

FACULDADE DE TECNOLOGIA DE JAHU

JÉSSICA FERNANDA GARCIA DE SOUZA

**DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA
AUTOMAÇÃO DE CHECKLIST DE VISTORIA DE EMBARCAÇÃO**

JAHU

JULHO DE 2018

JÉSSICA FERNANDA GARCIA DE SOUZA

**DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA
AUTOMAÇÃO DE CHECKLIST DE VISTORIA DE EMBARCAÇÃO.**

Trabalho de Conclusão de Curso
apresentado ao Curso de Tecnologia em
Sistemas Navais da Faculdade de
Tecnologia de Jahu como requisito a
obtenção do grau de Tecnólogo em
Sistemas Navais.

Orientador: Marcos Shoiti Saito
Coorientador: Ademir Marques Junior

JAHU
JULHO DE 2018

Dedico este trabalho a Deus que esteve
sempre ao meu lado, me dando sabedoria e fé.

“Entregue seus cinco pães e os dois
peixes e Ele concederá o teu crescimento.”

Marcos 6 38:44

AGRADECIMENTOS

Agradeço a Deus pela saúde e pela capacidade de concluir este curso superior, ao meu orientador Marcos Shoiti Saito, ao meu amigo e coorientador Ademir Marques Junior, a minha amiga de trabalho Sueli Rezende que se dedicou a total apoio para ajudar na formatação deste trabalho, a minha amizade de infância Luana Nascimento, pelo acompanhamento. Sou grata a todos pela participação da minha conclusão de curso, que sempre tiveram paciência, amizade e dedicação.

Ao meu setor de trabalho Secretaria Acadêmica, em especial a minha líder de trabalho Isabela Nassif Ortolani Mendonça, pela compreensão e amizade, a minha amiga de trabalho Ana Paula dos Santos Vasques por sempre me motivar com pensamentos positivos, ao meu amigo de trabalho Evandro Ravalho pela simpatia e por trazer animação no ambiente de trabalho.

A minha família, em especial aos meus avós Ana Salete e Isaias, pela força e carinho que sempre me deram, a minha irmã Gabriela, que sempre me fez acreditar que em meio as dificuldades são possíveis vencer.

Ao Curso de Tecnologia em Sistemas Navais, na pessoa de sua coordenadora Professora Rosa Maria Padroni, pelo apoio e dedicação recebido.

A todos os Professores do curso, especialmente ao Sr Professor Marcos Shoiti Saito, pelas contribuições e sugestões no trabalho.

Aos meus amigos e parceiros de sala do 2º semestre de 2014 que sempre estivemos juntos enfrentando as dificuldades e superando desafios. Em especial os que estiveram comigo no fim do semestre: Valdinéia Cristini (Crixval), Paraguay (lo chico gordíneo), Samuel (Japoronga), Aspira (Vítas), Giovanni (o International), Bruno (o chorão), João Cesar (Tererezeiro Nutella).

LISTA DE ABREVIATURAS E SIGLAS

2G – Segunda Geração

3G – Terceira Geração

4G – Quarta Geração

5G – Quinta Geração

AB – Arqueação Bruta

AG – Agência

AMPS – *Advanced Mobile Phone System*

API – *Application Programming Interface*

APP – *Application*

BL – Borda Livre

BV – *Bureau Veritas*

CDMA – *Code Division Multiple Access*

COLREG – *Convention on the International Regulations for Preventing Collisions at Sea, 1972.*

CP – Capitania

CSN – Certificado de Segurança da Navegação

DL – Delegacia

DPC – Diretoria de Portos e Costas

EDGE – *Enhanced Data Rates*

GPRS – *General Packet Radio Service*

GRU – Guia de Recolhimento da União

GSM – *Global System Mobile*

HSPA - *High Speed Packet Access*

IBGE - Instituto Brasileiro de Geografia e Estatística

IBM – *International Business Machines*

IDE – *Integrated Development Environment*

IMO – *International Maritime Organization*

LESTA – Lei de Segurança do Tráfego Aquaviário

LR – *Lloyd's Register*

LTE – *Long Term Evolution*

MARPOL – *Prevention of Pollution from Ships*

MySQL – *My Structured Query Language*
NORMAM – Normas da Autoridade Marítima
NORTEC – Normas Técnicas da Autoridade Marítima
NPCP – Normas e Procedimentos das Capitánias
ONU – Organização das Nações Unidas
OS – *Operational System*
RLESTA – Regulamento da Lei de Segurança do Tráfego Aquaviário
SMS – *Short Message Service*
SOA – Sistema Operacional Android
SOLAS – *Safety of Life at Sea*
SQLite – *Structured Query Language*
STCW – *Standards of Training Certification and Watchkeeping for Seafarers*
TDMA – *Time Division Multiple Access*
TM – Tribunal Marítimo
UI – *User Interface*
UML – *Unified Modelling Language*
UMTS - *Universal Mobile Telecommunications System*
UX – *User Experience*
WCDMA – *Wide-Band Code-Divison Multiple Access*
XML – *Extensible Markup Language*

RESUMO

Com a crescente popularização e avanço tecnológico dos dispositivos móveis, o smartphone está cada vez mais presente no cotidiano de todos, por suas características como a portabilidade, conectividade e usabilidade, e funcionalidades que ajudam seus usuários, e, portanto, se tornam indispensáveis na vida de milhões de pessoas. Pensando nisso, a proposta deste trabalho é a utilização de um smartphone como ferramenta, de forma a auxiliar o vistoriador naval em sua vistoria nas embarcações, por se tratar de uma atividade bastante complexa e estar diretamente fundamentada em normas, leis e regimentos. Após um minucioso estudo das funções envolvidas, surge a ideia de desenvolver um aplicativo móvel específico para vistoria de borda livre, com o sistema *Android*, que poderá facilitar e agilizar sua tarefa de forma prática e sempre atualizada, mas principalmente, por não haver nenhum outro no mercado que atenda essa necessidade. O aplicativo desenvolvido atendeu o objetivo esperado para facilitar o trabalho do vistoriador naval, possui uma aplicação simples e funcional, faz referências as normas aplicáveis e gera um relatório final que facilita o acompanhamento dos requisitos que necessitem de alterações, como também, terá acesso a todas as embarcações cadastradas no aplicativo para consulta.

Palavras chave: Borda Livre, vistoria, vistoriador naval, Android, aplicação.

ABSTRACT

With the growing popularization and technological advancement of mobile devices, the smartphone is increasingly present in our daily life, due to its characteristics such as portability, connectivity, usability and features that help its users, and, therefore, become indispensable in the life of millions of people. In this regard, the proposal of this work is the use of a smartphone as a tool, to assist the naval surveyor in their inspection on the vessels, since it is a very complex activity and is directly based on norms, laws and regulations. After a thorough study of the involved functions, it was awakened the idea of developing a specific mobile app to a free boarder review for Android that can facilitate and streamline its task in a practical and always updated way, but mainly, because there is no other in the market that meets this need. The developed application met the expected objective to help the work of the naval surveyor, with a simple and functional application that references the applicable regulations and generates a final report that facilitates the monitoring of requirements that require corrections, as well as access to all the vessels registered in the application for further consultation.

Keywords: *Freeboard, inspection, naval surveyor, Android, application.*

LISTA DE FIGURAS

Figura 1 - Representação de Borda livre.....	23
Figura 2 - Disco de Plimsoll.....	24
Figura 3 – Disco de Plimsoll e especificações de Borda Livre.	24
Figura 4 - Ambiente de desenvolvimento Android Studio.....	32
Figura 5 - Acompanhamento da vistoria à embarcação.	34
Figura 6 – Medição suspiro do convés principal.....	36
Figura 7 - Medição de borda livre.....	36
Figura 8 - Abertura dos porões.....	37
Figura 9 - Disco de Plimsoll e marca da linha do convés	37
Figura 10 -Porta de visitas.....	38
Figura 11 -Balaustrada.	38
Figura 12 - Diagrama de caso de uso.	40
Figura 13 - Diagrama de atividades para a atividade principal da aplicação.....	41
Figura 14 - Diagrama de atividade para a criação de novas vistorias.	42
Figura 15 -Diagrama para a atividade de checklist.	42
Figura 16 - Diagrama de classes inicial simplificado.	43
Figura 17 - Diagrama de banco dados.	44
Figura 18 - Web APP Marvel para a criação e desenvolvimento de wireframes.	45
Figura 19 -Rascunho das telas e simulação de interação do usuário.	46
Figura 20 - Diagrama de classes gerado a partir da aplicação no Android Studio. ...	47
Figura 21 - Tela inicial para cadastro e tela das vistorias salvas.....	48
Figura 22 - Adicionando uma nova vistoria.	49
Figura 23 - Exibição do checklist para uma vistoria.	50
Figura 24 -Tela solicitando a entrada das observações sobre itens não marcados. .	51
Figura 25 - Tela questionando o usuário o próximo passo após criado o relatório. ..	52
Figura 26 - Estrutura de pastas do projeto no Android Studio.....	53

LISTA DE TABELAS

Tabela 1 - Evolução das tecnologias de comunicação e aparelhos.	30
Tabela 2 - Versões da plataforma Android.....	33
Tabela 3 - Detalhamento técnico da embarcação vistoriada.....	35

LISTA DE QUADROS

Quadro 1 - Itens em um checklist de borda livre.	26
---	----

SUMÁRIO

1 INTRODUÇÃO	15
1.1 Objetivos.....	16
1.2 Objetivos específicos	17
1.3 Justificativa	17
2 CERTIFICAÇÃO NAVAL E VISTORIA.....	18
2.1 Convenções, Normas e Regulamentos Marítimos	18
2.2 Regulamentação	19
2.3 Documentações Estatutárias	20
2.4 Tipos de Vistorias	21
2.5 Vistorias em embarcações	22
2.5.1 Definições e Requisitos Técnicos	22
2.5.2 Borda Livre	22
2.6 Vistoriador Naval.....	25
2.6.1 Procedimentos preliminares para a realização da Vistoria	25
2.6.2 Vistoria Anual de Borda Livre	26
2.7 Aplicativo para checklist de vistoria	27
3 TELEFONIA MÓVEL E <i>SMARTPHONE</i>	29
3.1 História da telefonia móvel.....	29
3.2 Sistema Operacional Linux	30
3.3 Plataforma Android.....	31
3.4 Versões do <i>Android</i>.....	32
4 METODOLOGIA.....	34
4.1 Estudo de caso – vistoria naval.....	34
4.1.1 Resultado da vistoria	39
4.2 Levantamentos de requisitos.....	39
4.3 Modelagem do <i>software</i>/aplicativo	40
4.3.1 Diagramas UML	40
4.3.2 Diagramas de Caso de Uso	40
4.3.3 Diagramas de Atividade	41
4.3.4 Diagrama de Classe	43
4.3.5 Diagrama de banco de dados	43

4.4 Desenvolvimento de UI (interface do usuário)	44
4.5 Desenvolvimento no Android Studio	46
5 CONCLUSÕES, CONSIDERAÇÕES E PROPOSTAS PARA TRABALHOS FUTUROS	54
BIBLIOGRAFIA	56
APÊNDICE	59

1 INTRODUÇÃO

Conforme o item 0201 do Capítulo 2 das Normas de Autoridade Marítima 02 (NORMAM-02/DPC, 2018) é determinado que toda embarcação brasileira deve ser inscrita e/ou registrada na Marinha do Brasil, segundo sua classificação e as normas vigentes, sendo que, quando se tratar de embarcação com Arqueação Bruta (AB) maior de 100 deverá também, ser inscritas no TM (Tribunal Marítimo) e as plataformas fixas ou móveis são consideradas embarcações e necessitam de inscrição e/ou registro.

Todo e qualquer flutuante (cais, hotéis, posto de combustível, casas, ou outra estrutura flutuante similar) está sujeito as disposições do Capítulo 1 (NORMAM-11/DPC) e as embarcações miúdas (comprimento total menor que 5 metros e inferior a 8 metros) com propulsão (movimentada por meio de máquinas ou motores) se emite inscrição simplificadas, enquanto que, as sem propulsão ficam dispensadas.

Quanto a regularização de embarcações, o item 0300 do Capítulo 3, tem como propósito, “Estabelecer procedimentos para enquadrar embarcações, construídas no Brasil ou no exterior para a bandeira brasileira, nos diversos processos de legalização de projetos” (MARINHA DO BRASIL, DIRETORIA DE PORTOS E COSTAS, 2018).

Com relação a obrigatoriedade do Certificado Nacional de Borda Livre para navegação interior, base deste estudo, esclarece a NORMAM-02/DPC, 2018

As embarcações que não sejam dispensadas de atribuição de borda-livre, conforme estabelecido no item 0601, deverão ser portadoras de um Certificado Nacional de Borda-Livre para a Navegação Interior, cujo modelo é apresentado no Anexo 6-A, doravante denominado Certificado. As embarcações cuja borda-livre tenha sido determinada utilizando-se o procedimento alternativo constante dos Anexos 6-K e 6-O estão dispensadas de possuir esse Certificado.

Foi acompanhado a vistoria realizada em um areeiro e conforme citado acima (Anexo 6-K), estabelece que neste tipo de embarcação o procedimento alternativo para determinação da borda-livre das embarcações empregadas no transporte de areia, poderá ser substituído pelo Capítulo 6 que especifica,

[...] para a atribuição de uma borda-livre de embarcações com convés de borda-livre descontínuo (conforme estabelecido no Item 0603 a) 3) empregadas exclusivamente no transporte de areia. Tal procedimento, que consiste na verificação de uma reserva de fluabilidade equivalente, é recomendado para aquelas embarcações que, por questões operacionais, sejam projetadas para navegarem com calado maior ou igual àquele correspondente ao convés de borda-livre. (NORMAM-02/DPC, 2018)

As vistorias periódicas realizadas pelos órgãos autorizados da Marinha do Brasil devem realizar perícia na embarcação, a cada cinco anos após a vistoria inicial, que trata do setor casco (intermediárias ou renovação) referentes ao Certificado de Segurança da Navegação (CSN), que comprove a manutenção de condições de estanqueidade das estruturas e equipamentos constantes das normas.

O Vistoriador Naval possuiu uma gama de especificações para a vistoria da embarcação e até hoje realiza-as por meio de fotos e *checklist* impresso. Através do uso de um aplicativo próprio, terá acesso ao conteúdo virtual, consultar as normas ali aplicáveis, seguirá um roteiro, e ao mesmo tempo, dará baixa aos itens já verificados de forma prática e rápida.

Nesse contexto, este estudo compreende a grande utilidade deste recurso e para desenvolver um aplicativo móvel será essencial o controle e execução por meio de automação de *checklist* de vistoria da embarcação e se abster do modo manuscrito. Para realização desta proposta, optou-se pelo Sistema Operacional Android (SOA) por se tratar do sistema operacional mais utilizado no mundo, segundo a matéria de Carvalho para o Olhar Digital (2017) explica,

O Android é o sistema operacional mais popular do mundo, e mantém essa liderança também no Brasil. De acordo com a empresa de análise de dados Kantar, o sistema do Google já domina 93,2% do mercado nacional de dispositivos móveis. O número é referente à análise conduzida entre dezembro de 2016, janeiro e fevereiro de 2017 no país. No mesmo período, um ano atrás, o Android já dominava 90,4% do mercado. (CARVALHO, 2017).

1.1 Objetivos

O objetivo deste trabalho é criar um aplicativo móvel (APP) que seja uma ferramenta essencial durante as vistorias realizadas nas embarcações, com o propósito de facilitar e minimizar o processo e, que garanta uma atualização constante, segundo as normas, leis, regulamentos etc.

1.2 Objetivos específicos

Este estudo busca realizar um levantamento quanto à atividade exercida pelo vistoriador e regulamentação que rege, como também, procurar soluções disponíveis no mercado de forma a aperfeiçoar uma solução existente ou desenvolver um aplicativo móvel que se aplique na função.

1.3 Justificativa

Diante de uma atividade fundamentada em normas específicas para os mais variados tipos de vistorias, que em sua maioria envolvem inúmeros itens correspondentes ao serviço aplicável, torna-se relevante determinar por que razões e importância da interação e comunicação com *softwares* que são desenvolvidos, a fim de facilitar qualquer processo para o dia a dia.

Esse trabalho apresenta uma nova alternativa para o vistoriador naval, que pode reduzir tempo e melhorar sua produtividade. Atualmente a atividade se faz de forma manual, através de formulários, prancheta e caneta esferográfica. A utilização de um APP pode facilitar o serviço, principalmente em locais confinados e proporciona uma melhor percepção visual de forma mais atrativa e segura, como também poderá utilizar-se do smartphone como: lanterna, fazer registros através da câmera fotográfica, se necessário, solicitar uma comunicação externa rápida (celular), em ambientes que ofereçam riscos à atividade etc.

