a tecnologia dos *Web services* foi criada, permitindo assim, disponibilizar formas de integrar sistemas distintos, modularizar serviços e capacitar a integração e consumo de informações".

De acordo com a W3C (2004 *apud* MORO; DORNELES; REBONATTO, 2011, p. 38) "um *Web Service* é um sistema de software desenvolvido para suportar interoperabilidade entre máquinas sobre uma rede, o qual pode apresentar uma interface que o descreve (WSDL, WADL). Outros sistemas interagem com os *Web services* por meio de mensagens SOAP, geralmente usando HTTP com serialização XML em conjunto com outros padrões relacionados a Web. Com esta tecnologia, torna-se possível que aplicações diferentes interajam entre si e sistemas desenvolvidos em plataformas diferentes se tornem compatíveis. Os *Web services* são componentes que permitem que aplicações enviem e recebam dados em

formatos variados. Cada aplicação pode ter a sua própria linguagem, que é traduzida para uma linguagem universal, como é o caso do formato XML".

Dentro desta tecnologia temos que escolher qual arquitetura utilizarmos, e devido experiências práticas passadas com a arquitetura REST utilizando o formato JSON, ela foi a escolhida utilizando o *framework* Jersey[3], que é de código aberto, para acelerar o desenvolvimento.

No trabalho de Moro, Dorneles e Rebonatto (2011), afirma-se que o termo REST é geralmente usado para descrever qualquer interface que transmita dados de um domínio específico sobre HTTP sem uma camada adicional de mensagem como SOAP ou *session tracking* via *cookies* HTTP.

2.7 Web sockets

Web sockets definem um canal de comunicação *full-duplex* que opera sobre um único socket na Web. Isto não é apenas uma melhoria incremental para as convencionais comunicações HTTP, *Web sockets* representam um avanço colossal, especialmente para aplicações web em tempo real. Esta tecnologia pode oferecer uma incrível redução do tráfego desnecessário e da latência se comparado às soluções convencionais. Normalmente, quando um navegador visita uma página, uma requisição HTTP é enviada para o servidor que hospeda a página, que reconhece essa requisição e envia de volta uma resposta, e se você quiser a informação mais atual você poderia atualizar a página manualmente (recarregar a página), o que obviamente não é a melhor solução. Atuais tentativas de prover uma aplicação em tempo real envolvem *polling* e outras tecnologias *push* no lado do servidor, que é uma boa solução se o exato intervalo de entrega das mensagens é conhecido. (LUBBERS; GRECO, 2010).

Está tecnologia é utilizada neste projeto para fazer a conexão entre o *back-end* da aplicação com o seu *front-end*.

2.8 Gerenciador de dependências

Maven é uma ferramenta para auxiliar na gerência de configuração, ele utiliza-se de um padrão na forma de construir projetos, com uma definição clara do que constitui o projeto, uma forma fácil de publicar informações do projeto e uma forma de compartilhar

[3] Jersey: https://jersey.java.net/

JARs entre vários projetos. Essa ferramenta pode ser utilizada para construir e gerenciar qualquer projeto Java, e com ela espera-se fazer o dia-a-dia de trabalho de desenvolvedores Java mais fácil.

A configuração dessa ferramenta se dá através do POM (*Project Object Model*), onde o desenvolvedor define as dependências que serão utilizados pelo projeto com suas respectivas versões. Podendo também definir versão do Java utilizado no projeto, isso facilita muito a configuração de ambiente.

2.9 Controle de versão

Git é um sistema de controle de versão grátis e de código aberto (*open source*) feito para lidar com tudo, dos pequenos projetos aos grandes projetos, com velocidade e eficiência. GitHub é um serviço de host de projetos utilizando Git, que é grátis para códigos open-source públicos, como é o caso do sistema a ser desenvolvido por este projeto.

Todos os códigos fontes estão disponíveis nos seguintes endereços:

- https://github.com/rhonan/radar-livre-interface
- https://github.com/rhonan/radar-livre-coletor

3 PROCEDIMENTOS METODOLÓGICOS

O projeto será realizado em seis etapas, detalhadas nos próximos itens.

3.1 Elicitação dos requisitos

Nesta atividade, de acordo com Sommerville (2002), se trabalha com o cliente para descobrir mais informações sobre o domínio da aplicação, que serviços o sistema deve fornecer, o desempenho exigido do sistema, as restrições de hardware e assim por diante.

Essa atividade está dividida entre outras etapas que são:

1. Compreensão do domínio: Os analistas devem desenvolver sua compreensão do domínio da aplicação. Por exemplo: se for exigido um sistema para supermercado, o analista deverá descobrir com operam os supermercados.

2. Coleta de requisitos: É o processo de interagir com os *stakeholders* (parte interessada) do sistema para descobrir seus requisitos. Obviamente, parte da compreensão do domínio desenvolve-se durante essa atividade.

3. Classificação: Essa atividade considera o conjunto não estruturado dos requisitos e os organiza em grupos coerentes.

4. Resolução de conflitos: Quando múltiplos *stakeholders* estão envolvidos, os requisitos apresentarão conflitos. Essa atividade ocupa-se de encontrar e solucionar esses conflitos.

5. Definição das prioridades: Em qualquer conjunto de requisitos, alguns serão mais importantes do que os outros. Esse estágio envolve a interação com os *stakeholders*, para descobrir os requisitos mais importantes.

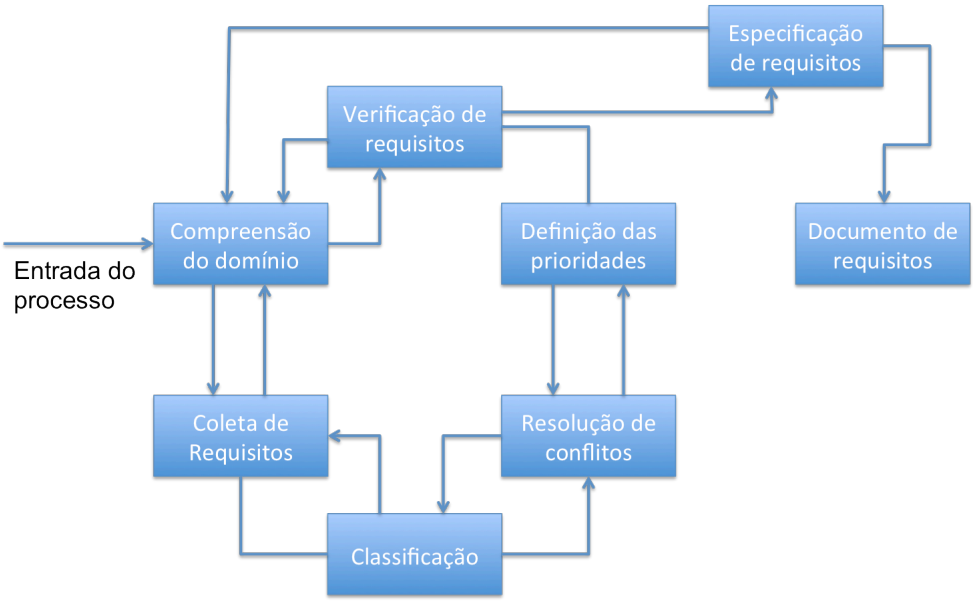
6. Verificação dos requisitos: Os requisitos são verificados, a fim de se descobrir se eles são completos e consistentes e se estão em concordância com o que os *stakeholders* realmente desejam do sistema.