2 CERTIFICAÇÃO NAVAL E VISTORIA

Este capítulo detalha o processo de regularização e vistoria das embarcações, conforme a Diretoria de Portos e Costa (DPC) com resolução da Normas da Autoridade Marítima (NORMAM) e nas Convenções e Códigos Internacionais, onde se aplicarem. Vistoria se trata da definição segundo a Lei de Segurança do Tráfego Aquaviário (LESTA),

XXI - Vistoria - ação técnico-administrativa, eventual ou periódica, pela qual é verificado o cumprimento de requisitos estabelecidos em normas nacionais e internacionais, referentes à prevenção da poluição ambiental e às condições de segurança e habitabilidade de embarcações e plataformas.

2.1 Convenções, Normas e Regulamentos Marítimos

Os navios de longo curso são regidos por uma série de Convenções Internacionais, por leis e/ou regras específicas de cada país. A *International Maritime Organization* (IMO) trata de convenções internacionais, órgão pertencente a Organização das Nações Unidas (ONU) com o principal objetivo, de garantir uma operação segura das embarcações. Ainda se aplicam outras regras Nacionais e Internacionais:

Legislação Nacional:

- LESTA e RLESTA (Segurança do Tráfego Aquaviário);
- Lei do Óleo (Prevenção da Poluição);
- Lei nº97/1999 (Autoridade Marítima);
- NORMAM (Normas de Autoridade Marítima);
- NORTEC (Normas Técnicas da Autoridade Marítima);
- NPCP (Normas e Procedimentos das Capitânias).

Legislação Internacional – Convenções:

- SOLAS (Salvaguarda da Vida Humana no Mar);
- MARPOL (Prevenção de Poluição por Navios);
- LOAD LINE (Linhas de Carga);

- STCW (Instrução e Certificação para Marítimos);
- COLREG (RIPEAM).

Cada país possui seus órgãos reguladores e regras adicionais, no Brasil, a Marinha do Brasil é responsável pelo cumprimento das leis e regras vigentes no setor naval. Compete ainda aos órgãos responsáveis, fiscalizar se o projeto de construção e operação de navios cumprem com as regulamentações. O cumprimento destas normas garante a segurança do transporte aquaviário e a prevenção da poluição ambiental no ambiente naval.

A Autoridade Marítima Brasileira se fundamenta pela NORMAM, sendo que a NORMAM-02/DPC reconhece as Sociedades Classificadoras e Entidades Certificadoras que representam o país.

2.2 Regulamentação

As embarcações classificadas pela Autoridade Marítima, quanto ao tipo de navegação, a atividade ou serviço, consideram também o tipo de propulsão, onde periodicamente passam por vistorias e, o vistoriador naval fiscaliza e verifica se as mesmas cumprem as leis, normas e regulamentos dela decorrentes, também das resoluções internacionais validadas pelo Brasil, a fim de garantir a navegação segura e a salvaguarda da vida humana, seja em mar aberto ou hidrovias, não só pelas embarcações, mas, de plataformas fixas ou instalações de apoio.

As Capitania dos Portos (CP), Delegacias (DL) e Agências (AG) são responsáveis pela inscrição, documentação e normalização das embarcações dependendo do porte da mesma, baseada na Arqueação Bruta (AB), como também, possuir AB menor ou igual a 100. Estas inscrições/registros pertencem a jurisdição ao qual o proprietário ou armador reside ou onde for navegar conforme especifica o item 0203 do Capítulo 2, (NORMAM-02/DPC, 2018).

A vistoria executada pelo representante da certificadora, juntamente com a autora, foi realizada em um areeiro com propulsão própria e AB maior que 100, cuja atividade é explorar a hidrovia na extração de areia e segundo o item 0216 do Capítulo 2 (NORMAM-02/DPC, 2018) classifica como: navegação interior e a atividade/serviço de dragagem.

2.3 Documentações Estatutárias

Segundo consta no item 0801 do capítulo 8, (NORMAM-02/DPC, 2018) a obrigatoriedade do CSN, conforme explica a RECORD-Certificação Naval (2018), se aplicam

As embarcações que se enquadrem em quaisquer das situações listadas e definida na NORMAM 01 e/ou 02, independentemente de sua classificação, estão sujeitas a vistorias iniciais, intermediárias, anuais e de renovação e deverão portar este certificado. (RECORD, 2018)

O item 0601 do Capítulo 6 (NORMAM-02/DPC, 2018) especifica como são realizados os cálculos e quais embarcações são enquadradas à vistoria anual de borda livre, assunto relacionado a esta pesquisa de campo, a empresa RECORD, Certificação Naval (2018) esclarece que,

As embarcações que não sejam dispensadas de atribuição de borda-livre, conforme estabelecido na NORMAM 01 e/ou 02, deverão ser portadoras de um Certificado Nacional de Borda-Livre. Será expedido um Certificado da Borda-Livre às embarcações que apresentarem os cálculos de estabilidade de acordo com as regras da NORMAM pertinente. É no corpo deste certificado que é apresentado o posicionamento do disco de Plimsoll, a distância vertical, na meia-nau, entre a aresta superior da linha do convés e a aresta superior da linha horizontal da marca de borda-livre. (RECORD, 2018).

No que se refere a arqueação bruta da embarcação que trata no item 0701 do Capítulo 7 da NORMAM, justifica

Será expedido um Certificado da Arqueação a toda embarcação cujas arqueações bruta e líquida tenham sido determinadas conforme as disposições do Regulamento da NORMAM específica a cada tipo de embarcação. (NORMAM-02/DPC, 2018)

2.4 Tipos de Vistorias

Segundo o item 0803 do Capítulo 8 da NORMAM-02/DPC (2018) classifica os tipos de vistorias: Inicial; Anual, Intermediária e de Renovação; Documentação para Requerer Vistorias Especiais.

- Vistoria Inicial: realizada (a embarcação em seco e flutuando) durante e/ou após a construção, ou qualquer alteração que ocorra na embarcação, de acordo com a expedição do CSN.
- Vistoria Anual: realizada para endosso do CSN, não necessita a docagem;
- Vistoria Intermediária: o vistoriador deverá fazer a análise da medição de espessuras do chapeamento do casco, fundo, convés principal e anteparas estanques, que conforme o relatório assinado pelo profissional, o documento deverá apresentar condições aptas a resistências estrutural. É realizada para endosso do CSN e não necessita de docagem da embarcação;
- Vistoria de Renovação: Efetuada para a renovação do CSN, sendo vistoriada a embarcação flutuando e a seco.

A documentação para requerer as vistorias descritas acima, o interessado deverá apresentar a solicitação, cópia do CSN, Guia de Recolhimento da União (GRU), do comprovante de pagamento, referente aos serviços de Vistoria Anual, Intermediária e Renovação (seco e flutuando), com exceção a órgãos públicos.

Em se tratando de uma vistoria especial para regular uma embarcação sujeita a vistoria estatutária ou classe, de prova de máquina/navegação, é necessária uma execução de testes e verificação antes da conclusão da vistoria, para a emissão, renovação e endosso de certificados, o Certificado Nacional de Borda Livre, se enquadra conforme Capítulo 6 NORMAM-02/DPC, 2018.

Quando necessária a emissão de um Laudo Pericial, o vistoriador responsável deverá medir todos os parâmetros necessários para o cálculo de arqueação bruta, o volume existente acima do convés e as características principais, incluindo líquida para a emissão do Certificado Nacional de Arqueação.

2.5 Vistorias em embarcações

A Vistoria Naval é uma Perícia Técnica realizada na embarcação, acordada entre o armador ou seu representante e o agente da Autoridade Marítima, onde se verifica o cumprimento dos requisitos prescritos na NORMAM, nas Convenções, Códigos e Portarias.

2.5.1 Definições e Requisitos Técnicos

As embarcações são classificadas pela Marinha do Brasil e se enquadram de acordo as suas especificações. Em se tratando das vistorias de borda-livre (BL), segundo a NORMAM-02 as embarcações estão isentas da atribuição desse certificado, se apresentarem:

1) AB menor ou igual a 50; 2) Comprimento de regra (L) inferior a 20 m; 3) Embarcações destinadas exclusivamente a esporte e/ou recreio e comprimento menor que 24 m; e 4) Navios de guerra. (NORMAM-02/DPC, 2018).

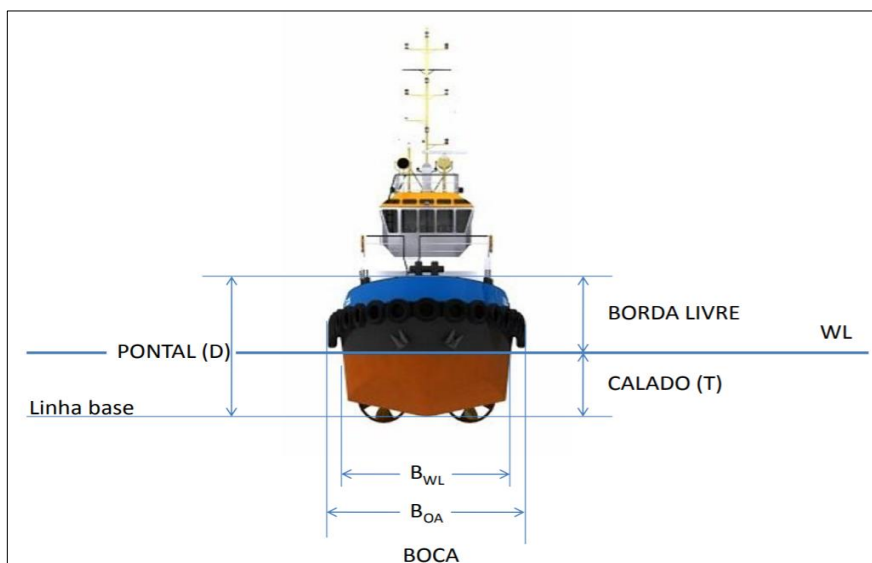
2.5.2 Borda Livre

Definição: A borda livre é a distância vertical, na meia-nau, localizada no meio do comprimento de regra entre a aresta superior da linha de convés e a aresta superior da linha horizontal da marca de borda-livre, é o resultado da altura do pontal menos o valor do calado (Fig.1).

Através de cálculos se estabelece uma linha limite para o carregamento da embarcação, ou seja, uma linha que é a distância vertical entre o plano de flutuação para a imersão máxima permitida.

Segundo ASSI (2011), “A borda-livre determina o peso máximo que o navio pode alcançar, ou seja, o seu deslocamento máximo. Em outras palavras, a borda livre fixa qual a reserva de flutuabilidade mínima permitida ao navio, em uma dada situação”.

Figura 1 - Representação de Borda livre.

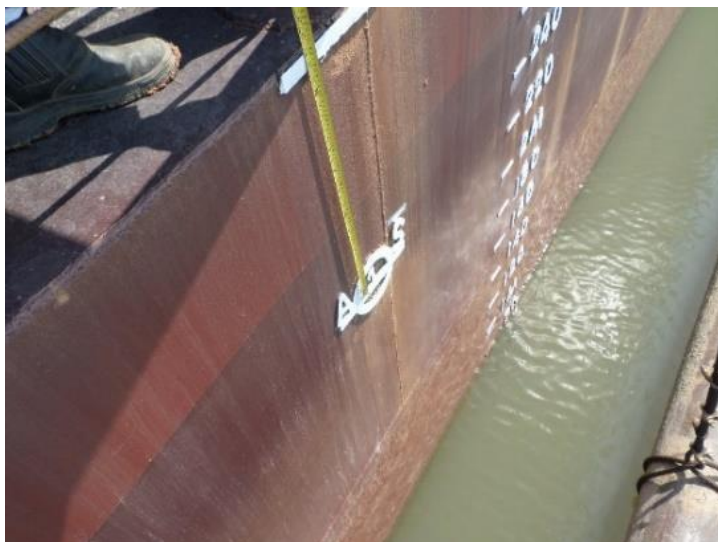


Fonte: ASSI (2011).

Inicialmente, foram propostas as bases para convenção por Samuel Plimsoll (1873/76) no parlamento inglês, para garantir a segurança dos navios quanto ao limite de carga e flutuabilidade. Esta marca foi adotada (1894) pela *Load Line Convention* (1930) e pela *International Convention on Load Lines* (1966), porém a convenção internacional foi implantada em 1968 com modificações ao longo dos anos aliadas a outras convenções de linha de carga como Marpol e Solas (1973/78).

A marca de Plimsoll é representada por uma circunferência cortada ao meio por uma reta, esta marca deverá ser pintada no costado (que representa o revestimento do casco acima do bojo) conforme apresenta a figura 2.

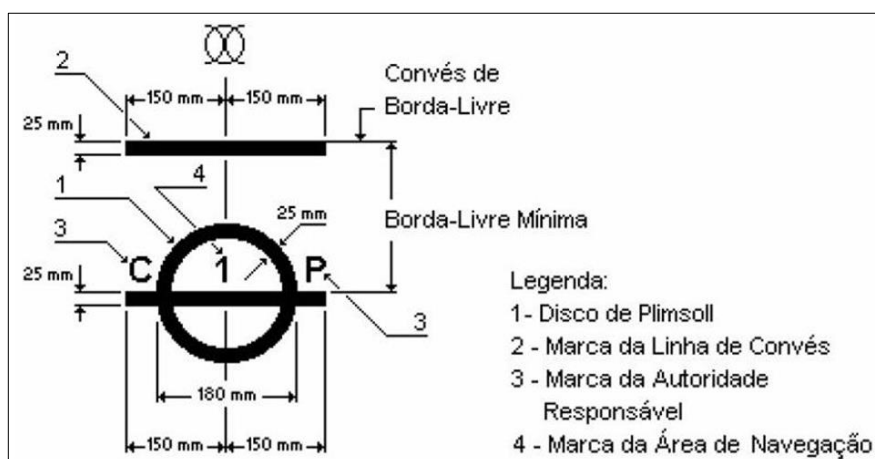
Figura 2 - Disco de Plimsoll.



Fonte: Autora.

Conforme a classificação da embarcação, a linha de carga (BL) deve ficar ao lado do disco de Plimsoll, nos costados a bombordo (lado esquerdo) e a boreste (lado direito) que limitam os calados máximos e se divide em zonas conforme a área em que se navega, como zonas de: verão (V), tropical (T), inverno (I), e também em regiões de água salgada (dens. 1,025 t/m³), água doce e água doce tropical (dens. 1,0 t/m³) conforme apresenta a figura 3. As companhias de Seguro, também utilizam essas linhas, e caso haja irregularidades, correm o risco de não cobrir os prejuízos em caso de algum sinistro do casco e/ou carga.

Figura 3 – Disco de Plimsoll e especificações de Borda Livre.



Fonte: NORMAM 02, Cap.6 (2018).

2.6 Vistoriador Naval

O vistoriador é o representante legal das certificadoras, podem ser oficiais da Marinha do Brasil ou civis contratados e aprovados em curso para a formação de vistoriadores navais. Durante sua atividade faz uso de equipamento de segurança para a realizar uma vistoria, como também, é necessária a utilização de alguns materiais como: trena, lanterna, caneta, prancheta, luva, etc., o que dificulta seu trabalho, quanto a utilização de corrimão, acesso a porões, anotações em ambiente pouco iluminados, dentre outros, isto justifica o uso de um smartphone com o aplicativo de vistoria, facilitando muito a execução do seu trabalho, para estar com as mãos livres quando se desloca, inclusive.

2.6.1 Procedimentos preliminares para a realização da Vistoria

Existe alguns procedimentos que devem ser cumpridos, as vistorias deverão ser realizadas em portos ou em áreas abrigadas, estando a embarcação fundeada ou atracada, assim, deverá seguir o *checklist* segundo a NORMAM, 2018. A vistoria deverá ser realizada preferencialmente no período diurno, por um Vistoriador Naval que primeiramente faz a verificação da área externa antes de entrar a bordo, para então verificar documentação (certificados estatutários, de classe e serviços periódicos).

O responsável pela embarcação deverá arcar com todas as despesas, dará total assistência para facilitar as tarefas e consultas que o vistoriador necessite, assim como, os instrumentos, aparelhos, manuais, laudos periciais, protocolos e demais elementos que venham a ser solicitados. O vistoriador poderá cancelar a vistoria caso: a embarcação não estiver devidamente preparada para a vistoria; os acessos à embarcação sejam inadequados e/ou inseguros; ou qualquer outra circunstância que limite a eficácia da vistoria.