A Figura 3 mostra que o levantamento e análise de requisitos é um processo iterativo, com *feedback* contínuo de cada atividade para as outras. O ciclo começa com a compreensão do domínio e termina com a verificação dos requisitos. A compreensão dos requisitos pelo analista melhora a cada fase do ciclo.

Neste projeto consideramos como cliente o professor orientador deste trabalho, pois este também é o professor orientador do projeto Atualização da monitoração aeronáutica e auto-sustentabilidade, do qual este trabalho também faz parte.

O produto obtido desta etapa pode ser encontrado no Apêndice A.

Figura 5 – Processo de levantamento e análise de requisitos.



Fonte: SOMMERVILLE, 2002

3.2 Estudo das tecnologias necessárias

Esta etapa consiste no estudo e aprendizado das tecnologias necessárias para o desenvolvimento do projeto que ainda não são totalmente dominadas, ou são novas para o desenvolvedor, por exemplo: Google Maps API, JavaScript. Ainda durante a execução desta etapa é possível iniciar a etapa seguinte (Criação de protótipos com fins de aprendizado e melhor entendimento dos requisitos), pois ambas podem possuir algumas atividades em comum, como exemplo podemos citar a própria criação de protótipos.

Esta etapa já foi iniciada anteriormente com o aprendizado da Google Maps API e para fixar o conteúdo adquirido sobre a Google Maps API, foi ministrado um minicurso de 6 horas na Escola de Férias 2012 da Universidade Federal do Ceará, Campus Quixadá. No minicurso foi apresentada grande parte das funções da API e para a fixação do conteúdo apresentado foram resolvidos diversos exercícios.

Durante a etapa de desenvolvimento viu-se a necessidade de voltar a esta etapa, pois houve a necessidade de utilização de uma tecnologia que não estava prevista inicialmente (*Web Sockets*). Houve uma dificuldade inicial devido às formas de implementação encontradas na rede serem compatíveis apenas com versões específicas do servidor de

aplicações (Tomcat), mas esta dificuldade logo foi superada e conseguimos utilizar a implementação mais atual com uma versão do servidor de aplicação bem atual.

3.3 Criação de protótipos com fins de aprendizado e melhor entendimento dos requisitos

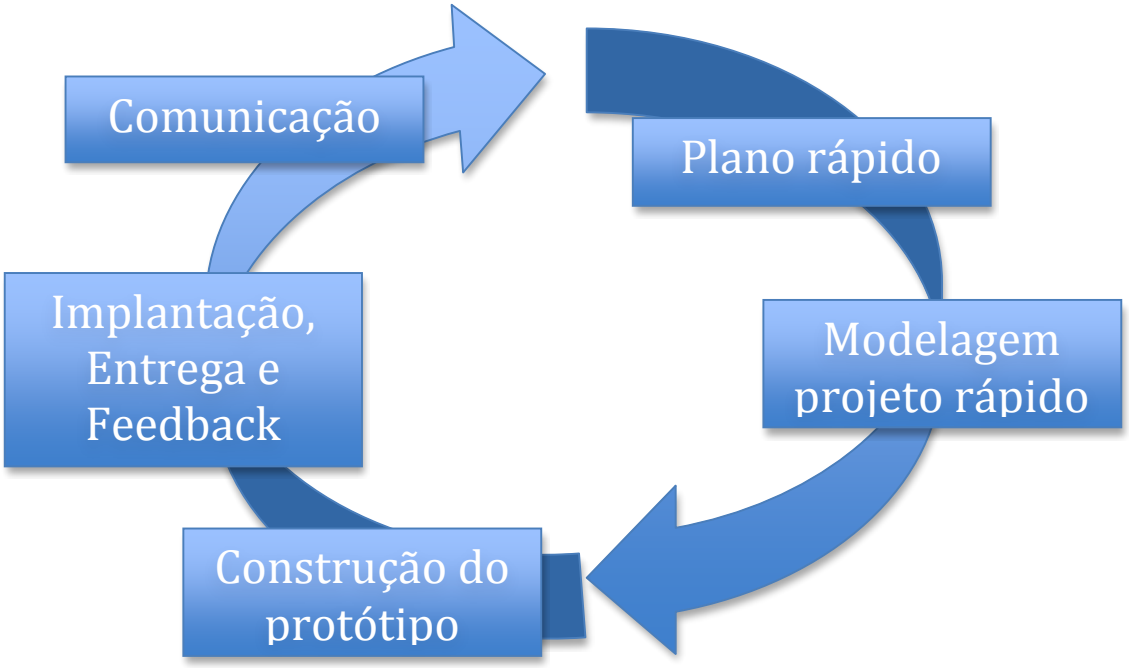
Segundo Pressman (2006), o cliente, frequentemente, define um conjunto de objetivos gerais para o software, mas não identifica detalhadamente requisitos de entrada, processamento, ou saída. Em outros casos, o desenvolvedor pode estar inseguro da eficiência de um algoritmo, da adaptabilidade de um sistema operacional ou da forma que a interação homem/máquina deve assumir. Nessas, e em muitas outras situações, um paradigma de prototipagem pode oferecer a melhor abordagem.

Independente da maneira como é aplicado, o paradigma de prototipagem auxilia o engenheiro de software e o cliente a entenderem melhor o que dever ser construído quando os requisitos estão confusos.

O paradigma de prototipagem começa com a comunicação. O engenheiro de software e o cliente encontram-se e definem os objetivos gerais do software, identificam as necessidades conhecidas e delineiam áreas que necessitam de mais definições. Uma iteração de prototipagem é planejada rapidamente e a modelagem ocorre. O projeto rápido concentra-se na representação daqueles aspectos do software que estão visíveis para o cliente/usuário. O projeto rápido leva à construção de um protótipo, que é implantado e depois avaliado pelo cliente/usuário. O feedback é usado para refinar os requisitos do software. A iteração ocorre à medida que o protótipo é ajustado para satisfazer as necessidades do cliente, e, ao mesmo tempo, permitir ao desenvolvedor entender melhor o que precisa ser feito.