2.6.2 Vistoria Anual de Borda Livre

Neste trabalho foi aplicado o processo de vistoria anual que se realiza para o endosso do CSN, de acordo com o capítulo 6 da NORMAM-02, para navegação interior através de um *checklist* para acompanhamento dos itens a serem verificados, não sendo necessária a docagem da embarcação. Para a realização de uma vistoria requer alguns procedimentos que as empresas adotam para facilitar o trabalho de verificação dos itens. O modelo apresentado no Quadro 1 pertence a uma certificadora que atende a região.

Quadro 1 - Itens em um *checklist* de borda livre.

REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0631 alínea C) sub alínea3)
1. Verificar visualmente a embarcação para assegurar se não foram feitas alterações no casco e/ou nas superestruturas que possam alterar a borda livre anteriormente atribuída.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0631 alínea C) sub alínea3) I
1. Verificar o estado de conservação e a manutenção da proteção de aberturas das condições de estanqueidade dos(s) porão(ões) de carga.
2. Verificar o estado de conservação e a manutenção da proteção de aberturas das condições de estanqueidade dos escotilhões de acesso abaixo do convés de borda livre. O fechamento de um escotilhão deverá ser necessariamente efetuado por intermédio de tampas com atracadores permanentemente fixados.
3. Verificar o estado de conservação e manutenção da proteção de aberturas das condições de estanqueidade da(s) porta(s) de visita.
4. Verificar o estado de conservação e a manutenção da proteção de aberturas das condições de estanqueidade das vigias e olhos de boi existentes nos costados abaixo do convés de borda livre.
5. Verificar o estado de conservação e a manutenção da proteção de aberturas das condições de estanqueidade dos albos para iluminação e/ou ventilação natural.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0631 alínea c) sub alínea 3) II)
1. Verificar o estado de conservação das balaustradas e das bordas falsas das partes expostas dos conveses de borda livre e de superestrutura.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0631 alínea c) sub alínea 3) III)
1. Verificar se as saídas d'água se encontram permanentemente desobstruídas.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0631 alínea c) sub alínea 3) IV
1. Verificar se a posição das marcas de borda livre se encontra de acordo com o estabelecido no Certificado Nacional de Borda Livre em vigor.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea g) sub alínea 1)
1. Verificar a existência de uma passagem permanentemente desobstruída de proa a popa da embarcação, a qual não poderá ser efetivada por cima de tampas e escotilhas.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0625 alínea a)
1. Verificar se as marcas de borda livre estão permanentemente fixadas em ambos os bordos da embarcação sendo que para embarcação de aço, as marcas devem ser soldados ou buriladas de forma permanente.

REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0625 alínea b)
1. Verificar se as marcas estão pintadas em branco ou amarelo quando fixadas em fundo escuro ou em preto com fundo claro.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0625 alínea c)
1. Verificar se todas as marcas de borda livre estão facilmente visíveis ou se necessário arranjo especial pode ser feito com este propósito.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea d) sub alínea 1) a) b)
1. Verificar se o(s) suspiro(s) localizado(s) acima do convés de borda livre é em forma de U invertido e possuam, no mínimo, uma altura de 450 mm, medidos da aresta inferior do suspiro até o convés de borda livre.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea e) sub alínea c)
1. Verificar se as soleiras de portas, braçolas de escotilhas, escotilhões ou qualquer elemento que de acesso ao interior do casco possuam altura mínima de 150 mm (quando em ÁREA 1).
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0612 alínea f) sub alínea 3) c)
1. Verificar se as soleiras de portas, braçolas de escotilhas, escotilhões ou qualquer elemento que de acesso ao interior do casco possuam altura mínima de 260 mm (quando em ÁREA 2).
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea b) sub alínea 4)
1. Verificar a condição de estanqueidade das anteparas estanques e se as portas de visita, nelas instaladas, estão devidamente aparafusadas e em condições estanques.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0603 alínea x)
1. Verificar a bordo o ponto de alagamento, caso a mesma o possua, analisar se o mesmo está de acordo com o informado no Estudo de Estabilidade e representado no Plano de Arranjo Geral.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea b) sub alínea 2)
1. Verificar se as tampas das aberturas de escotilha, dos escotilhões e seus receptivos dispositivos de fechamento, quando existentes, tem resistência suficiente que permita satisfazer as condições de estanqueidade previstas para o tipo de embarcação considerada e deverão apresentar todos os elementos necessários para assegurar a estanqueidade.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0612 alínea c) sub alínea 3)
1. Verificar se as tampas das aberturas de escotilha, dos escotilhões e seus receptivos dispositivos de fechamento, quando existentes, tem resistência suficiente que permita satisfazer as condições de estanqueidade previstas para o tipo de embarcação considerada e deverão apresentar todos os elementos necessários para assegurar a estanqueidade.
REFERÊNCIA: NORMAM 02/DPC Capítulo 6 item 0611 alínea g) sub alínea 4)
1. Verificar se a altura da(s) balaustrada(s) e/ou borda(s) falsa(s) são > ou = a 1,00 m e também verificar se a abertura inferior da(s) balaustradas(s) apresentam altura < ou = 230 mm e se os demais vãos da(s) balaustrada(s) apresentam altura < ou = 380 mm.

Fonte: NORMAM (2018).

2.7 Aplicativo para checklist de vistoria

Embora não exista nenhum aplicativo aberto ou gratuito para essa finalidade, MEILI (2013) criou um software para desenvolvimento de projetos (Java) baseados

na NORMAM 02 e aplicável para a web, mostrando que há espaço para desenvolvimento de aplicações. Em se tratando de aplicativos, necessita que se aborde os smartphones e aplicativos, assunto do próximo capítulo.

3 TELEFONIA MÓVEL E *SMARTPHONE*

Nesta etapa se apresenta uma abordagem dos conceitos e história da telefonia móvel, *smartphone*, sistema *Android* e desenvolvimento de aplicativo móvel.

3.1 História da telefonia móvel

No ano de 1983, após décadas de estudos e pesquisas, o primeiro aparelho celular comercial do mundo, o DynaTAC 8000X entrou no mercado com sua massificação. É no contexto da difusão da Internet no final do século XX, que o primeiro celular considerado *smartphone* apareceu no mundo, o Simon. No ano de 1992, que por sua vez desenvolvido pela Máquinas de Negócios Internacionais (IBM), um aparelho que possuía uma tela *touchscreen* (sensível ao toque), foi algo extremamente revolucionário na época por permitir ao usuário a receber e enviar mensagens de *fax* e *e-mails*.

O *smartphone* anunciado e utilizado no ano 2000, pela Ericsson, com o lançamento da nova inovação do celular R380. Com o alto número de vendas, um ano antes, o *smartphone* Nokia 9000 Communicator, foi eleito por sua excelência até o ano 2011, embalado pelo seu Sistema Operacional da Nokia (OS), foi líder de mercado dentre os demais sistemas operacionais, mesmo assim, perdendo a liderança, para o recém lançado da Google, o Android.




Marcando início de uma nova era, no ano de 2007, a Apple revolucionou o mercado mundial com o lançamento da tendência do primeiro *smartphone*, o Iphone, uma inovação de formatos e aplicações que perdura até hoje. Vale ressaltar, um assunto notório que, os aparelhos celulares se popularizam e se espalharam pelo Brasil, com uma quantidade de milhões de *smartphone* de vários modelos, marcas e formas.

Segundo XAVIER (2006), explica que a partir da primeira geração ou 1G se utilizava a tecnologia *Advanced Mobile Phone System* (AMPS) que é um sistema analógico de comunicação, na segunda geração, surge o sistema digital de

comunicação, o *Code Division Multiple Access* (CDMA), o *Time Division Multiple Access* (TDMA) e *Global System Mobile* (GSM).

Com a implementação das redes de segunda geração com comunicação digital, possui envio de mensagens *Short Message Service* (SMS) e acesso à internet. A capacidade de envio e transmissão de dados servem de suporte para a evolução das funcionalidades dos aparelhos celulares e smartphones conforme apresenta a Tabela 1. Atualmente, encontra-se em expansão as redes 5G (LIMA, THEODORO, DANTAS, 2014) ou, de quinta geração, enquanto no Brasil, ainda atua a rede 4G (MORAES. 2015).

Tabela 1 - Evolução das tecnologias de comunicação e aparelhos.

Geração	2G			3G			4G	5G
Aparelhos destaques de cada época								
Tecnologia	GSM	GPRS	EDGE	WCDMA (UMTS)	HSPA	HSPA+	LTE	-
Taxa de dados para usuário	10-40 Kbps	40-50 Kbps	100-130 Kbps	128-384 Kbps	0,3-1 Mbps	3-6 Mbps	5-12 Mbps	100 Mbps

Fonte: TELECO (2018).

3.2 Sistema Operacional Linux

Responsável por gerenciar a memória, *threads*, processos, redes, *drivers*¹ e segurança dos arquivos e pastas, o sistema operacional do Android foi baseado no *kernel* (*núcleo do sistema operacional*) do Linux, e se encarrega de tudo isso. Segundo

¹ Componentes adicionais instalados no sistema que permitem a comunicação do Sistema operacional com algum dispositivo de *hardware*.

CAMPOS, (2006), as distribuições Linux se tornaram populares por serem alternativas livres, sendo populares em ambientes corporativos em servidores.

Toda a segurança do Android é baseada na segurança do Linux. No Android cada aplicação é executada em um único processo e cada processo por sua vez possui um *thread* dedicada. É criado um usuário no sistema operacional para ter acesso a sua estrutura de diretórios para cada aplicação instalada no celular. Desta forma nenhum outro usuário pode ter acesso a esta aplicação.

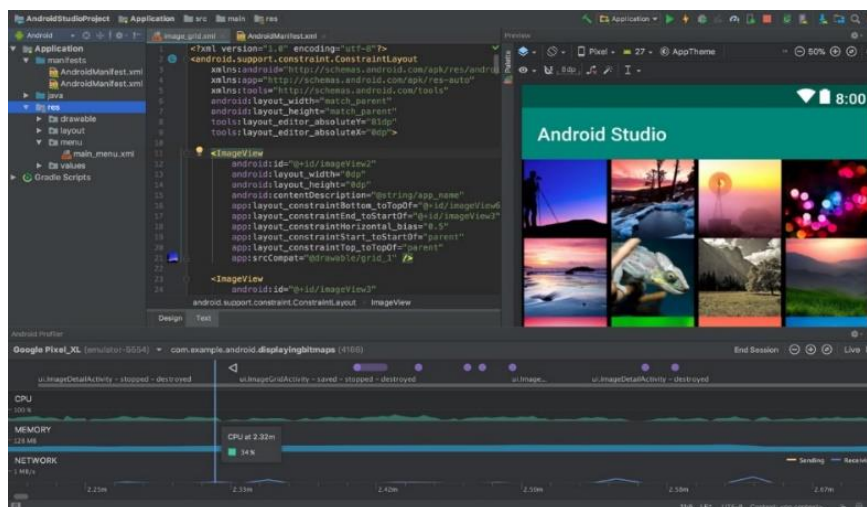
3.3 Plataforma Android

A plataforma *Android* contém um sistema operacional baseado em *Linux*, de código aberto, com a vantagem de corrigir erros mais rapidamente e criar uma comunidade de programadores que centralizam nessa funcionalidade, de modo que essas modificações não fiquem disponíveis a outras companhias, tendo uma nova plataforma de desenvolvimento para aplicativos móveis como os *smarthphones*, *tablets*, e *smartwatches*, incluindo diversas aplicações já instaladas, com um ambiente bastante poderoso, inovador e flexível, formando uma rica interface visual com a linguagem Java para desenvolver as aplicações.

O Java é uma linguagem orientada a objetos e a programação de interface gráfica para o usuário possui amplas bibliotecas de classe que ajudam a desenvolver inúmeros aplicativos, torna-se uma escolha lógica para a plataforma Android, pois é gratuita, por esse intermédio os programadores se aprofundam no desenvolvimento utilizando as Interfaces de Programação de Aplicativo (APIs) do Google e outros.

Para esse desenvolvimento, atualmente é utilizado o Android Studio (Fig.4), um Integrated Development Environment (IDE). O *software* é disponível gratuitamente sob a Licença Apache 2.0, o mesmo foi enunciado na conferência Google no dia 16 de Maio de 2013.

Figura 4 - Ambiente de desenvolvimento Android Studio.



Fonte: Developers (2018).

O Android Studio passou por várias versões, e teve acesso antecipado começando pela versão 0.1 em Maio de 2013, entrando em estágio beta a partir da versão 0.8 que foi lançada em Junho de 2014. Em dezembro de 2014 foi realizada a primeira compilação da versão 1.0.

Para um trabalho mais produtivo e mais rápido, a versão 3.1.2, conforme figura 20, é a mais atualizada do Android Studio, oferecendo um editor de código inteligente com preenchimento de códigos para as linguagens Java, C e C++.

3.4 Versões do Android

De acordo com a empresa Developers (2018) periodicamente são disponibilizadas novas versões do Android, contendo novos recursos, melhoria de desempenho e correção de problemas. Cada versão do Android utiliza um nível de *Application Programming Interface* (API) diferentes, na Tabela 2 estão as versões mais recentes do Android. Quando é desenvolvida uma aplicação que faz o uso de uma versão de API de nível inferior a mais recente, essa aplicação será compatível em 100% com a versão em que foi desenvolvido e com todas as versões posteriores, e ao utilizar o último nível da API do Android, ele não será compatível com os níveis e versões inferiores do Android.

Tabela 2 - Versões da plataforma Android

Versão	Nome de código	API	Distribuição
2.3.3 2.3.7	Gingerbread	10	0.3%
4.0.3 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%

Fonte: Developers (2018).

4 METODOLOGIA

Este capítulo detalha o desenvolvimento do trabalho, incluindo a participação de campo em uma vistoria realizada utilizando métodos tradicionais, e o desenvolvimento de um aplicativo, que é composto do levantamento de requisitos, modelagem da aplicação, diagramas *Unified Modelling Language* (UML), *sketchs*², trechos de código, telas do programa, teste do programa e seu manual de uso.

4.1 Estudo de caso – vistoria naval

O curso Superior Tecnológico em Sistemas Navais abriu a oportunidade para este estágio e através dos conhecimentos adquiridos na área naval, e, associados aos do Curso Superior em Sistemas para Internet, tornaram possível a realização deste estudo e a criação do aplicativo que atenda a necessidade do vistoriador.

No dia 02 de maio de 2018, a autora deste estudo acompanhou uma vistoria naval juntamente com o profissional de uma Certificadora, sediada na cidade de Pederneiras (Fig.5). Realizou-se a vistoria de borda livre na embarcação pertencente a uma empresa que executa trabalhos na extração de areia (Tab.4).

Figura 5 - Acompanhamento da vistoria à embarcação.



Fonte: Autora.

² Esboço, rascunho ou desenho de menor qualidade para representar o estado inicial ou protótipo.

Tabela 3 - Detalhamento técnico da embarcação vistoriada.

Características	Especificação
Tipo de embarcação	Areeiro
Comprimento total	40,940 m
Comprimento entre perpendiculares	40,940 m
Boca moldada	6,500 m
Pontal moldado	3,060 m
Área de navegação / Tipo de serviço	Área 1 / Draga de areia
Material do casco	Aço carbono
Número de tripulantes	00
Calado Máximo carregado	2,540 m
Deslocamento carregado	589,796 t
Arqueação líquida	63
Arqueação Bruta	191
Porte bruto	434,850 t

Fonte: Autora.

O processo acontece por etapas, começando pela análise do projeto de arranjo geral, visualizando o perfil do plano do alto e comparando com a embarcação vistoriada. Primeiramente foram aplicadas as verificações visuais da embarcação para assegurar que não aconteceram alterações no casco e/ou nas superestruturas que possam alterar a borda livre anteriormente atribuída, conforme MARINHA DO BRASIL, DIRETORIA DE PORTOS E COSTAS, 2018.

Verificou-se visualmente se as condições do casco são satisfatórias e garantam: a segurança e resistência estrutural e não apresentem deterioração, mossas, trincas ou furos por corrosão que comprometam a estanqueidade da embarcação de acordo com item 40 do anexo 8-A. A verificação dos porões estanques é feita visualmente, de dentro deles, se existem avarias nas anteparas, no fundo, no costado e convés.

A figura 6 ilustra a medição realizada e a verificação visual quanto as condições dos suspiros do convés principal de borda-livre, segundo o item 0611 que determina,

Extremidade superior do suspiro em forma de “U” invertido ou com arranjo que proteja a sua abertura da entrada de água proveniente das intempéries; e (b) Distância vertical entre o ponto a partir da qual a água efetivamente tem acesso ao tanque ou compartimento abaixo e o convés onde o suspiro se encontra instalado maior ou igual a 450 mm. (MARINHA DO BRASIL, DIRETORIA DE PORTOS E COSTAS, 2018)

Figura 6 – Medição suspiro do convés principal.