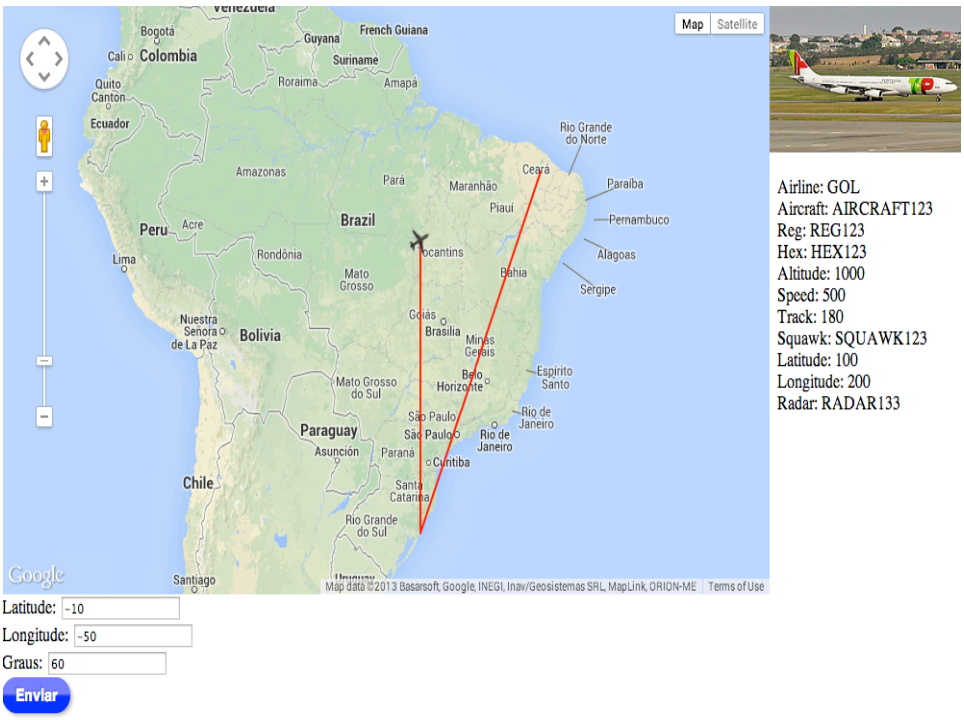
Na Figura 7 podemos ver um dos protótipos elaborados nesta fase para melhor entendimento dos requisitos e utilizado também para etapa de estudo das tecnologias necessárias. A principal função deste protótipo foi para o aprendizado das funcionalidades disponíveis na API do Google Maps.

Figura 6 – Processo de Prototipação.



Fonte: PRESSMAN, 2006

Figura 7 – Protótipo da interface



Fonte: Elaborada pelo autor

3.4 Desenvolvimento

Durante a etapa de desenvolvimento buscar-se-á seguir os ideais do manifesto ágil que são: Indivíduos e interação entre eles mais que processos e ferramentas, Software em funcionamento mais que documentação abrangente, Colaboração com o cliente mais que negociação de contratos, Responder a mudanças mais que seguir um plano. O período de desenvolvimento foi dividido em treze *Sprints,* que serão curtas, uma semana, devido ao curto tempo e também para aumentar o número de entregas e o feedback.

Nenhum processo de desenvolvimento foi definido para esta etapa por ser um projeto com apenas um desenvolvedor, não há nenhum time para se fazer reuniões de revisão e retrospectiva de *Sprints*.

O desenvolvimento contou com *Sprints* com duração de uma semana, onde funcionalidades do sistema eram apresentadas ao cliente semanalmente via reunião presencial. Em cada reunião foi gerada uma ata de reunião informal pelo próprio cliente, para que o mesmo pudesse acompanhar o que foi feito e o que seria feito para a próxima reunião. Estas atas de reunião estão acessíveis para os integrantes do projeto na ferramenta Oro-aro[4].

3.5 Testes

Para Pressman (2006), uma vez gerado o código-fonte, o software deve ser testado para descobrir (e corrigir) tantos erros quanto possível antes da entrega ao seu cliente. Sua meta é projetar uma série de casos de teste que têm uma grande probabilidade de encontrar erros; mas como? É ai que as técnicas de teste de software entram em cena. Essas técnicas fornecem diretrizes sistemáticas para projetar testes que exercitam a lógica interna e as interfaces de cada componente de software, e exercitam os domínios de entrada e saída do programa para descobrir erros nas funções, no comportamento e no desempenho do programa.

Revisões e outras atividades podem e efetivamente descobrem erros, mas elas não são suficientes. Toda vez que o programa é executado, o cliente testa. Assim temos que executar um programa antes que ele chegue ao cliente, com o objetivo específico de encontrar e remover todos os erros. Para encontrar o maior número possível de erros, testes devem ser conduzidos sistematicamente e casos de teste devem ser projetados usando técnicas disciplinadas.

[4] Oro-aro: http://www.oro-aro.com/

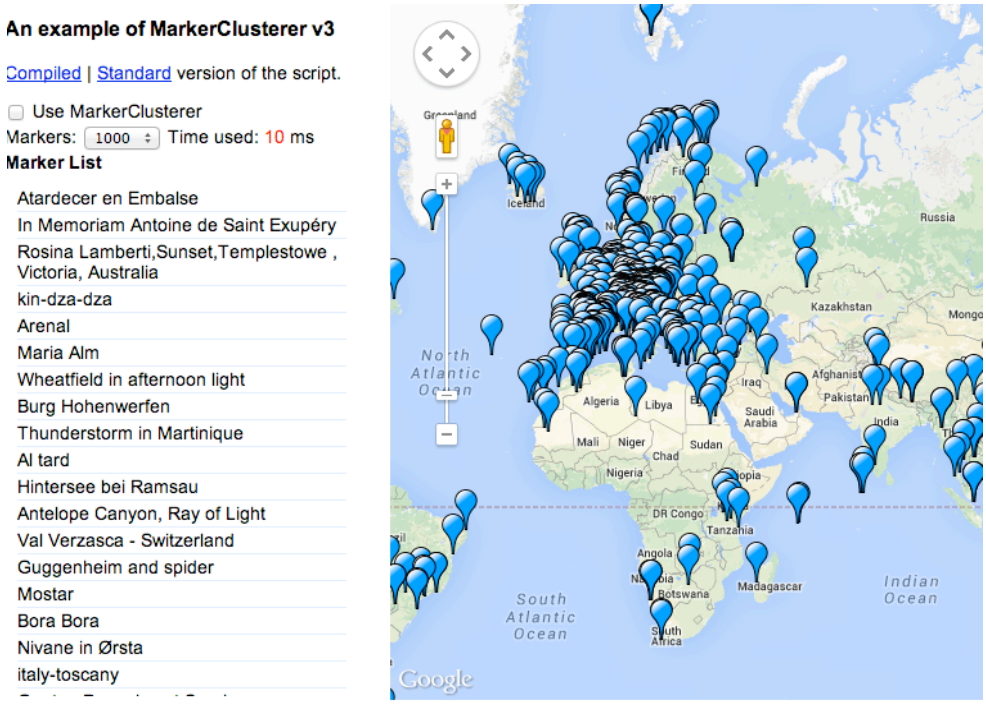
Para aplicações convencionais, o software é testado sob duas perspectivas diferentes: a lógica interna do programa é exercitada usando técnicas de projeto de casos de teste "caixa-branca". Requisitos de software são exercitados por meio de técnicas de projeto de casos de teste "caixa-preta". Para aplicações orientadas a objetos, o teste começa antes da existência do código-fonte, mas, uma vez gerado o código, uma série de testes é projetada para exercitar operações em uma classe e examinar se existem erros à medida que uma classe colabora com outras. Conforme as classes são integradas para formar um subsistema, testes baseados no uso, com abordagens baseadas em falhas, são aplicadas para exercitar plenamente as classes que colaboram. Finalmente, casos de uso ajudam no projeto de testes a descobrir erros no âmbito da validação do software. Em cada caso, o objetivo é encontrar o maior número de erros com a menor quantidade de esforço e tempo.

Devido ao cronograma apertado, não houve tempo para o desenvolvimento de testes automatizados. Testes foram feitos na interface ao final de cada *Sprint*, quando erros foram encontrados, eles foram corrigidos na Sprint seguinte.

Para testar a performance da API do Google Maps com diversos marcadores, foi utilizado uma aplicação web[5] desenvolvida por terceiros onde podemos adicionar vários marcadores de uma só vez e a aplicação nos diz quanto tempo o navegador demorou para renderizar todos os marcadores. Em todos os testes o tempo levado pelo navegador para renderizar vários marcadores foi muito curto, menos de um segundo, como pode ser observado na Figura 8.

[5] MarkerClusterer v3 Example: http://google-maps-utility-library-v3.googlecode.com/svn/trunk/markerclusterer/examples/speed_test_example.html

Figura 8 – *Speed test*



Fonte: Elaborada pelo Autor

4 DESENVOLVIMENTO

Nesta sessão será apresentado como foi desenvolvida cada parte deste projeto, apresentando seus desafios e soluções encontradas pelo decorrer do desenvolvimento. Note que também foi desenvolvido parcialmente o Coletor, que não se encontra no escopo da interface.

4.1 Servidor

Servidor é a aplicação Web que os usuários irão acessar para usufruir deste serviço, essa aplicação rodará em um servidor de aplicação Tomcat 7.0.47 ou superior, devido a algumas bibliotecas presentes nestas versões que são necessárias para o uso de Web Sockets. Por estas mesmas razões o projeto rodará em Java 7.

4.1.1 Front-end

No desenvolvimento do *Front-end*, a primeira etapa é criar um layout básico, isto foi feito com a ajuda do framework Bootstrap que já foi mencionado anteriormente. Após

feito o posicionamento destes componentes do Bootstrap, devemos importar a API de mapas escolhida à nossa página, que no caso é a API do Google Maps. Essa importação é feita como na Figura 8, após ter criado a *tag* no html com o id mapa, e ter importado a API com sua devida chave. Para gerar a chave podemos ver o passo a passo na própria documentação do Google[6].

Figura 9 – Importando o Google Maps para sua página

```
var mapOptions = {
  center: new google.maps.LatLng(-14.239424,-53.186502),
  zoom: 4,
  disableDefaultUI: true,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
map = new google.maps.Map(document.getElementById("mapa"),
    mapOptions);
```

Fonte: Elaborada pelo autor

Os demais métodos fornecidos pela API do Google Maps podem ser encontrados em sua própria documentação, eles não serão abordados aqui por serem diversos.

Para exibir as informações detalhadas de uma aeronave foi utilizado o plugin do jQuery Sidr, que exibe de forma simples e responsiva um menu lateral. Foi adicionado um Listener (elemento da API do Google Maps), em cada marcador (aeronave) para quando forem clicados chamarem a abertura deste menu com as informações do marcador em questão.

A parte mais importante do *Front-end* é o HTML5 Web Sockets, através dele que iremos atualizar os marcadores no mapa em tempo real.

Figura 10 – Inicializando o Web Socket no *Front-end*

```
var wsUri = "ws://" + document.location.host + "/Radar-Livre/websocket";
var websocket = new WebSocket(wsUri);

websocket.onopen = function(event) { onOpen(event); };
websocket.onmessage = function(event) { onMessage(event); };
```

Fonte: Elaborada pelo autor

[6] Documentação Google Maps API: https://developers.google.com/maps/documentation/javascript/tutorial?hl=pt-br#api_key

Apenas com essas linhas de código o *socket* é aberto com o *Back-end*. O método *onopen* é chamado assim que a conexão é aberta e o método *onmessage* é chamado sempre que uma nova mensagem é recebida pela interface, neste caso, serão as informações de atualização, adição e remoção de marcadores.

Um pequeno problema que ocorreu nesta fase foi a impossibilidade de rotacionar um marcador para indicar sua direção. Para resolver este problema foi utilizado um recurso do HTML5 chamado APPCACHE para já deixar pré-carregado 360 imagens de nossos marcadores (um para cada grau), assim sempre que o grau muda alteramos a imagem de nosso marcador para uma das 360 imagens que temos pré-carregadas no cache do navegador. Este recurso é utilizado da seguinte forma: <html manifest="radar-livre.appcache">, neste arquivo .appcache temos escrito o endereço de cada arquivo que gostaríamos de carregar no cache do navegador assim que o site é aberto.

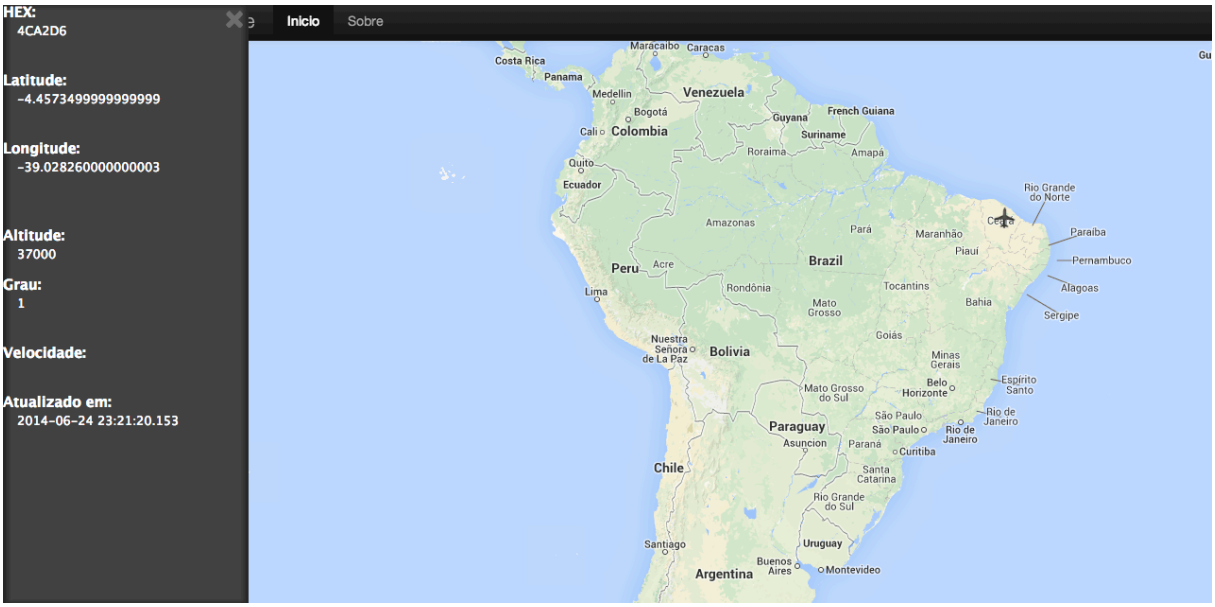
Figura 11 – Exemplo de um arquivo .appcache simples

```
CACHE MANIFEST
img/aeronaves/rotacionado0.png
img/aeronaves/rotacionado1.png
img/aeronaves/rotacionado2.png
img/aeronaves/rotacionado3.png
img/aeronaves/rotacionado4.png
img/aeronaves/rotacionado5.png
img/aeronaves/rotacionado6.png
img/aeronaves/rotacionado7.png
img/aeronaves/rotacionado8.png
img/aeronaves/rotacionado9.png
img/aeronaves/rotacionado10.png
img/aeronaves/rotacionado11.png
```

Fonte: Elaborada pelo autor

Na Figura 12 é mostrado o resultado final da Interface web em funcionamento em conjunto com o coletor.