Fonte: Autora.

Realizou-se a medição da distância de borda livre, feita através de uma trena (Fig.7) posicionada verticalmente de cima para baixo a meia nau da embarcação do lado superior da linha do convés à margem superior da linha de carga correspondente, como também, foi observada a sigla estampada no casco da embarcação correspondente a certificadora responsável e a área de navegação, de acordo com a documentação da mesma.

Figura 7 - Medição de borda livre.



Fonte: Autora.

Todas as aberturas dos porões e convés da embarcação foram abertas, por ocasião da vistoria, para a facilidade de acesso ao vistoriador, com a retirada dos parafusos de fixação e borrachas de vedação, conforme apresenta a figura 8.

Figura 8 - Abertura dos porões



Fonte: Autora.

A figura 9 representa o disco de Plimsoll dado das marcas da linha de carga, de acordo com item 0622 da NORMAM 02/DPC (MARINHA DO BRASIL, DIRETORIA DE PORTOS E COSTAS, 2018) foi visto a marca do convés que foram examinadas: a pintura do casco e a cor de contraste de fundo fixadas em ambos os bordos da embarcação, conforme informado no Certificado Nacional de Borda Livre da embarcação. Também foram conferidas as respectivas siglas da certificadora de acordo com a área de navegação, no caso área 1. Outro item observado foi a reavivação da escala de calado em cada lado da embarcação (proa, popa e meia-nau), e também as marcações das escalas representadas a cada 20 cm até o limite do pontal.

Figura 9 - Disco de Plimsoll e marca da linha do convés



Fonte: Autora.

O estado de conservação e manutenção da proteção de aberturas de portas de visitas horizontal foram examinadas, para constatar se possuem condições de estanqueidade, representada na figura 10. Conferiu-se a altura das soleiras de acesso ao interior da embarcação para impedir a entrada de água, que deve ser de no mínimo 150 mm segundo o item 0611 no Capítulo 6.

Figura 10 -Porta de visitas.



Fonte: Autora.

Para finalizar a vistoria, verificou-se o estado de conservação da balaustrada e a medição, a partir do topo do convés principal até a altura da balaustrada (Fig.11) que deve obedecer a medida mínima de um metro, para embarcações com AB maior que 20, a fim de garantir a segurança dos tripulantes que circulam no conveses e superestruturas, conforme especificações no item 0427, Capítulo 4 NORMAM 02/DPC (MARINHA DO BRASIL, DIRETORIA DE PORTOS E COSTAS, 2018).

Figura 11 -Balaustrada.



Fonte: A autora (2018).

4.1.1 Resultado da vistoria

Não houve nenhuma inconformidade quanto a vistoria de borda livre em qualquer um dos itens verificados. Em contrapartida, enquanto acompanhava o trabalho do vistoriador constatou-se que o profissional enfrentou dificuldades dentro da praça de máquinas pela falta de luz, atrapalhando a visibilidade para ler o *checklist*, e dificuldades para subir e descer a escada com os equipamentos nas mãos, necessitando da ajuda de uma outra pessoa na transferência dos materiais que carregava, principalmente na entrada e saída do porão, reforçando os pontos da problemática deste trabalho que poderá ser facilitado com a utilização do APP.

4.2 Levantamentos de requisitos

Após análise do processo de vistoria citado na seção anterior, chegou-se as principais necessidades para o desenvolvimento da aplicação, entre eles os requisitos funcionais e os não funcionais. Os requisitos funcionais definem as ações fundamentais através das quais o produto aceita e processa as entradas especificadas, gerando as respectivas saídas.

Todo o detalhamento é feito nesta seção a nível suficiente para o desenho do produto, de seus testes de aceitação e de seu manual de usuário, tais como:

- A aplicação deve executar na maioria dos aparelhos Android no mercado;
- Diversos tipos de vistoria devem ser possíveis;
- O usuário poderá salvar várias vistorias;
- O app facilitará o acesso a relatório de vistorias já realizadas.

Os requisitos não funcionais são os que incluem requisitos de desempenho e outros atributos de qualidade do produto. Por isso, a maioria das decisões de arquitetura será consequência dos requisitos não-funcionais, a aplicação deverá:

- Garantir a segurança dos dados;
- Possuir interface amigável;
- Bom desempenho.

4.3 Modelagem do software/aplicativo

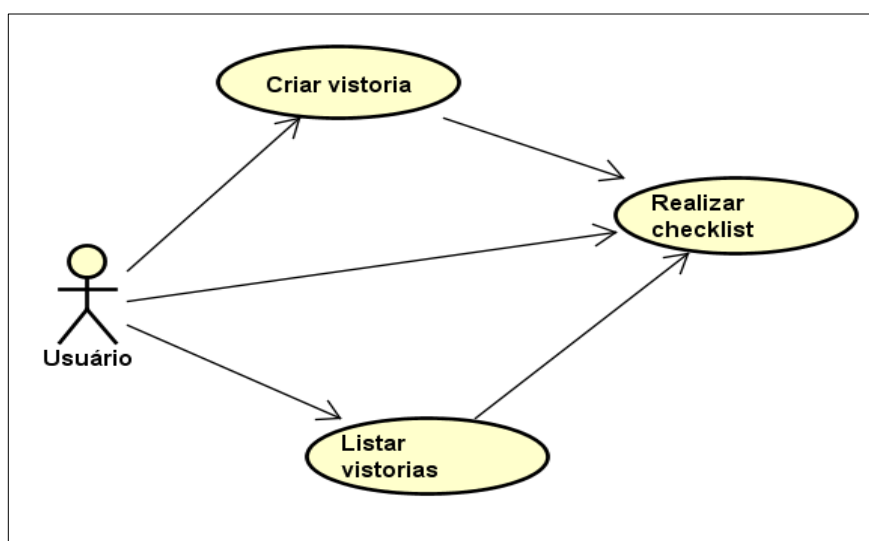
4.3.1 Diagramas UML

Os diagramas *Unified Modelling Language* (UML) são importantes ferramentas para a engenharia de software dos processos de desenvolvimento e criação de software, pois segundo *Booch, Rumbaugh* e *Jacobson* (2006) permite através de gráficos relativamente simples especificar o funcionamento e estruturar a criação de sistemas. Nas próximas seções estão dispostos os principais diagramas UML criados durante o processo de modelagem no software de modelagem *Astah Community*.

4.3.2 Diagramas de Caso de Uso

Como primeiro passo no desenvolvimento da aplicação foram identificados os principais atores e funcionalidades esperadas para a mesma. Por ser uma aplicação simples temos somente um ator que interage com o sistema, que é o próprio usuário da aplicação ilustrado na figura 12.

Figura 12 - Diagrama de caso de uso.



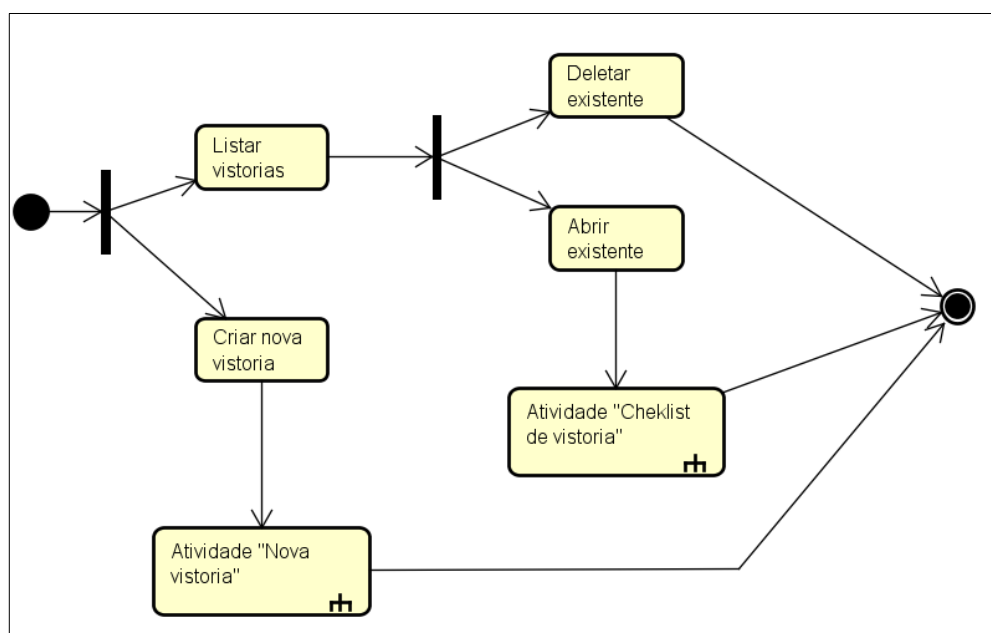
Fonte: Autora.

As principais funcionalidades esperadas em que o usuário tem interação direta são a criação e realização de novos *checklists* de vistorias.

4.3.3 Diagramas de Atividade

Na figura 13 apresenta o diagrama de atividades que ilustram o comportamento esperado por um software ou fluxo de processos, descreve em um nível mais amigável o que se espera que o algoritmo faça antes de se ter acesso ao código.

Figura 13 - Diagrama de atividades para a atividade principal da aplicação.



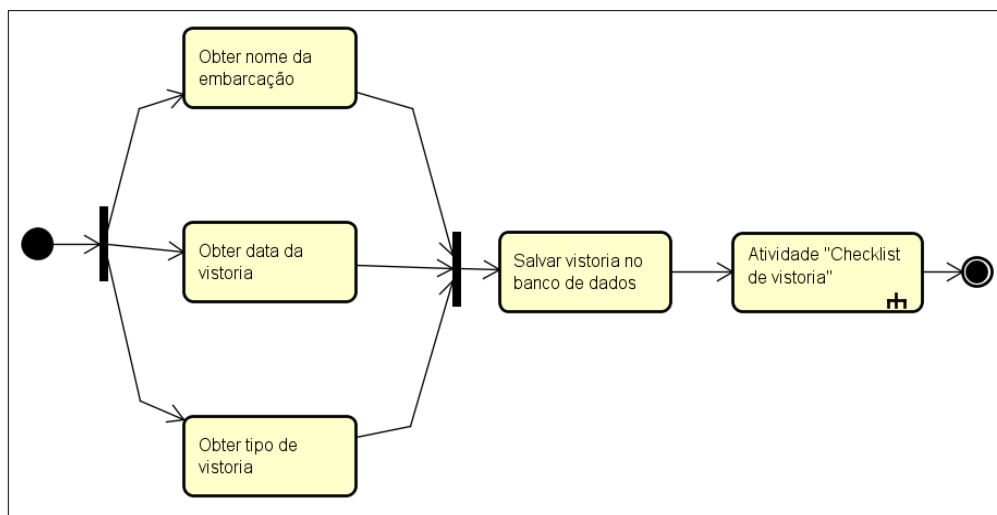
Fonte: Autora.

As principais funcionalidades dispostas pelo diagrama de caso de uso são melhor descritas em seu fluxo de processo. O primeiro diagrama de atividade descrito é o diagrama relativo a atividade principal, onde observando a estrutura de aplicativos Android, analogamente descreve o funcionamento da *Activity MainActivity* da nossa aplicação.

No diagrama o fluxo esperado acontece da esquerda para a direita, onde a barra representa uma bifurcação, ou escolhas possíveis, e as caixas com cantos arredondados são as tarefas. Certas tarefas remetem à outras atividades ou *Activities*

no Android, como a tarefa de criação “Criar nova vistoria” que inicia a atividade “Nova vistoria” que é ilustrada na figura 14.

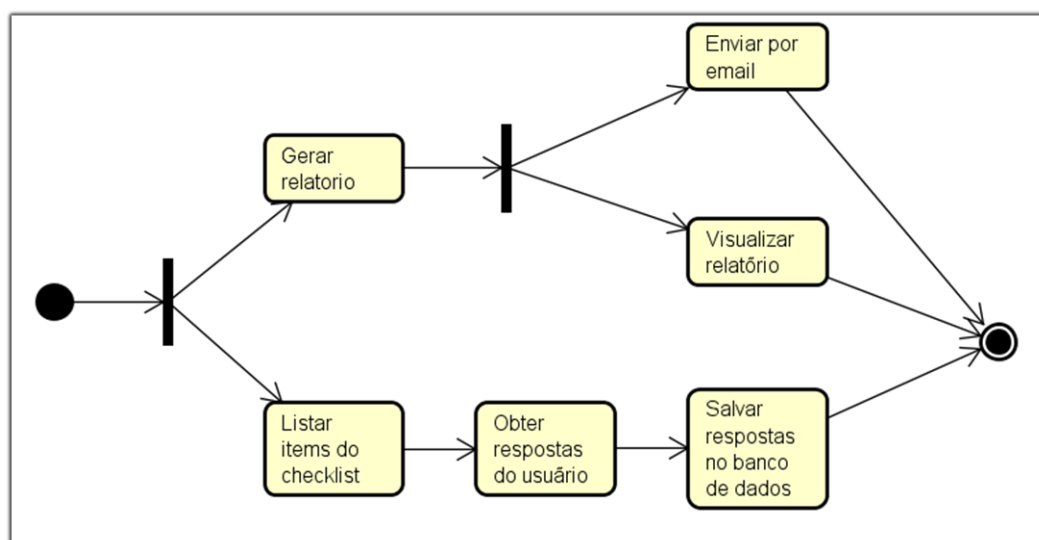
Figura 14 - Diagrama de atividade para a criação de novas vistorias.



Fonte: Autora.

No diagrama da figura 15, a barra onde somente uma seta sai a esquerda indica que todas as tarefas que chegam até este ponto devem ser completadas, como é o caso dos campos requeridos para a criação da nova vistoria. Como tarefa final para essa atividade temos a atividade externa de “*ChecklistCategoria*”, que é a atividade mais simples onde temos somente a listagem dos itens para a vistoria.

Figura 15 -Diagrama para a atividade de checklist.



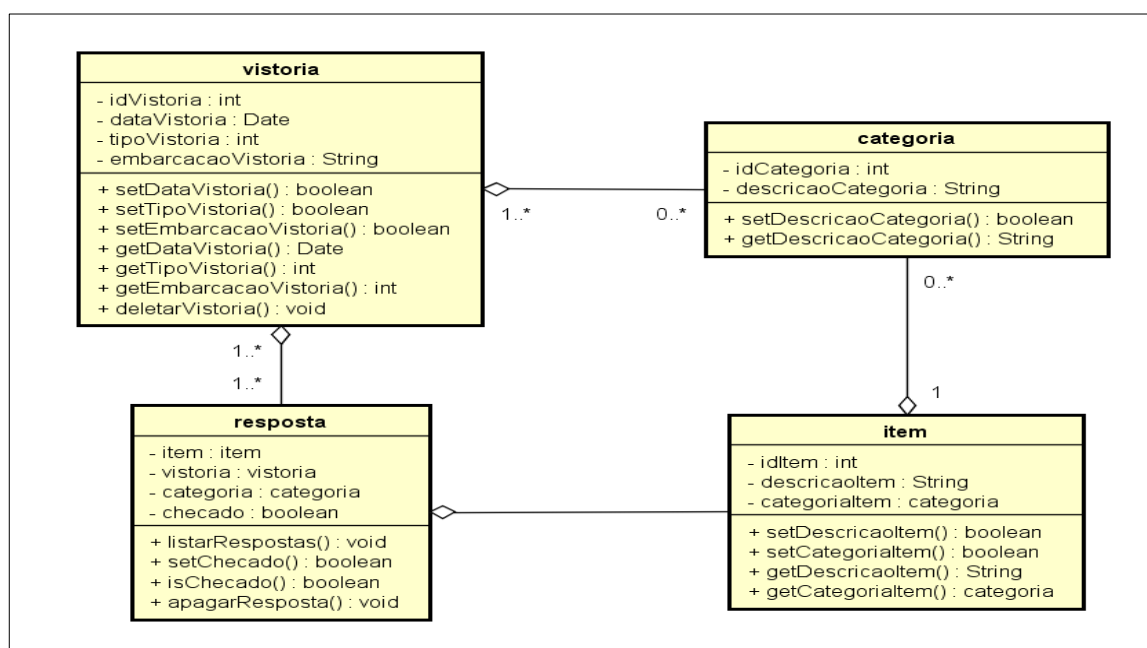
Fonte: Autora.

4.3.4 Diagrama de Classe

O diagrama de classes serve como representação dos objetos e suas relações e dependências. Está fortemente ligado à programação orientada a objetos, como é a linguagem Java, presente no desenvolvimento de aplicações Android. Em um objeto temos as características identificadas pelos atributos e as operações ou tarefas identificadas pelos métodos.

O diagrama de classes inicial da aplicação desenvolvida (Fig.16). Neste diagrama temos quatro classes principais, onde são mostradas as suas dependências e restrições de suas relações. Por exemplo, um objeto da classe vistoria somente deve ter um objeto categoria associado a ele, enquanto que a classe categoria pode ter vários itens associados.

Figura 16 - Diagrama de classes inicial simplificado.



Fonte: Autora.

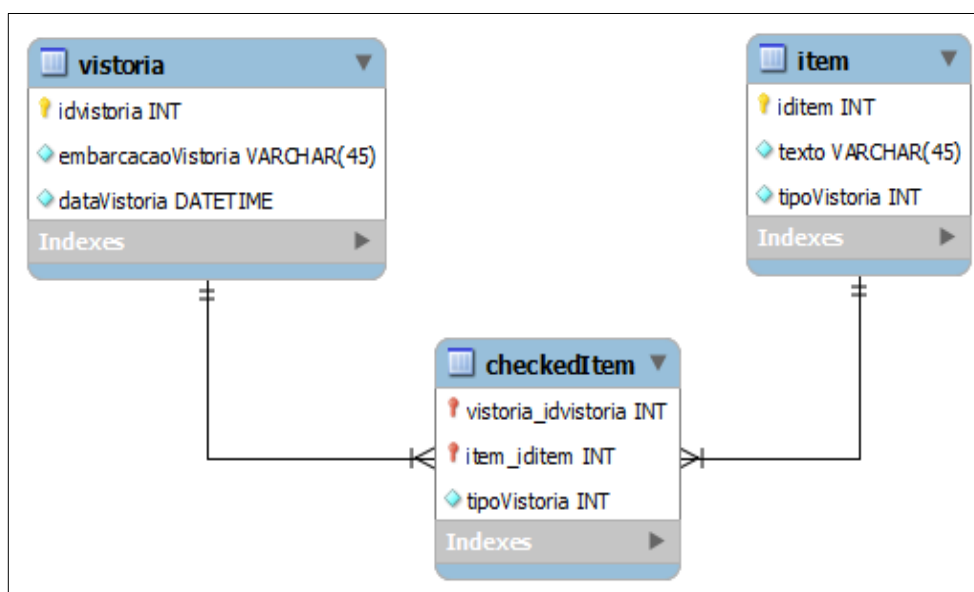
4.3.5 Diagrama de banco de dados

A plataforma Android possui diversas formas de aplicar persistência (ou gravar dados), temos o salvamento em arquivo (texto ou binário) para registro de informações

e a persistência de classes utilizando banco de dados, sendo o gerenciamento de banco de dados feito pela Linguagem de Consulta Estruturada SQLite.

A figura 17 ilustra o banco de dados para a aplicação desenvolvida, onde temos as tabelas que foram utilizadas para a persistências de algumas classes. Enquanto que a maioria das classes foram representadas aqui, a classe (categoria aqui identifica como o campo tipoVistoria) foi armazenada como uma estrutura de dados inserida no programa por conter poucos itens e não prever a inserção ou remoção de categorias.

Figura 17 - Diagrama de banco dados.



Fonte: Autora.

Refletindo a implementação do banco de dados na aplicação, a tabela “checkedItem” representa a classe resposta. Uma entrada nessa tabela representa a marcação do *checkbox* em um item do *checklist*, enquanto que a exclusão de uma entrada acontece quando o usuário desmarca um item.

O diagrama de banco de dados foi feito a partir do programa MySQL (*My Structured Query Language*) Workbench, apesar do mesmo ser desenvolvido para trabalhar com um banco de dados mais robusto como o MySQL.

4.4 Desenvolvimento de UI (interface do usuário)

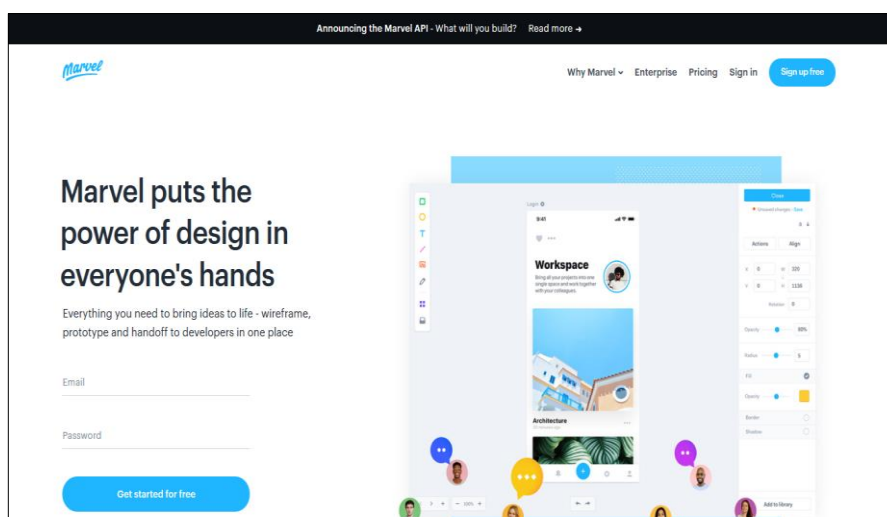
O desenvolvimento da interface do usuário, segue inicialmente com as técnicas de *wireframe*, que se utiliza de rascunhos das possíveis telas em que o usuário irá

interagir. A parte final do desenvolvimento de interface é a prototipação, que utiliza-se da criação das telas com mais detalhes, e é onde temos o desenvolvimento da interface próximo da versão funcional obtida durante o processo de codificação da aplicação.

A criação em *wireframe* pode partir da criação de rascunhos das telas desenhados manualmente, ou com o uso de programas de desenho e edição de imagem. Além disso, existem os programas e recursos criados especialmente para a criação dos rascunhos, como os listados por Naylor (2017).

Como a maioria desses programas estão disponíveis apenas para computadores Apple, e/ou são pagos, o programa escolhido para a criação de *wireframes* foi a aplicação web Marvel (2018), ilustrada na figura 18.

Figura 18 - Web APP Marvel para a criação e desenvolvimento de *wireframes*.



Fonte: Autora.

O desenho das telas seguiu as orientações presentes no guia de práticas *Material Design* (GOOGLE. 2018) mantido pela Google com o objetivo de fornecer um manual para o desenvolvimento de aplicativos Android sendo pensada para a experiência do usuário (UX).

O desenho das possíveis telas se baseou nas atividades previstas anteriormente, que foram convertidas durante a codificação no par *Activity/Interface*, onde a *Activity* é um código Java e a interface é uma estrutura escrita na Linguagem Extensível de Marcação Genérica (*xml*).

A figura 19, ilustra o funcionamento da aplicação utilizando os rascunhos gerado na *web APP* Marvel.

Figura 19 -Rascunho das telas e simulação de interação do usuário.



Fonte: Autora.

4.5 Desenvolvimento no Android Studio

Para o desenvolvimento da aplicação foi utilizado o Android Studio na versão 3.1.2, sendo que os testes foram realizados no emulador presente na *Integrated Development Environment* (IDE) e em *smartphones* Android.

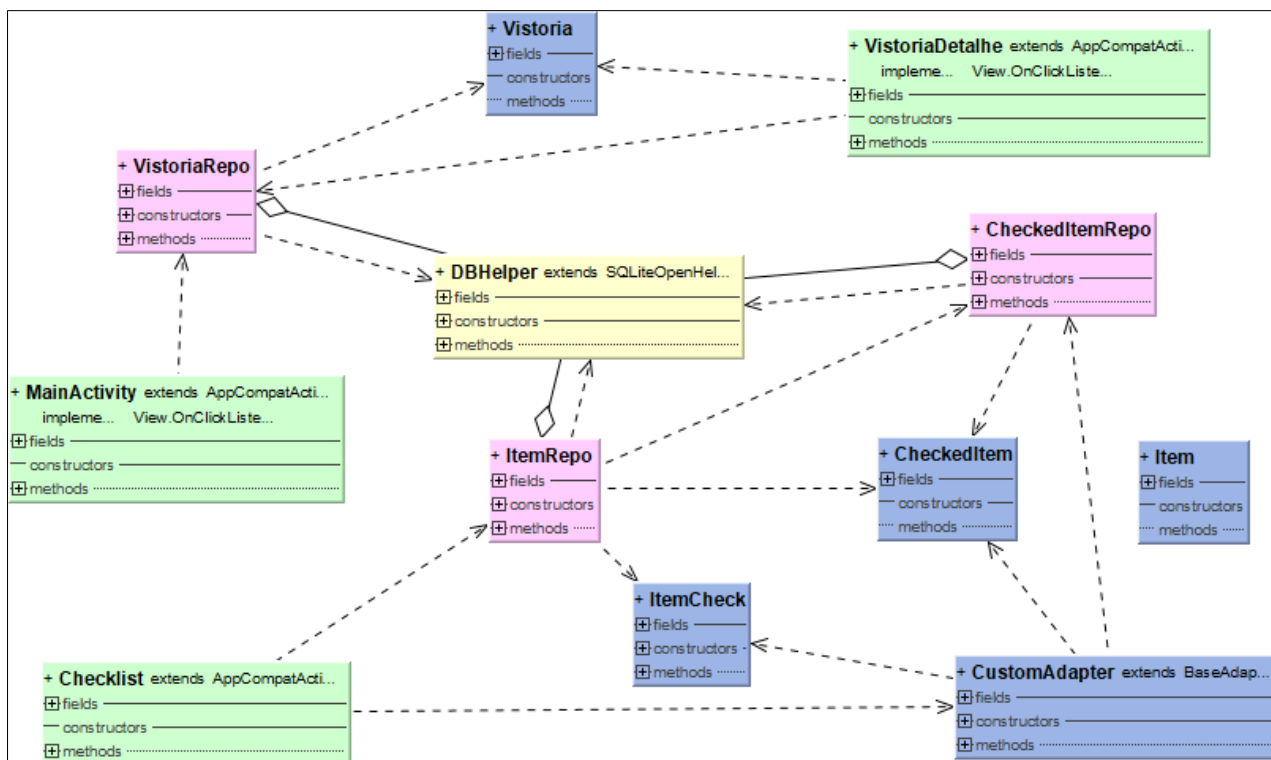
As aplicações Android são desenvolvidas a partir das definições de suas *Activities* ou Atividades, partindo do pressuposto de que a partir delas é que o usuário irá interagir com o aplicativo. As *Activities* são compostas pela classe *java* e pelo desenho de sua interface em padrão *xml*.

A aplicação possui três telas ou *Activities* sendo elas; a *MainActivity* ou atividade principal que aparece quando o aplicativo é aberto; a *Activity VistoriaDetalhe* que é onde uma nova vistoria é criada, ou aberta para edição; e a *Activiy Cheklist* que lista os itens da NORMAM de acordo com o tipo de vistoria escolhida, salvando no banco de dados os itens marcados.

A figura 20, ilustra o diagrama de classes gerado a partir do Android Studio utilizando o plugin simpleUML. Na figura as classes relacionadas à *Activities* estão

indicadas na cor verde, as classes de apoio a persistência das classes estão em cor rosa, e as classes de apoio e outros tipos estão em cor azul.

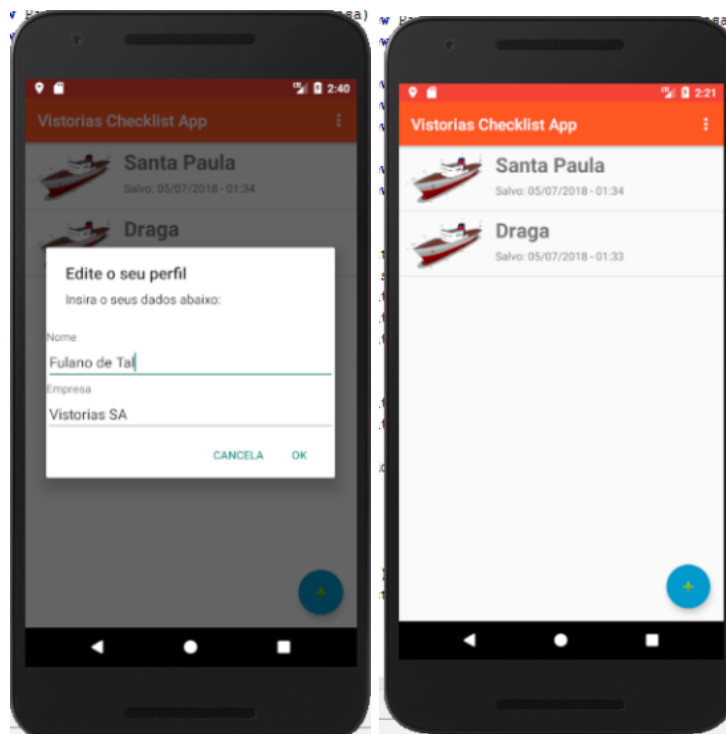
Figura 20 - Diagrama de classes gerado a partir da aplicação no Android Studio.



Fonte: Autora.

A classe *MainActivity* é a classe que lista as entradas para as vistorias já salvas em um objeto *ListView* (objeto para exibição de lista no Android) onde é possível abrir detalhes de uma vistoria salva, ou criar uma vistoria tocando no botão com o símbolo “+”. Na figura 21 se apresenta a tela inicial do programa, onde se apresenta o cadastro das embarcações. Esta classe é ligada à classe *VistoriaRepo* que possui métodos para recuperar as vistorias salvas implementando a classe *DBHelper*.

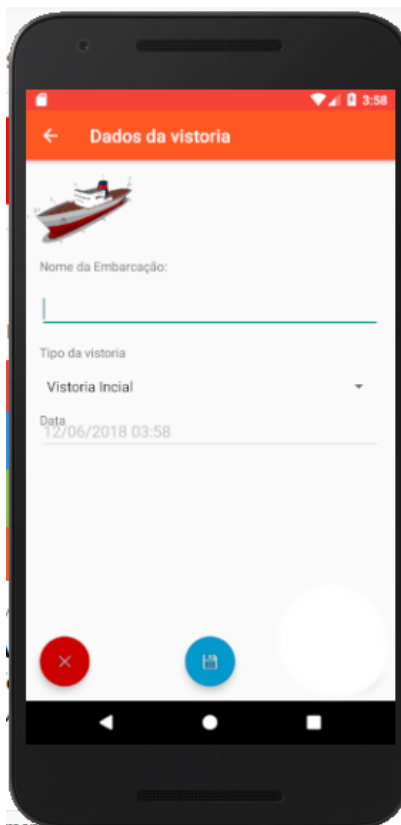
Figura 21 - Tela inicial para cadastro e tela das vistorias salvas.



Fonte: Autora.

Ao clicar em adicionar vistoria ou exibir uma vistoria da lista um *Intent* é lançado chamando a *Activity* *VistoriaDetalhe*. A classe *VistoriaDetalhe* exibe o nome embarcação e o horário da última alteração ou o horário atual caso uma nova vistoria esteja sendo criada conforme figura 22, nesta tela será realizado o cadastro da embarcação que será vistoriada. A partir desta *Activity* é possível excluir uma vistoria, salvar alterações e abrir o *checklist* excluído. Esta classe utiliza a classe *VistoriaRepo* para salvar ou excluir uma vistoria no banco de dados.

Figura 22 - Adicionando uma nova vistoria.

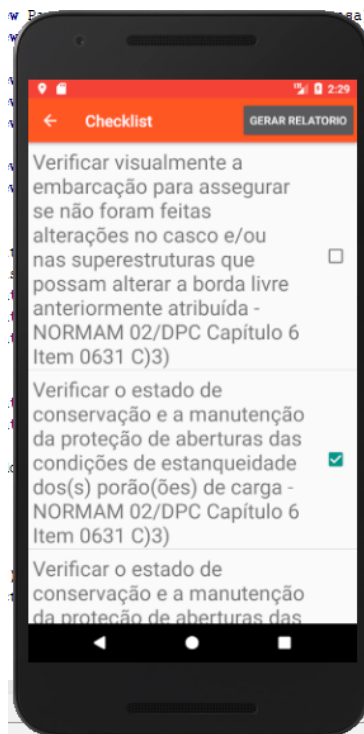


Fonte: Autora.

Ao ser tocado ou clicado, o botão com o símbolo X, a partir dos métodos da classe *VistoriaRepo* exclui a vistoria e os itens checados para aquela vistoria do banco de dados. para a *Activity* principal. O botão salvar ou abrir, lança o método para salvar a vistoria, e lança um *Intent* para iniciar a *Activity Checklist* de acordo com as opções escolhidas.