Figura 12 – Interface web



Fonte: Elaborada pelo autor

Na Interface web à cima podemos observar apenas uma aeronave no mapa e o menu lateral do lado esquerdo com informações detalhadas desta aeronave, este menu foi desenvolvido utilizando o plugin Sidr. Note que este menu nem sempre está visível, apenas quando clicamos na aeronave, ele pode ser fechado clicando no “X” ou clicando em qualquer lugar do mapa.

4.1.2 Back-end

Para receber as informações vindas dos coletores, foi criado um *Web Service* utilizando o padrão REST no servidor utilizando a biblioteca Jersey. Sua implementação é bastante simples, utiliza-se de notações como: @GET, @POST, @PUT, @DELETE, @Path, @Produces, @Consumes @PathParam. Estas notações no geral indicam que tipo de método HTTP o método Java está esperando para ser chamado, o seu endereço, se ele produz alguma saída ou se ele irá consumir alguma coisa, no caso desta aplicação o método mais utilizado será o que consome JSON, para consumir as informações que irão vir do coletor em formato JSON e então salvar no banco de dados local.

Para salvar as informações no banco de dados local foi decidido não utilizar nenhum *framework* de persistência (exemplo: Hibernate), essa decisão foi tomada buscando

maior controle nas consultas e inserções no banco de dados. Dentro do projeto existe o pacote DAO (*Data Access Object*), que segue um padrão de projeto, onde esses objetos estão designados apenas para acesso ao banco. Neste pacote DAO é que estão todas as consultas e inserções ao banco de dados que podem ser feitas pelo servidor.

No *Back-end* também teremos que desenvolver a parte do servidor do *Web Socket*. Criamos uma classe para isto e através da notação @ServerEndpoint definimos qual será o endereço do nosso *Web Socket* para que o *front-end* possa acessar. Assim como no *front-end*, o *back-end* também terá métodos *onmessage*, *onopen*, *onclose*, que serão chamados sempre que uma dessas ações ocorrer. Podemos ter uma visão geral de como se parece a classe que define o servidor do *Web Socket* na Figura 12. O método que está con a notação *onmessage* não faz nada neste caso, pois em nenhum momento o *front-end* envia mensagens para o *back-end*, apenas o contrário. Já o método *onopen* envia uma mensagem para o cliente que acaba de se conectar no *Web socket*, ele cria uma sessão e adiciona à lista de conexões, então cria um objeto JSON para enviar para este cliente que acaba de se conectar.

Figura 13 – *Web Socket* no *back-end*

```
@ServerEndpoint("/websocket")
public class WSocket {
    private static final Set<WSocket> connections = new CopyOnWriteArraySet<>();
    private Session session;

    @OnMessage
    public String onMessage(String message) {
        return null;
    }

        @OnOpen
    public void onOpen (Session session) throws IOException {
        this.session = session;
        connections.add(this);
        Observacao observacao = new Observacao(-14.239424,-53.186502, 180,"TESTE", "ADD");
        Gson gson = new Gson();
        session.getBasicRemote().sendText(gson.toJson(observacao));
    }
```

Fonte: Elaborada pelo autor

Na Figura 13 é mostrado como *Web Socket* faz para enviar uma mensagem para todos os clientes conectados (*Broadcast*). No projeto foi desenvolvido um método *broadcast* na classe do *Web Socket* para fazer isto. O método basicamente percorre a lista de conexões enviando um objeto JSON para cada um dos clientes que estão na lista de conexões.

Figura 14 – Como o *Web Socket* envia uma mensagem para todos os clientes

```
for(WSocket client : connections){
        try{
                client.session.getBasicRemote().sendText(gson.toJson(new Observacao(
        }catch(IOException e){
                connections.remove(client);
                try{
                        client.session.close();
                }catch(IOException el){

                }
        }
```

Fonte: Elaborada pelo autor

4.2 Coletor

Para começar o desenvolvimento do coletor era necessário termos ao nosso alcance as mensagens vindas da antena ADS-B, durante algumas pesquisas a sites e forums que tratam deste assunto foi encontrado o software chamado ADSBOX. Durante os testes deste software houve um problema que não era possível fazer o software funcionar corretamente, com mais pesquisas a sites e fórums foi descoberto que era necessário rodar um comando para reiniciar a porta serial para então o ADSBOX funcionar corretamente. Segue abaixo este comando:

echo '#43-02' > /dev/ttyACM0

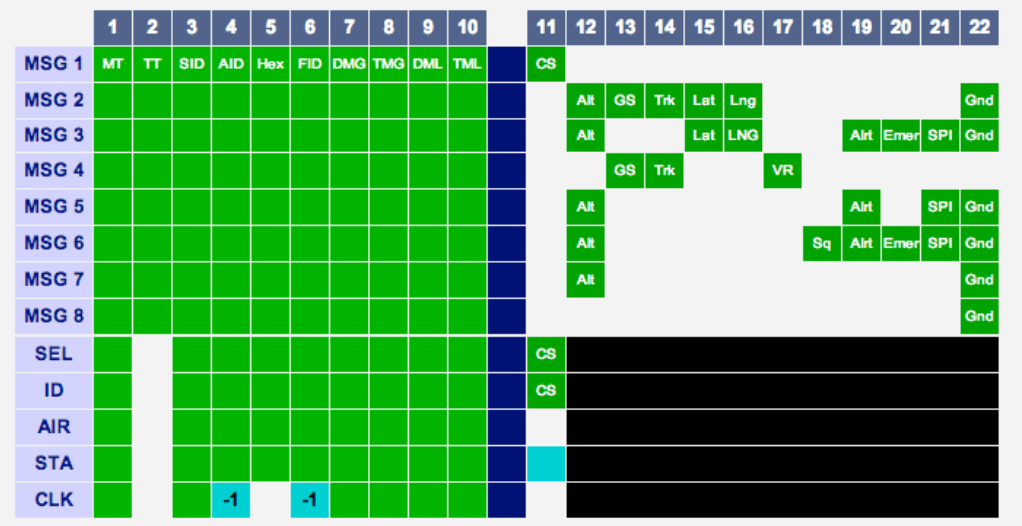
Após executado este comando podemos executar o ADSBOX corretamente pelo comando abaixo, note que o parâmetro –l e o parâmetro –g são respectivamente a latitude e longitude aproximada de onde o coletor se encontra:

./adsbox -s /dev/ttyACM0 -l -4.970833 -g -39.015 –d

Agora com o ADSBOX rodando temos acesso as mensagens ADS-B no socket que é aberto na porta 30003, como pode ser visto na Figura 4 mostrada anteriormente. O trabalho então foi decifrar estas mensagens que não são apresentadas de uma forma clara. Inicialmente este trabalho teve uma grande dificuldade, pois teríamos que deduzir o que vinha em cada campo de cada tipo de mensagem, mas após muitas pesquisas foi descoberto um site[7] onde as mensagens ADS-B estavam bem decifradas, como pode ser visto na Figura 12.