A *Activity Checklist* exibe uma lista com os itens da NORMAM 02/DPC, 2018 para a vistoria, juntamente com um *checkbox*, nesta tela aparecem todos os itens relativos a vistoria de Borda Livre e poderão ser marcados ou desmarcados (Fig.23).

Figura 23 - Exibição do *checklist* para uma vistoria.

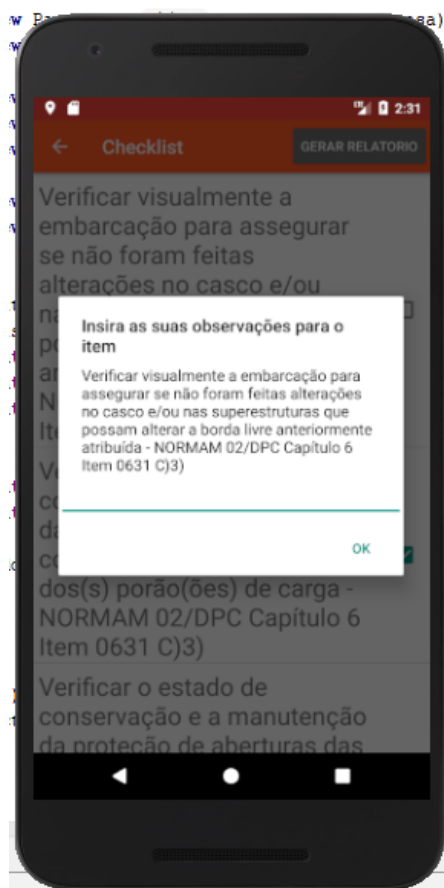


Fonte: Autora.

A classe *Checklist* com o auxílio das classes *ItemRepo*, *ItemCheck* e *CustomAdapter* verifica se um item do *checklist* foi marcado ou desmarcado, salvando ou excluindo respectivamente a entrada do item no banco de dados. Quando o usuário clica na seta voltar a *Activity* é fechada, e é exibida novamente a *Activity* principal.

Caso o usuário opte por clicar em gerar relatório, uma tela surgirá para o usuário poder entrar com as observações sobre os itens não conformes (itens desmarcados no checklist) como apresentado na figura 24.

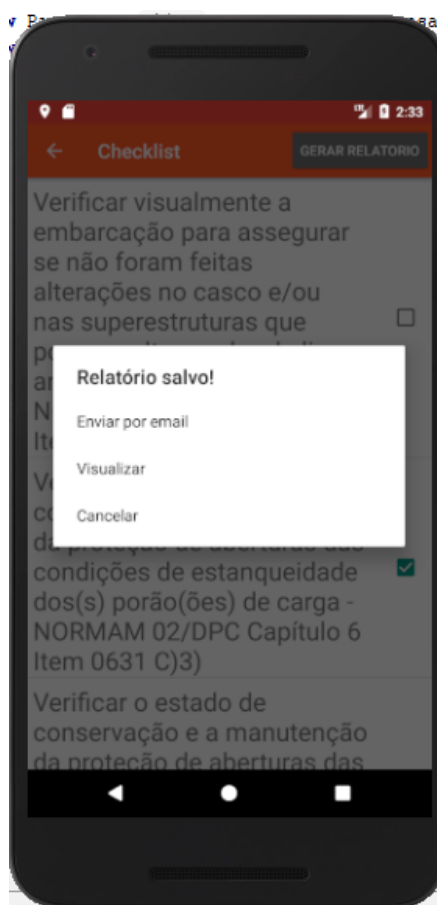
Figura 24 -Tela solicitando a entrada das observações sobre itens não marcados.



Fonte: Autora.

Após inserir todas as observações, se requisitadas, o aplicativo gerará um relatório em *pdf*, questionando se usuário deseja visualizar o arquivo no próprio aparelho, ou enviar por *email*, como na Figura 25. Caso o usuário abra o mesmo *checklist* posteriormente, aparecerá uma outra tela perguntando se o usuário deseja abrir o mesmo ou gerar um novo relatório.

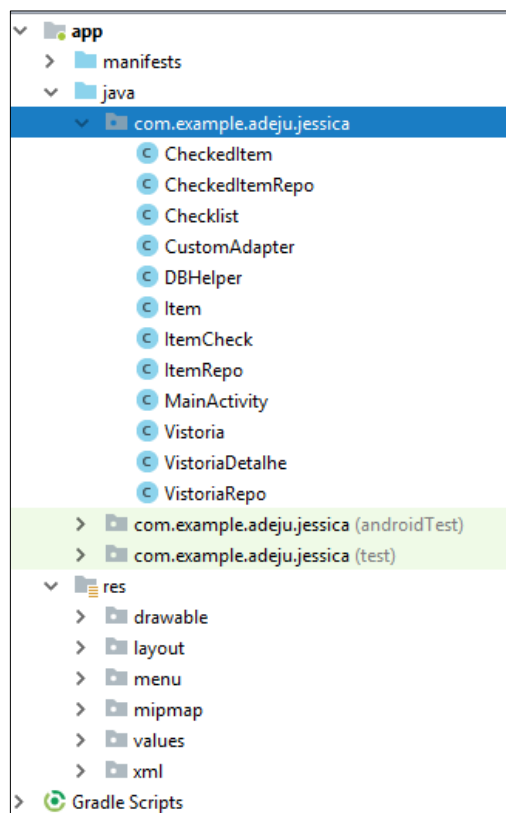
Figura 25 - Tela questionando o usuário o próximo passo após criado o relatório.



Fonte: Autora.

A estrutura de arquivos da aplicação pode ser vista na figura 26, onde temos a pasta “java” onde estão armazenadas as classes, a pasta “res” onde estão armazenados os recursos como ícones (sub-pasta *drawable*), desenho das telas em xml (sub-pasta *layout*), e recursos de texto como listas e nomes (sub-pasta *values*). O código fonte da aplicação encontra-se no apêndice.

Figura 26 - Estrutura de pastas do projeto no Android Studio.



Fonte: Autora.

5 CONCLUSÕES, CONSIDERAÇÕES E PROPOSTAS PARA TRABALHOS FUTUROS

Este aplicativo é recomendado especificamente para as entidades certificadoras, denominado como: “J.A Vistoria”, atende especificamente a vistoria de borda livre da embarcação. Neste capítulo descrevem-se as características do aplicativo e suas funcionalidades, como também, dificuldades apresentadas e possíveis soluções e melhorias.

Usabilidade do aplicativo: na tela principal aparece um espaço para preencher o nome da embarcação, a seguir, seleciona-se o tipo de vistoria, que futuramente poderá se acrescer novos tipos, porém no momento a de borda livre está disponível. Numa nova tela, o profissional pode selecionar cada item desejado, conforme sua rotina e na ordem que achar necessária e salvar suas alterações.

Ao final da seleção poderá gerar um relatório com as exigências necessárias para alterações futuras e acompanhar os requisitos que estavam em inconformidade. Em qualquer momento o vistoriador poderá, na tela de dados da vistoria, procurar a embarcação cadastrada e visualizará todos os dados da mesma na tela de *checklist*. Quando o usuário desejar deletar a embarcação cadastrada, deverá clicar no botão excluir, que está programado na tela Dados da Vistoria, assim, a tela volta ao menu principal. Durante os testes realizados foram encontrados alguns detalhes que foram corrigidos.

O aplicativo funcionou perfeitamente, ainda que de forma simples, atingiu seu objetivo. Portanto, se considerar a enorme lista de diferentes tipos e especificações para vistoria, percebe-se que há muito a ser estudado para implantação de novos itens que facilitem o trabalho do vistoriador naval.

Ao testar o aplicativo foi encontrado alguns pontos que podem ser melhorados:

1. Cadastrar as características principais da embarcação, inserir dados específicos que facilitem a vistoria e a consulta das mesmas;
2. Inserir o tipo da embarcação, especificar material do casco, o armador, o construtor, posto de inscrição;
3. Inserir foto da embarcação em questão;

4. Registrar os dados do armador e/ou construtor, como: nome, CPF, CNPJ, endereço, celular, e-mail, etc.;
5. Área de navegação a qual pertence.

Para aprimorar, sugere-se que sejam criados links em cada uma das normas inclusas, e quando necessário possam direcionar a NORMAM específica, para que em tempo real possa conferir sua aplicabilidade de forma atualizada com as mudanças frequentes que costumam ocorrer.

Este estudo abrirá espaço para novas ideias de aplicativos, utilizando e melhorando a forma apresentada, com objetivos dos mais variados, que atendam a área comercial e profissional.

BIBLIOGRAFIA

ASSI, Gustavo R.S. Departamento de Engenharia Naval e Oceânica Escola Politécnica da Universidade de São Paulo. 2011. **Representação geométrica da embarcação: Fundamentos da Engenharia Naval Prof. Dr.** Disponível em: <http://www.ndf.poli.usp.br/~gassi/disciplinas/pnv2341/PNV2341_Representa%C3%A7%C3%A3o_geomatrica_do_casco.pdf>. Acessado em: 20 Jun.2018.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário.** Elsevier Brasil, 2006.

BUXTON, Bill. Buxton **Collection: Simon.** Disponível em <<https://www.microsoft.com/buxtoncollection/detail.aspx?id=40>>. Acessado em: 05 Jun.2018.

CAMPOS, Augusto. **O que é Linux.** BR-Linux. Florianópolis, março de 2006. Disponível em: <<http://br-linux.org/linux/faq-linux>>. Acessado em: 20 mai.2018.

CARVALHO, Lucas. **Android cresce no Brasil e aumenta distância para IOS e Windows Phone**, 2017. Disponível em: <<https://olhardigital.com.br/noticia/android-cresce-no-brasil-e-aumenta-distancia-para-ios-e-windows-phone/68023>>. Acessado em: 07 mai.2018.

CORDEIRO, Fillipe. **Android SDK: O que é? Para que serve? Como usar?**. Disponível em: <<https://www.androidpro.com.br/blog/android-studio/android-sdk/>>. Acessado em 24 mai.2018.

DEITEL, Paul; DEITEL, Harvey; WALD, Alexander. **Android 6 para Programadores-3ª Edição: Uma Abordagem Baseada em Aplicativos.** Bookman Editora, 2016. p.3-4.

DEVELOPERS. **Versões da plataforma.** Disponível em: <<https://developer.android.com/about/dashboards/?hl=pt-br>>. Acessado em: 23 mai.2018.

DEVMEDIA. **Top 10 linguagens de programação mais usadas no mercado.** Disponível em: <<https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635>>. Acessado em 24 mai.2018.

DUB SOLUÇÕES. **Estatística de Uso de Aplicativos no Brasil**, 2017. Disponível em: <<https://www.dubsolucoes.com/single-post/estatisticas-de-uso-de-aplicativos-no-Brasil>>. Acessado em: 12 abr.2018.

GOOGLE. **Material Design - Guidelines.** Disponível em: <<https://material.io/design/guidelines-overview/#addition>>. Acessado em: 06 Jun.2018.

KONRADSSON, T. **ART and Dalvik performance compared.** UMEA Universitet 2015. Disponível em:

<<http://www8.cs.umu.se/education/examina/Rapporter/TobiasKonradsson.pdf>>
Acessado em: 19 jun.2018.

LECHETA, Ricardo R. **Google Android-4ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Novatec Editora, 2015. p-27-28.

LIMA, Pedro Pereira Lima; THEDORO, Luiz Cláudio; DANTAS, Servolo. **Quinta Geração das redes móveis**. XII CEEL - Conferência de Estudos em Engenharia Elétrica. Universidade Federal de Uberlândia. Uberlândia, 2014.

Marinha do Brasil – Diretoria de Portos e Costas: “Normas da Autoridade Marítima para Embarcações Empregadas na Navegação Interior – Normam – 02”. 2018.

Marinha do Brasil – Diretoria de Portos e Costas: “Normas da Autoridade Marítima para Embarcações Empregadas na Navegação Interior – Normam – 11”. 2018.

MARVEL. **Everything you need to bring ideas to life**. Disponível em <<https://marvelapp.com/why-marvel>>. Acessado em: 06 Jun.2018.

MEILI, Leandro. **Software para desenvolvimento de projetos navais de acordo com a NORMAM 02**. 8º Seminário de Transporte e Desenvolvimento Hidroviário Interior - SOBENA HIDROVIÁRIO 2013. Fatec Jahu, Jaú-SP, 2013.

MORAES, Renata Santos de. **Aplicação da tecnologia 4G em projetos de telefonia**. Trabalho de Graduação (Graduação em Engenharia Elétrica) - Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2015.

MUKHERJEE, Sangeeta. **Smartphone Evolution: From IBM Simon To Samsung Galaxy S3**. May, v. 8, p. 3, 2012.

National Ocean Service. **What is a Plimsoll line?** Disponível em:<<https://oceanservice.noaa.gov/facts/plimsoll-line.html>>. Acessado em: 25 de Jun.2018.

NAYLOR, Ian. **The 7 best wireframes and prototyping tools for app makers**. InfoWorld. 2017. Disponível em <<https://www.infoworld.com/article/3199370/development-tools/the-7-best-wireframing-and-prototyping-tools-for-app-makers.html>>. Acessado em: 06 de Jun.2018.

PEREIRA, Lucio Camilo Oliva; DA SILVA, Michel Lourenço. **Android para desenvolvedores**. Brasport, 2009.

POTHITOS, Adam. **The History of the Smartphone**. Mobile industry Review. Outubro, 2016. Disponível em <<http://www.mobileindustryreview.com/2016/10/the-history-of-the-smartphone.html>>. Acessado em: 05 de Jun.2018.

RABELLO, R.R. **Android: um novo paradigma de desenvolvimento móvel**. Disponível em: <https://dpwinfo.files.wordpress.com/2013/04/wm18_android.pdf>. Acessado em: 10 mai.2018.

RECORD CERTIFICADORA NAVAL. **Vistoria para emissão de certificados.**

Disponível em: <<http://www.certificadorarecord.com.br/servicos.php>> Acessado em: 20 Jun.2018

RENATO, Flávio. **A história dos telefones celulares**, 2012. Disponível em:

<<http://www.techtudo.com.br/artigos/noticia/2012/06/historia-dos-telefones-celulares.html>>. Acessado em: 12 abr.2018.

RIBEIRO, Quéven.; B.A.R.S, Rebecca. Os Impactos dos Dispositivos Móveis nas Pessoas. **Análise e Desenvolvimento de Sistemas - Revista Fatec Zona Sul**, São Paulo, v.2,n.1, p.2-3, abril, 2015.

SILVEIRA, Felipe. Activity - **O que é isso?** 2 de maio de 2010. Disponível em

<<http://www.felipesilveira.com.br/2010/05/activity-o-que-e-isso/>>. Acessado em: 06 Jun.2018.

TELECO. **Tecnologias de Celular**. 2018. Disponível em

<<http://www.teleco.com.br/tecnocel.asp>>. Acessado em: 05 Jun.2018

TERRA. **Você sabia? Quem inventou o telephone celular?** Disponível em <

<https://www.terra.com.br/noticias/tecnologia/celular/voce-sabia-quem-inventou-o-telefone-celular,9ae917e79a3207d7e3ced75f7a4f1f05tebbkqzg.html>>. Acesso em: 09 Jul.2018.

TYNAN, Dan. **The 50 greatest gadgets of the past 50 years**. PC World, December, v. 24, p. 123950-3, 2005. Disponível em

<https://www.pcworld.com/article/123950/the_50_greatest_gadgets_of_the_past_50_years.html>. Acessado em: 05 Jun.2018

VENTURA, Manuel. **Convenção Internacional das Linhas de Carga, 1966.**

Disponível em:< http://www.mar.ist.utl.pt/mventura/projecto-navios-i/pt/pn1.2.2-linhas_carga.pdf. >. Acessado em: 09 Jul.2018.

XAVIER, Jonas et al. **Estudo da evolução da telefonia móvel no Brasil**. X

Encontro de Iniciação Científica e VI Encontro de Pós-Graduação—Universidade do Vale do Paraíba. São José dos Campos-SP, 2006.