[7] SBS-1 BaseStation Tutorial 4.2: http://www.homepages.mcb.net/bones/SBS/Article/Barebones42_Socket_Data.htm

Figura 15 – Tabela das Mensagens ADS-B decifradas



Fonte: SBS-1 BaseStation Tutorial 4.2, 2014

Com as mensagens decifradas em mãos, foi necessário fazer um software para ficar conectado no *socket* da porta 30003 lendo as mensagens e salvando-as de acordo com o seu tipo. Para o tratamento da mensagem que vem da seguinte forma: *MSG,8,5,211,4CA2D6,10057,2008/11/28,14:53:50.391,2008/11/28,14:58:51.153,,,,,,,,,,0*, foi utilizado o método SPLIT do Java que quebra a mensagens em partes sempre que encontra um dado caractere, no caso a “,” (vírgula).

Figura 16 – Utilização do SPLIT para tratar as mensagens ADS-B

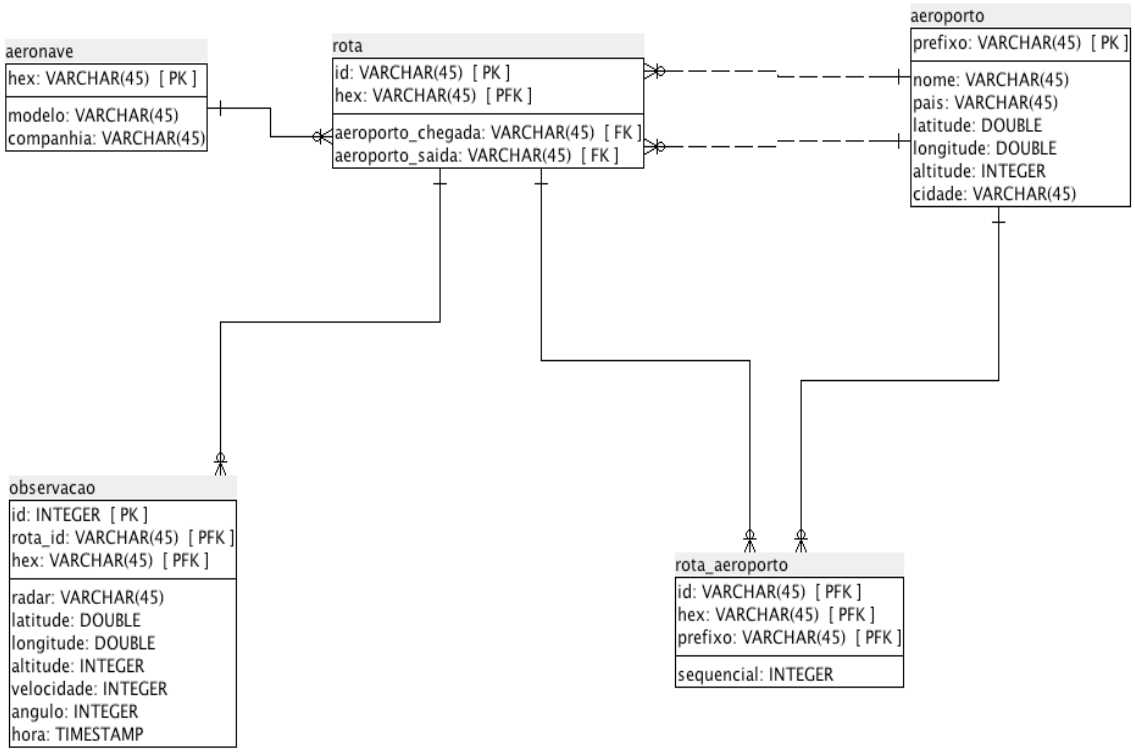
```
String mensagem = in.readLine();
String[] mensagem_quebrada = mensagem.split(Pattern.quote(","));
```

Fonte: Elaborada pelo autor

Para a modelagem visual do banco de dados e geração do script do banco de dados a partir dessa modelagem visual, foi utilizado a ferramenta SQL Power Architect[8].

[8] SQL Power Architect: http://www.sqlpower.ca/page/architect

Figura 17 – Modelo relacional do banco



Fonte: Elaborada pelo autor

No coletor foi desenvolvido o lado cliente do *framework* Jersey, para que cada mensagem que o coletor receba, além de salvar em seu banco de dados local, envie para o *Web Service* no servidor.

5 TRABALHOS FUTUROS

Com o que foi desenvolvido neste trabalho abrimos campo para diversos trabalhos futuros, entre eles facilmente podemos citar: melhorias na interface e integração desta interface com outros serviços já existentes, como por exemplo algum serviço que entregue dados sobre o voo (origem, destino, empresa) através do código do voo, em uma rápida pesquisa foi encontrado alguns destes serviços mas todos pagos, outro trabalho que poderia ser feito seria em cima do coletor, terminando o seu desenvolvimento e atentando para a persistência das mensagens no banco de dados, inserindo dados já tratados e fundidos (não inserir linhas no banco com informações faltando), pois cada mensagem só vem uma pequena

parte das informações, para ter uma informação completa é necessário fundir várias mensagens ADS-B, e às vezes podemos não receber todas estas mensagens.

6 CONSIDERAÇÕES FINAIS

Com todas as informações repassadas aqui, podemos concluir que a interface web de monitoramento de tráfego aéreo foi desenvolvida utilizando apenas tecnologias de ponta e atuais, que são padrões no mercado. Por esta aplicação se encontrar em um repositório público (GitHub) qualquer pessoa terá acesso ao seu código fonte, e caso está pessoa deseje implementar alguma nova funcionalidade ou fazer alguma melhoria em seu código fonte, facilmente encontrará documentações e artigos sobre as tecnologias necessárias.

Para que seja obtido uma aplicação completa em funcionamento (*online*), será necessário a conclusão da implementação do coletor, deixando-o robusto e fazendo uma fusão e tratamento dos dados de forma completa, como foi explicado no tópico que fala sobre o coletor e deixado como trabalho futuro. Está implementação do software coletor não foi colocada no escopo deste trabalho por questão de tempo e das dificuldades que seriam encontradas, como foi o caso do formato das mensagens ADS-B que não era conhecido por ninguém do projeto.

Este trabalho deixa uma grande contribuição para o projeto Atualização da Monitoração Aeronáutica e Auto-Sustentabilidade, pois a construção da interface atinge um dos principais objetivos deste projeto que é trazer um ganho para a transparência do espaço aéreo brasileiro, e também deixa encaminhado um software coletor que terá seu desenvolvimento concluído pelos membros do projeto.

REFERÊNCIAS

ADSBox - quiet and economical server for broadcasting ADS-B!. [site corporativo]. Disponível em < http://adsbradar.ru/adsbox>. Acesso em: 19 mai. 2014.

ADS-B Technologies. [site corporativo]. Disponível em <http://www.ADS-B.com>. Acesso em: 31 mai. 2013.

BARSHESHAT, A. A. **Implementation of ADS-B Systems - Benefits and Considerations in Proceedings of ESAV**, 2011 - September 12 - 14 Capri, Italy.

BAUD, O.; HONORE, N.; TAUPIN, O. **Radar/ADS-B Data Fusion Architecture for Experimentation Purpose**. Em Proceedings of the 9th International Conference on Information Fusion, Florence, Italy, 10–13 jul 2006; pp. 1–6.

Bootstrap. [site corporativo]. Disponível em <http://getbootstrap.com/>. Acesso em: 18 mai. 2014.

CAMPOS, B. F. **Estudo de caso da migração de um sistema legado para arquitetura orientada a serviços**. Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2013.

CARNEIRO, R.; CALLADO, A. **Google Maps API**, Minicurso na Escola de Férias 2012, Universidade Federal do Ceará, Campus de Quixadá, 1 a 5 de Outubro de 2012.

CARVALHO, S. A. L. **Tecnologia ADS-B no Linux, uma implementação de decodificação e persistência de mensagens**. Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2012.

Data Modeling & Profiling Tool: SQL Power Architect | SQL Power Software. [site corporativo]. Disponível em <http://www.sqlpower.ca/page/architect>. Acesso em: 20 mai. 2014.

Documentação de desenvolvedor do Google Maps. [site corporativo]. Disponível em <https://developers.google.com/maps/documentation/?hl=pt-br>. Acesso em: 31 mai. 2013.

Documentação do PotgreSQL 8.0.0. [site corporativo]. Disponível em <http://pgdocptbr.sourceforge.net/pg80/index.html>. Acesso em: 31 mai. 2013.

Flightradar24.com – Live flight tracker!. [site corporativo]. Disponível em <http://www.flightradar24.com/>. Acesso em: 1 jul. 2013.

Git [site corporativo]. Disponível em <http://git-scm.com/>. Acesso em: 31 mai. 2013.

Git Hub - Build software better, togetther. [site corporativo]. Disponível em <https://github.com/>. Acesso em: 31 mai. 2013.

HICOCK, D. S.; LEE, D. **Application of ADS-B for airport surface surveillance**, Em Proc. IEEE/AIAA 17th Digital Avionics Systems Conf., 1998.

Jersey. [site corporativo]. Disponível em <https://jersey.java.net/>. Acesso em: 19 mai. 2014.

LUBBERS, P.; GRECO, F. **HTML5 Web Sockets: A Quantum Leap in Scalability for the Web**, SOA World Magazine, Mar. 2010; Disponível em <http://soa.sys-con.com/node/1315473>. Acesso em: 19 mai. 2014.

Manifesto para o desenvolvimento ágil de software. [site corporativo]. Disponível em <http://manifestoagil.com.br/>. Acesso em: 31 mai. 2013.

Maven – What is Maven?. [site corporativo]. Disponível em <http://maven.apache.org/what-is-maven.html>. Acesso em: 20 mai. 2014.

MOHOROVICIC, S. **Implementing responsive web design for enhanced web presence**, 2013 - May 20 - 24 Opatija, Croatia.

MOREIRA NETO, O. **Entendendo e Dominando o Java para Internet**. São Paulo : Digerati Books, 2006.

MORO, T. D.; DORNELES, C. F.; REBONATTO, M. T. **Web services WS-* versus Web Services REST**. Em REIC - Revista de Iniciação Científica, 2011, volume 11, número 1, pp. 36-51.

O que é o Java e por que é necessário? [site corporativo]. Disponível em <http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em: 31 mai. 2013.

POWERS, S. **Aprendendo JavaScript**, 1ª edição, 2010.

PRESSMAN, R. S. **Engenharia de Software**, 6ª edição. 2006.

SBS-1 BaseStation Tutorial 4.2. [site corporativo]. Disponível em < http://www.homepages.mcb.net/bones/SBS/Article/Barebones42_Socket_Data.htm>. Acesso em: 20 mai. 2014.

Sidr – A jQuery plugin for creating side menus. [site corporativo]. Disponível em <http://www.berriart.com/sidr>. Acesso em: 18 mai. 2014.

SILVA, M. S. **JQuery – A Biblioteca do Programador JavaScript**, 2a edição, 2010.

SOMMERVILLE, I. **Engenharia de Software**, 6ª edição. 2003.

STANISCIA, G. F. **Gerenciamento de Tráfego Aéreo de Nova Geração**. ATECH Tecnologias Críticas. LAAD 2007 – Latin America Aero & Defence 07.

Transair – Flight Equipment. [site corporativo]. Disponível em <http://www.transair.co.uk/>. Acesso em: 19 mai. 2014.

WANG, P.; YANG, C.; YU, Z.; REN, Y. **A load balance algorithm for WMS**, Em Proc. of Geoscience and Remote Sensing Symposium 2004 (IGARSS'04), IEEE, Sep. 2004, pp. 2920-2922.

APÊNDICES

APÊNDICE A – Documento de Requisitos

Documento de Requisitos

Sistema de Controle e Monitoramento Aéreo

Versão 2.0

Histórico de Alterações

Data	Versão	Descrição	Autor
20/06/2013	1.0	Criação e entrada de dados no documento	Edywaldo Rhonan Otávio Araújo
03/10/2013	2.0	Documento transferido para outro template e finalizado	Edywaldo Rhonan

Sumário

1. Introdução

1.1 Finalidade

Este documento especifica os requisitos do Sistema de Controle e Monitoramento Aéreo a ser desenvolvido pelos bolsistas de iniciação científica que estão alocados nesse projeto. Este documento possui as informações necessárias para os desenvolvedores implementarem o sistema.

1.2 Escopo

Este documento realiza a elicitação de requisitos do Sistema de Controle e Monitoramento Aéreo, o mesmo é um sistema que monitora o espaço aéreo no intuito de controlar e minimizar os acidentes aéreos.

1.3 Definições, Acrônimos e Abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

1.3.1 Identificação dos requisitos

Por convenção, a referência a requisitos é feita através do nome da seção onde eles estão descritos, seguidos do identificador do requisito. Os requisitos devem ser identificados com um identificador único. A numeração inicia com o identificador [RF001] ou [RNF001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos.

1.3.2 Prioridades dos requisitos

Para estabelecer a prioridade dos requisitos, foram adotadas as denominações: essencial, importante e desejável.

- **Essencial**: é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos fundamentais para o funcionamento do sistema que tem a maior prioridade a ser implementado.

- **Importante**: é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.

- **Desejável**: é o requisito que não compromete as funcionalidades básicas do sistema. Requisitos desejáveis podem ser deixados para versões posteriores do sistema.

1.3.3 Antena ADS-B

São antenas que captam sinais que são transmitidos pelas aeronaves que estão em voo. Nesses sinais terá mensagens que contém informações sobre a aeronave como por exemplo: origem, destino, altitude entre outras informações.

2 Descrição Geral

O Sistema de Monitoramento e Controle Aéreo será dividido em duas partes: coletores e um servidor central.

- Coletores: têm a função de coletar dados das antenas ADS-B, em seguida alimentar um banco de dados local e mostrar em uma aplicação web seus dados coletados. Os coletores também irão alimentar um banco de dados central que irá receber todos os dados de todos os coletores.

- Servidor Central: tem a funções de receber os dados de cada coletor, realizar a fusão dos dados (dados repetidos, inconsistentes, etc.), alimentar um banco de dados central e em paralelo a isto fazer detecções de possíveis problemas (colisões, acidentes) para então de acordo com esses possíveis problemas emitir alertas para a aeronave. O servidor central também deverá disponibilizar uma interface web disponível para o público.