APÊNDICE

Código Fonte

MainActivity.java

```
package com.example.adeju.jessica;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    //Iniciando as variáveis globais da classe
    TextView vistoria_Id;
    ListView lv;
    ListAdapter adapter;
    String nome, empresa;

    /**
     * Método para gerenciar os toques na tela
     * @param view
     */
    @Override
    public void onClick(View view) {
        if (view == findViewById(R.id.fab)) {
            Intent intent = new Intent(this, VistoriaDetalhe.class);
```

```

        intent.putExtra("vistoria_Id", 0);
        startActivity(intent);
    } else {
        updateList(this);
    }
}

/**
 * Método que carrega e atualiza a lista (ListView)
 * @param view
 */
public void updateList(MainActivity view) {
    VistoriaRepo repo = new VistoriaRepo(this); //Instancia um objeto VistoriaRepo para acessar o
    banco de dados
    ArrayList<HashMap<String, String>> vistoriaList = repo.getVistoriaList(); //Armazena as vistorias
    em uma lista do tipo Hashmap

    lv = (ListView) findViewById(R.id.vistorialist); //Referenciar widget por seu id
    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() { //Configura a resposta ao clique
    em um dos itens da lista
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            vistoria_Id = (TextView) view.findViewById(R.id.vistoria_Id); //Referenciar widget por seu id
            String vistoriald = vistoria_Id.getText().toString(); //Armazena o Id do item clicado em uma
String
            Intent objIntent = new Intent(getApplicationContext(), VistoriaDetalhe.class); //Cria um Intent
para a Activity VistoriaDetalhe
            objIntent.putExtra("vistoria_Id", Integer.parseInt(vistoriald)); //Insere no Intent a Id da vistoria
            startActivity(objIntent); //Inicia a Activity a partir do Intent

        }
    });
    adapter = new SimpleAdapter(MainActivity.this, vistoriaList, R.layout.view_vistoria_entrada, new
String[]{"id", "embarcacao", "data"}, new int[]{R.id.vistoria_Id, R.id.vistoria_embarcacao,
R.id.vistoria_data}); //Instancia o Adapter a ser utilizado no ListView
    lv.setAdapter(adapter); //Configura o Adapter no ListView

    if (vistoriaList.size() == 0) { //Se a lista estiver vazia mostra uma mensagem
        Toast.makeText(this, "Lista vazia!", Toast.LENGTH_SHORT).show();
    }
}

/**
 * Método chamado quando a Activity é criada
 * @param savedInstanceState
 */
@Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main); //Configura o arquivo de layout para esta Activity
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //Referenciar widget por seu id
    setSupportActionBar(toolbar); //Inicializa a barra superior do App

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab); //Referenciar widget por
    seu id
    fab.setOnClickListener(this); //Configura o Listener para o botão flutuante

    SharedPreferences preferences = getSharedPreferences("prefs", MODE_PRIVATE);
    nome = preferences.getString("nome", "");
    empresa = preferences.getString("empresa", "");

    if (nome == "") {
        profileEditor();
    }

    updateList(this); //Atualiza a lista ao iniciar
}

/**
 * Inicializa o menu na action bar
 * @param menu
 * @return
 */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu); //Inicializa o menu da barra superior (Action
    bar)
    return true;
}

/**
 * Método que gerencia os clicks no menu da barra superior (action bar)
 * @param item
 * @return
 */
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId(); //Obtem o Id do item clicado

    if (id == R.id.action_about) { //Se o usuário clicou no item "Sobre" mostrar uma mensagem
        new AlertDialog.Builder(this).
            setMessage("Aplicativo para checklist de vistorias baseado nas Normas da Autoridade
            Maritima (NORMAM 02, 2005)\n\n" +
                "\nDesenvolvimento: Ademir Marques Junior e Jéssica Garcia." +

```

```

        "\nVersão: 1.0").
        setPositiveButton("OK", new DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
                //Nada a fazer
            }
        }).
        show();
        return true;
    }

    if (id == R.id.action_profile) {
        profileEditor();
    }

    return super.onOptionsItemSelected(item);
}

/**
 * Método chamado quando a Activity volta a ser exibida
 */
@Override
public void onResume() {
    super.onResume();
    updateList(this); //Atualizar a lista quando voltar a Activity
}

/**
 * Método chamado quando a Activity reinicia
 */
@Override
public void onRestart() {
    super.onRestart();
    updateList(this); //Atualizar a lista quando voltar a Activity
}

public void profileEditor () {
    final SharedPreferences preferences = getSharedPreferences("prefs", MODE_PRIVATE);
    final SharedPreferences.Editor editor = preferences.edit();
    LayoutInflater inflater = getLayoutInflater();
    View alertlayout = inflater.inflate(R.layout.perfil, null);

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Edite o seu perfil");
    builder.setMessage("Insira o seus dados abaixo:");
    builder.setView(alertlayout);
    final EditText nomeEditText = (EditText) alertlayout.findViewById(R.id.nomeText);

```

```

final EditText empresaEditText = (EditText) alertlayout.findViewById(R.id.empresaText);

empresaEditText.setText(preferences.getString("empresa", ""));
nomeEditText.setText(preferences.getString("nome", ""));

builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        editor.putString("nome", String.valueOf(nomeEditText.getText()));
        editor.putString("empresa", String.valueOf(empresaEditText.getText()));
        editor.apply();
    }
});
builder.setNegativeButton("Cancela", null);

AlertDialog dialog = builder.create();
dialog.show();

}

}

```

VistoriaDetalhe.java

```

package com.example.adeju.jessica;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class VistoriaDetalhe extends AppCompatActivity implements View.OnClickListener {

    //Inicialização de variáveis globais
    EditText editTextEmbarcacao;
    EditText editTextData;
    Spinner spinner;
    private int _Vistoria_Id = 0;

```

```

/**
 * Método chamado quando a classe é criada
 * @param savedInstanceState
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_vistoria_detalhe); //Configura o arquivo de layout para esta
Activity

    //Inicialização do menu de ferramentas superior
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //Referenciar widget por seu id
    setSupportActionBar(toolbar);
    //Ativar o botão voltar no menu de ferramentas
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setDisplayShowHomeEnabled(true);

    editTextEmbarcacao = (EditText) findViewById(R.id.editTextEmbarcacao);
    editTextData = (EditText) findViewById(R.id.editTextData);

    FloatingActionButton fab_del = (FloatingActionButton) findViewById(R.id.fab_del); //Referenciar
widget por seu id
    fab_del.setOnClickListener(this); //Configurar listener para aguardar click do usuário

    FloatingActionButton fab_save = (FloatingActionButton) findViewById(R.id.fab_save); //Referenciar
widget por seu id
    fab_save.setOnClickListener(this); //Configurar listener para aguardar click do usuário

    FloatingActionButton fab_open = (FloatingActionButton) findViewById(R.id.fab_open); //Referenciar
widget por seu id
    fab_open.setOnClickListener(this); //Configurar listener para aguardar click do usuário

    spinner = (Spinner) findViewById(R.id.spinner); //Referenciar widget por seu id
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
        R.array.tipos_vistoria, android.R.layout.simple_spinner_item); //Configurar adaptador e
carregar lista de tipos de vistoria
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item); //Especifica
o tipo de layout do spinner
    spinner.setAdapter(adapter); //Configurar o adapter ao spinner

    //Configurar o Intent filter para receber os valores passados
    _Vistoria_Id = 0;
    Intent intent = getIntent();
    _Vistoria_Id = intent.getIntExtra("vistoria_Id", 0);

    //Inicializar um objeto VistoriaRepo com os valores passados
    VistoriaRepo repo = new VistoriaRepo(this);
    Vistoria vistoria = repo.getVistoriaById(_Vistoria_Id);

```



```

    if (_Vistoria_Id == 0){ //Se for uma nova vistoria, desabilita o botão abrir
        fab_open.setEnabled(false); //Desabilita o botão
        fab_open.setVisibility(View.INVISIBLE); //Deixa o botão invisível
    }

    editTextEmbarcacao.setText(vistoria.embarcacao); //Altera o campo de texto com o valor
correspondente no objeto

    DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm"); //Configura o formato de
data e hora
    String formattedDateString = formatter.format(vistoria.data); //Formata a data e hora do objeto
    editTextData.setText(String.valueOf(formattedDateString)); //Altera o campo de texto com o valor
correspondente no objeto

    spinner.setSelection(vistoria.tipo); //Altera o item selecionado com o valor correspondente no
objeto
}

/**
 * Método para gerenciar os toques na tela
 * @param view
 */
public void onClick(View view) {
    if (view == findViewById(R.id.fab_save)) { //Se usuário clicar no botão salvar, as alterações são
salvas e o checklist é aberto

        //Inicializar as variáveis e objetos a serem utilizados
        VistoriaRepo repo = new VistoriaRepo(this);
        Vistoria vistoria = new Vistoria();
        vistoria.embarcacao = editTextEmbarcacao.getText().toString();
        vistoria.data = System.currentTimeMillis();
        vistoria.tipo = spinner.getSelectedItemPosition();
        vistoria.id = _Vistoria_Id;

        if (vistoria.embarcacao.isEmpty()) { //Verifica se o campo com o nome da vistoria está vazio
            Toast.makeText(this, "Insira todos os campos", Toast.LENGTH_SHORT).show();
            return;
        } else {

            if (_Vistoria_Id == 0) {
                _Vistoria_Id = repo.insert(vistoria);

                Toast.makeText(this, "Nova vistoria salva", Toast.LENGTH_SHORT).show();
            } else {

                repo.update(vistoria);
                Toast.makeText(this, "Dados de vistoria atualizados", Toast.LENGTH_SHORT).show();
            }
        }
    }
}

```

```

Intent intent = new Intent(this, Checklist.class); //Cria o Intent para chamar a Activity Checklist
intent.putExtra("vistoria_Id", _Vistoria_Id); //Insere as variáveis e valores no Intent
intent.putExtra("tipo", spinner.getSelectedItemPosition()); //Insere as variáveis e valores no
Intent

startActivity(intent); //Inicializa a Activity com o Intent criado
finish(); //Encerra a Activity atual

    }

    } else if (view == findViewById(R.id.fab_open)) { //Se o usuário clicar no botão abrir, abre o checklist
sem salvar as alterações

        Intent intent = new Intent(this, Checklist.class); //Cria o Intent para chamar a Activity Checklist
        intent.putExtra("vistoria_Id", _Vistoria_Id); //Insere as variáveis e valores no Intent
        intent.putExtra("tipo", spinner.getSelectedItemPosition()); //Insere as variáveis e valores no
Intent

        startActivity(intent); //Inicializa a Activity com o Intent criado
        finish(); //Encerra a Activity atual

    } else if (view == findViewById(R.id.fab_del)) { //Se o usuário clica no botão deletar, uma confirmação
é exibida

        //Inicializa o popup de confirmação
        new AlertDialog.Builder(this)
            .setTitle("Atenção!")
            .setMessage("Você deseja apagar esta vistoria?")
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int whichButton) { //Se for confirmado
                    VistoriaRepo repo = new VistoriaRepo(VistoriaDetalhe.this); //Instancia o objeto
                    repo.delete(_Vistoria_Id); //Apaga o objeto no banco de dados
                    Toast.makeText(VistoriaDetalhe.this, "Vistoria apagada", Toast.LENGTH_SHORT);
//Exibe mensagem de sucesso

                    finish(); //Encerra a Activity atual
                }
            })
            .setNegativeButton(android.R.string.no, null).show(); //Se o usuário cancela, nada é feito
    }

}

}

```

Checklist.java

```
package com.example.adeju.jessica;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class VistoriaDetalhe extends AppCompatActivity implements View.OnClickListener {

    //Inicialização de variáveis globais
    EditText editTextEmbarcacao;
    EditText editTextData;
    Spinner spinner;
    private int _Vistoria_Id = 0;

    /**
     * Método chamado quando a classe é criada
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_vistoria_detalhe); //Configura o arquivo de layout para esta
Activity

        //Inicialização do menu de ferramentas superior
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //Referenciar widget por seu id
        setSupportActionBar(toolbar);

        //Ativar o botão voltar no menu de ferramentas
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        editTextEmbarcacao = (EditText) findViewById(R.id.editTextEmbarcacao);
        editTextData = (EditText) findViewById(R.id.editTextData);

        FloatingActionButton fab_del = (FloatingActionButton) findViewById(R.id.fab_del); //Referenciar
widget por seu id
    }
```

```

fab_del.setOnClickListener(this); //Configurar listener para aguardar click do usuário

FloatingActionButton fab_save = (FloatingActionButton) findViewById(R.id.fab_save); //Referenciar
widget por seu id
fab_save.setOnClickListener(this); //Configurar listener para aguardar click do usuário

FloatingActionButton fab_open = (FloatingActionButton) findViewById(R.id.fab_open); //Referenciar
widget por seu id
fab_open.setOnClickListener(this); //Configurar listener para aguardar click do usuário


spinner = (Spinner) findViewById(R.id.spinner); //Referenciar widget por seu id
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
    R.array.tipos_vistoria, android.R.layout.simple_spinner_item); //Configurar adaptador e
carregar lista de tipos de vistoria
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item); //Especifica
o tipo de layout do spinner
spinner.setAdapter(adapter); //Configurar o adapter ao spinner


//Configurar o Intent filter para receber os valores passados
_Vistoria_Id = 0;
Intent intent = getIntent();
_Vistoria_Id = intent.getIntExtra("vistoria_Id", 0);


//Inicializar um objeto VistoriaRepo com os valores passados
VistoriaRepo repo = new VistoriaRepo(this);
Vistoria vistoria = repo.getVistoriaById(_Vistoria_Id);


if (_Vistoria_Id == 0){ //Se for uma nova vistoria, desabilita o botão abrir
    fab_open.setEnabled(false); //Desabilita o botão
    fab_open.setVisibility(View.INVISIBLE); //Deixa o botão invisível
}


editTextEmbarcacao.setText(vistoria.embarcacao); //Altera o campo de texto com o valor
correspondente no objeto


DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm"); //Configura o formato de
data e hora
String formattedDateString = formatter.format(vistoria.data); //Formata a data e hora do objeto
editTextData.setText(String.valueOf(formattedDateString)); //Altera o campo de texto com o valor
correspondente no objeto


spinner.setSelection(vistoria.tipo); //Altera o item selecionado com o valor correspondente no
objeto
}

/**
 * Método para gerenciar os toques na tela
 * @param view

```

```

*/
public void onClick(View view) {
    if (view == findViewById(R.id.fab_save)) { //Se usuário clicar no botão salvar, as alterações são
salvas e o checklist é aberto

        //Inicializar as variáveis e objetos a serem utilizados
        VistoriaRepo repo = new VistoriaRepo(this);
        Vistoria vistoria = new Vistoria();
        vistoria.embarcacao = editTextEmbarcacao.getText().toString();
        vistoria.data = System.currentTimeMillis();
        vistoria.tipo = spinner.getSelectedItemId();
        vistoria.id = _Vistoria_Id;

        if (vistoria.embarcacao.isEmpty()) { //Verifica se o campo com o nome da vistoria está vazio
            Toast.makeText(this, "Insira todos os campos", Toast.LENGTH_SHORT).show();
            return;
        } else {

            if (_Vistoria_Id == 0) {
                _Vistoria_Id = repo.insert(vistoria);

                Toast.makeText(this, "Nova vistoria salva", Toast.LENGTH_SHORT).show();
            } else {

                repo.update(vistoria);
                Toast.makeText(this, "Dados de vistoria atualizados", Toast.LENGTH_SHORT).show();
            }

            Intent intent = new Intent(this, Checklist.class); //Cria o Intent para chamar a Activity Checklist
            intent.putExtra("vistoria_Id", _Vistoria_Id); //Insere as variáveis e valores no Intent
            intent.putExtra("tipo", spinner.getSelectedItemId()); //Insere as variáveis e valores no
Intent
            startActivity(intent); //Inicializa a Activity com o Intent criado
            finish(); //Encerra a Activity atual

        }
    } else if (view == findViewById(R.id.fab_open)) { //Se o usuário clicar no botão abrir, abre o checklist
sem salvar as alterações

        Intent intent = new Intent(this, Checklist.class); //Cria o Intent para chamar a Activity Checklist
        intent.putExtra("vistoria_Id", _Vistoria_Id); //Insere as variáveis e valores no Intent
        intent.putExtra("tipo", spinner.getSelectedItemId()); //Insere as variáveis e valores no
Intent
        startActivity(intent); //Inicializa a Activity com o Intent criado
        finish(); //Encerra a Activity atual

    } else if (view == findViewById(R.id.fab_del)) { //Se o usuário clica no botão deletar, uma confirmação
é exibida

```

```

//Inicializa o popup de confirmação
new AlertDialog.Builder(this)
    .setTitle("Atenção!")
    .setMessage("Você deseja apagar esta vistoria?")
    .setIcon(android.R.drawable.ic_dialog_alert)
    .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {

        public void onClick(DialogInterface dialog, int whichButton) { //Se for confirmado
            VistoriaRepo repo = new VistoriaRepo(VistoriaDetalhe.this); //Instancia o objeto
            repo.delete(_Vistoria_Id); //Apaga o objeto no banco de dados
            Toast.makeText(VistoriaDetalhe.this, "Vistoria apagada", Toast.LENGTH_SHORT);
//Exibe mensagem de sucesso

            finish(); //Encerra a Activity atual
        }
    })
    .setNegativeButton(android.R.string.no, null).show(); //Se o usuário cancela, nada é feito
}
}
}