3 Requisitos Elicitados

3.1 Requisitos Funcionais

3.1.1 *RF01* - Os coletores devem receber os dados de todas as aeronaves acessíveis no momento.

Através das antenas ADS-B os coletores devem receber os dados de todas as aeronaves acessíveis no momento.

Prioridade: **➡> Essencial** Importante Desejável

3.1.2 *RF02* - Os coletores devem alimentar um banco de dados local.

Os coletores devem armazenar os dados obtidos através das antenas ADS-B em um banco de dados local.

Prioridade: **➡ > Essencial** Importante Desejável

3.1.3 *RF03* - O servidor central deve receber os dados de todos os coletores.

O servidor central recebe os dados que estão armazenados no banco de dados local dos coletores.

Prioridade: Essencial **➡> Importante** Desejável

3.1.4 *RF04* - O servidor central realiza fusão dos dados.

O servidor central ao captar os dados dos coletores, irá verificar se os dados estão repetidos ou inconsistentes e se necessário serão tratados.

Prioridade: Essencial **➡ > Importante** Desejável

3.1.5 *RF05* - O servidor central alimenta um banco de dados central.

Ao receber as informações dos coletores, o servidor central, após realizar a fusão dos dados, alimentará um banco de dados central.

Prioridade: **➡> Essencial** Importante Desejável

3.1.6 *RF06* - O servidor central deve possuir uma interface web para exibição dos dados.

No servidor terá uma aplicação web aberta para o público, onde será mostrado de uma forma amigável as informações coletadas no intuito de monitorar e controlar o espaço aéreo.

Prioridade: Essencial **➡ > Importante** Desejável

3.1.7 *RF07* - O servidor central deve emitir alertas para problemas eminentes.

Caso o servidor detecte alguma possível colisão ou erro de rota, o servidor deve emitir algum tipo de alertar para que seja corrigido o percurso e não tenha nenhum acidente.

Prioridade: Essencial Importante **➡ > Desejável**

3.2 Requisitos Não Funcionais

3.2.1 RFN001 - Robustez:

Não pode ocorrer perda de dados, por mais que haja algum defeito os coletores não podem deixar de enviar os dados.

3.2.2 RFN002 - Segurança

O sistema tem que realizar o controle e o monitoramento de forma precisa e os dados recebidos dos coletores tem que ser tratados com de forma segura e sigilosa.

3.2.3 RFN003 - Rápido

O sistema tem que realizar respostas rápidas para que o controle e o monitoramento seja o mais próximo possível do real, seja preciso.

3.2.4 RFN004 - Confiável

O sistema tem que realizar o controle e o monitoramento de forma que os usuários possam confiar que os dados por ele fornecido são os mais precisos possíveis.

3.2.5 RFN005 - Usabilidade

O sistema tem que ter uma interface simples e fácil de usar e o usuário não precise navegar por muitas telas para encontrar o que deseja.

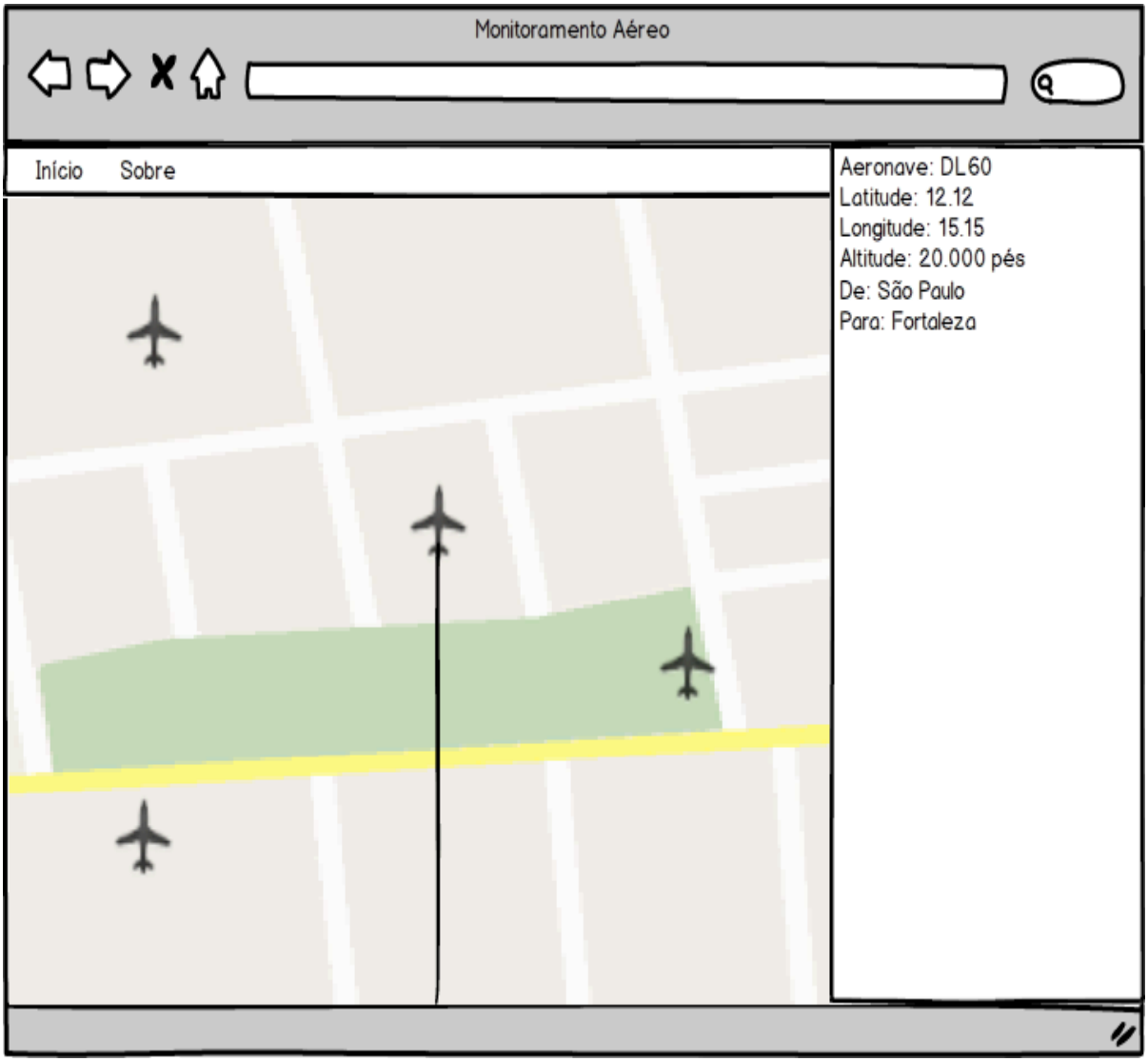
4 Informações de suporte

4.1 Protótipos de Tela

Os protótipos a seguir servem apenas como base para entendimento do sistema, provendo informações sobre suas características básicas. Desta forma, os mesmos serão descartados.

4.1.1 Tela inicial do sistema de Web

4.1.2 Tela quando o usuário acionar uma aeronave



4.1.3 Protótipo do sistema como um todo