```

DBHelper.java

```

package com.example.adeju.jessica;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 13;

    private static final String DATABASE_NAME = "crud.db";

    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    /**
     * Método chamado na criação da classe
     * @param db
     */
    @Override
    public void onCreate(SQLiteDatabase db) { //Cria e insere dados nas tabelas do banco de dados

```

```
String CREATE_TABLE_ITEM = "CREATE TABLE " + Item.TABLE + "("
    + Item.KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT ,"
    + Item.KEY_TEXT + " TEXT, "
    + Item.KEY_TIPO + " INT )";

db.execSQL(CREATE_TABLE_ITEM);

db.execSQL("INSERT INTO " + Item.TABLE +
    "(" + Item.KEY_ID + " , " + Item.KEY_TEXT + " , " + Item.KEY_TIPO + " ) VALUES " +
    "(NULL, 'Verificar visualmente a embarcação para assegurar se não foram feitas alterações no
casco e/ou nas superestruturas que possam alterar a borda livre anteriormente atribuída - NORMAM
02/DPC Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar o estado de conservação e a manutenção da proteção de aberturas das
condições de estanqueidade dos(s) porão(ões) de carga - NORMAM 02/DPC Capítulo 6 Item 0631 C)3)',
5)," +
    "(NULL, 'Verificar o estado de conservação e a manutenção da proteção de aberturas das
condições de estanqueidade dos escotilhões de acesso abaixo do convés de borda livre. O fechamento
de um escotilhão deverá ser necessariamente efetuado por intermédio de tampas com atracadores
permanentemente fixados - NORMAM 02/DPC Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar o estado de conservação e manutenção da proteção de aberturas das
condições de estanqueidade da(s) porta(s) de visita - NORMAM 02/DPC Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar o estado de conservação e a manutenção da proteção de aberturas das
condições de estanqueidade das vigias e olhos de boi existentes nos costados abaixo do convés de
borda livre - NORMAM 02/DPC Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar o estado de conservação e a manutenção da proteção de aberturas das
condições de estanqueidade dos alboios para iluminação e/ou ventilação natural - NORMAM 02/DPC
Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar o estado de conservação das balaustradas e das bordas falsas das partes
expostas dos conveses de borda livre e de superestrutura - NORMAM 02/DPC Capítulo 6 Item 0631 C)3)',
5)," +
    "(NULL, 'Verificar se as saídas de água se encontram permanentemente desobstruídas -
NORMAM 02/DPC Capítulo 6 Item 0631 C)3)', 5)," +
    "(NULL, 'Verificar se a altura da(s) balaustrada(s) e/ou borda(s) falsa(s) são > ou = a 1,00 m e
também verificar se a abertura inferior da(s) balaustradas(s) apresentam altura < ou = 230 mm e se os
demais vãos da(s) balaustrada(s) apresentam altura < ou = 380 mm - NORMAM 02/DPC Capítulo 4 Item
0427 b)c)', 5)," +
    "(NULL, 'Verificar a existência de uma passagem permanentemente desobstruída de proa a
popa da embarcação, a qual não poderá ser efetivada por cima de tampas e escotilhas - NORMAM
02/DPC Capítulo 6 Item 0611 g)1)', 5)," +
    "(NULL, 'Verificar se as marcas de borda livre estão permanentemente fixadas em ambos os
bordos da embarcação sendo que para embarcação de aço, as marcas devem ser soldados ou buriladas
de forma permanente - NORMAM 02/DPC Capítulo 6 Item 0625 a)', 5)," +
    "(NULL, 'Verificar se as marcas estão pintadas em branco ou amarelo quando fixadas em
fundo escuro ou em preto com fundo claro - NORMAM 02/DPC Capítulo 6 Item 0625 b)', 5)," +
    "(NULL, 'Verificar se todas as marcas de borda livre estão facilmente visíveis ou se
necessário arranjo especial podem ser feito com este propósito. - NORMAM 02/DPC Capítulo 6 Item 0625
b)', 5)," +
    "(NULL, 'Verificar se a posição das marcas de borda livre se encontra de acordo com o
estabelecido no Certificado Nacional de Borda Livre em vigor - NORMAM 02/DPC Capítulo 6 Item 0631
c)3)IV', 5)," +
    "(NULL, 'Verificar se as tampas das aberturas de escotilha, dos escotilhões e seus receptivos
dispositivos de fechamento, quando existentes, tem resistência suficiente que permita satisfazer as
condições de estanqueidade previstas para o tipo de embarcação considerada e deverão apresentar
```

todos os elementos necessários para assegurar a estanqueidade - NORMAM 02/DPC Capítulo 6 Item 0611 e 0612', 5)," +

"(NULL, 'Verificar se o(s) suspiro(s) localizado(s) acima do convés de borda livre é em forma de U invertido e possuam, no mínimo, uma altura de 450 mm, medidos da aresta inferior do suspiro até o convés de borda livre - NORMAM 02/DPC Capítulo 6 Item 0611 d)1)a)b)', 5)," +

"(NULL, 'Verificar se as soleiras de portas, braçolas de escotilhas, escotilhões ou qualquer elemento que de acesso ao interior do casco possuam altura mínima de 150 mm (quando em ÁREA 1) e 260 mm (quando em ÁREA 2) - NORMAM 02/DPC Capítulo 6 Item 0611 e 0612', 5)," +

"(NULL, 'Verificar a condição de estanqueidade das anteparas estanques e se as portas de visita, nelas instaladas, estão devidamente aparafusadas e em condições estanques - NORMAM 02/DPC Capítulo 6', 5)," +

"(NULL, 'Verificar a bordo o ponto de alagamento, caso a mesma o possua, analisar se o mesmo está de acordo com o informado no Estudo de Estabilidade e representado no Plano de Arranjo Geral - NORMAM 02/DPC Capítulo 6 Item 0603 x)', 5)"

);

```
String CREATE_TABLE_VISTORIA = "CREATE TABLE IF NOT EXISTS " + Vistoria.TABLE + "("
    + Vistoria.KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT ,"
    + Vistoria.KEY_EMBARCACAO + " TEXT, "
    + Vistoria.KEY_TIPO + " INT, "
    + Vistoria.KEY_DATA + " DATETIME DEFAULT CURRENT_TIMESTAMP )";
db.execSQL(CREATE_TABLE_VISTORIA);
```

```
String CREATE_TABLE_CHECK = "CREATE TABLE IF NOT EXISTS " + CheckedItem.TABLE + "("
    + CheckedItem.KEY_ID_ITEM + " INTEGER NOT NULL, "
    + CheckedItem.KEY_ID_TIPO + " INTEGER NOT NULL, "
    + CheckedItem.KEY_ID_VISTORIA + " INTEGER NOT NULL, "
    + "PRIMARY KEY " +
    + "(" + CheckedItem.KEY_ID_ITEM + ", " + CheckedItem.KEY_ID_TIPO + ", " +
    + CheckedItem.KEY_ID_VISTORIA + ")";
db.execSQL(CREATE_TABLE_CHECK);
}
```

/**

* Método chamado quando há alteração na versão do banco de dados

* @param db

* @param oldVersion

* @param newVersion

*/

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

//Exclui as tabelas

```
db.execSQL("DROP TABLE IF EXISTS " + Item.TABLE);
```

```
db.execSQL("DROP TABLE IF EXISTS " + Vistoria.TABLE);
```

```
db.execSQL("DROP TABLE IF EXISTS " + CheckedItem.TABLE);
```

//Cria novas tabelas

```
onCreate(db);
```



```

    }

}

```

CustomAdapter.java

```

package com.example.adeju.jessica;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

import java.util.ArrayList;

public class CustomAdapter extends BaseAdapter {

    //Inicialização das variáveis globais
    ArrayList<ItemCheck> mylist = new ArrayList<>();
    private Context mContext;
    private int _Vistoria_Id;
    private int _Tipo_Id;

    /**
     * Construtor da classe
     * @param itemArray Item em uma lista
     * @param mContext Contexto atual
     * @param vistoria Id da vistoria
     * @param tipo tipo devistoria
     */
    public CustomAdapter(ArrayList<ItemCheck> itemArray, Context mContext, int vistoria, int tipo) {
        super();
        this.mContext = mContext;
        mylist = itemArray;
        _Vistoria_Id = vistoria;
        _Tipo_Id = tipo;
    }

    /**
     * Método para obter o tamanho da lista
     * @return Tamanho da lista
     */
    @Override

```

```

public int getCount() {
    return mylist.size();
}

/**
 * Método que retorna um item da lista de acordo com a sua posição
 * @param position
 * @return A posição de um item em uma string
 */
@Override
public String getItem(int position) {
    return mylist.get(position).toString();
}

/**
 * Metodo que retorna o Id de um item na lista
 * @param position
 * @return A posição de um item
 */
@Override
public long getItemId(int position) {
    return position;
}

public void onItemSelected(int position) {
}

/**
 * Obtem uma view em um ponto específico de acordo com o conjunto de dados
 * @param position
 * @param convertView
 * @param parent
 * @return Uma view em cache
 */
@Override
public View getView(final int position, View convertView,
                    ViewGroup parent) {
    // TODO Auto-generated method stub
    ViewHolder view = null; //Instancia uma view nula
    LayoutInflater inflater = ((Activity) mContext).getLayoutInflater(); //Instancia um objeto
    //LayoutInflater baseado no contexto
    if (view == null) {
        view = new ViewHolder(); //Atribui um objeto ViewHolder a view
        convertView = inflater.inflate(R.layout.view_checklist_item, null); //Expand a view (linha em uma
        //lista) e atribui a convertView
        view.nametext = (TextView) convertView.findViewById(R.id.item_text); //Referenciar widget por
        //seu id
    }
}

```

```

        view.tick = (CheckBox) convertView.findViewById(R.id.item_check); //Referenciar widget por seu
id
        //final ViewHolder finalView = view; //Atribui view a um objeto final

        final CheckedItemRepo checkedItemRepo = new CheckedItemRepo(this.mContext); //Instância um
objeto CheckedItemRepo para referência a itens já checados no banco de dados
        final CheckedItem row = new CheckedItem(); //Instancia um objeto CheckedItem com o nome de
row
        row.id_vistoria = _Vistoria_Id; //Atribui valores de atributos ao objeto row
        row.id_item = position; //Atribui valores de atributos ao objeto row
        row.id_tipo = _Tipo_Id; //Atribui valores de atributos ao objeto row
        view.tick.setChecked(checkedItemRepo.isChecked(row)); //Verifica se o checkbox deve estar
marcado ou desmarcado
        //finalView.tick.setChecked(true);

        view.tick.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
//Configura a resposta aos toques no checkbox - marcado/desmarcado
            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                                         boolean isChecked) {
                int getPosition = (Integer) buttonView.getTag(); //Obtem a posição do checkbox
                myList.get(getPosition).setChecked(buttonView.isChecked()); //Atribui a isChecked a
interação do usuário
                if (isChecked) {
                    checkedItemRepo.Check(row); //Salva o item checado no banco de dados
                } else {
                    checkedItemRepo.unCheck(row); //Exclui o item checado do banco de dados
                }
            }
        });
        convertView.setTag(view); //Atribui a convertView a view atual
    } else {
        view = (ViewHolder) convertView.getTag(); //Atribui a view a tag de convertView
    }
    view.tick.setTag(position); //Obtem a posição para o checkbox
    view.nametext.setText("" + myList.get(position).getTitle()); //Obtem o texto para a linha da lista
    view.tick.setChecked(myList.get(position).isChecked()); //Marca ou desmarca o checkbox
    return convertView; //Retorna a view convertView
}

/**
 * Configura a classe a ser exibida na view
 */
public class ViewHolder {
    public TextView nametext;
    public CheckBox tick;
}

/**

```

```

    * Retorna a lista gerada na classe
    */
    public ArrayList<ItemCheck> getMylist() {
        return mylist;
    }
}

```

VistoriaRepo.java

```
package com.example.adeju.jessica;
```

```

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

```

```

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;

```

```
public class VistoriaRepo {
```

```
    private DBHelper dbHelper;
```

```

    public VistoriaRepo(Context context) {
        dbHelper = new DBHelper(context);
    }

```

```

/**
 * Insere uma vistoria no banco de dados.
 * @param vistoria
 * @return O Id da vistoria
 */

```

```
    public int insert(Vistoria vistoria) {
```

```
        SQLiteDatabase db = dbHelper.getWritableDatabase(); //Instancia o banco de dados em modo de escrita
```

```
        ContentValues values = new ContentValues(); //Instancia um objeto ContentValues a ser inserido no banco de dados
```

```
        values.put(Vistoria.KEY_EMBARCACAO, vistoria.embarcacao); //Insere no objeto values um atributo recebido
```

```
        values.put(Vistoria.KEY_TIPO, vistoria.tipo); //Insere no objeto values um atributo recebido
```

```
        values.put(Vistoria.KEY_DATA, vistoria.data); //Insere no objeto values um atributo recebido
```

```
        long vistoria_id = db.insert(Vistoria.TABLE, null, values); //Insere a vistoria no banco de dados
```

```

        db.close(); //Fecha o banco de dados
        return (int) vistoria_id; //Retorna o Id da vistoria inserida
    }

    /**
     * Deleta a vistoria do banco de dados
     * @param vistoria_id
     */
    public void delete(int vistoria_id) {
        SQLiteDatabase db = dbHelper.getWritableDatabase(); //Instancia o banco de dados em modo de
        escrita
        db.delete(Vistoria.TABLE, Vistoria.KEY_ID + "=?", new String[]{String.valueOf(vistoria_id)});
        //Deleta a vistoria do banco de dados
        db.delete(CheckedItem.TABLE, CheckedItem.KEY_ID_VISTORIA + "=?", new
        String[]{String.valueOf(vistoria_id)}); //Deleta os itens de checklist checados para esta vistoria
        db.close(); //Fecha o banco dados
    }

    /**
     * Atualiza os dados da vistoria
     * @param vistoria
     */
    public void update(Vistoria vistoria) {

        SQLiteDatabase db = dbHelper.getWritableDatabase(); //Instancia o banco de dados em modo de
        escrita
        ContentValues values = new ContentValues(); //Instancia um objeto ContentValues a ser inserido no
        banco de dados
        values.put(Vistoria.KEY_TIPO, vistoria.tipo); //Insere no objeto values um atributo recebido
        values.put(Vistoria.KEY_EMBARCACAO, vistoria.embarcacao); //Insere no objeto values um
        atributo recebido
        values.put(Vistoria.KEY_DATA, vistoria.data); //Insere no objeto values um atributo recebido

        db.update(Vistoria.TABLE, values, Vistoria.KEY_ID + "= ?", new String[]{String.valueOf(vistoria.id)});
        //Atualiza a vistoria no banco dados com os valores recebidos
        db.close(); //Fecha o banco de dados
    }

    /**
     * Gera uma lista de vistorias
     * @return Lista HashMap com informações das vistorias obtidas no banco de dados
     */
    public ArrayList<HashMap<String, String>> getVistoriaList() {

        SQLiteDatabase db = dbHelper.getReadableDatabase(); //Instancia o banco de dados em modo de
        leitura

```

```

//A string com a consulta ao banco de dados é montada
String selectQuery = "SELECT " +
    Vistoria.KEY_ID + "," +
    Vistoria.KEY_TIPO + "," +
    Vistoria.KEY_EMBARCACAO + "," +
    Vistoria.KEY_DATA +
    " FROM " + Vistoria.TABLE + " ORDER BY " + Vistoria.KEY_DATA + " DESC";

ArrayList<HashMap<String, String>> vistoriaList = new ArrayList<HashMap<String, String>>();
////Instancia uma lista HashMap vazia para salvar as vistorias

Cursor cursor = db.rawQuery(selectQuery, null); //Salvar o retorno da consulta ao banco de dados
em um cursor

if (cursor.moveToFirst()) { //Percorrer todas as entradas do cursor e adicionar a lista
    do {
        HashMap<String, String> vistoria = new HashMap<String, String>(); //Instancia um objeto
HashMap de nome vistoria
        vistoria.put("id", cursor.getString(cursor.getColumnIndex(Vistoria.KEY_ID))); //Insere no
objeto vistoria um atributo da entrada do cursor atual
        vistoria.put("embarcacao",
cursor.getString(cursor.getColumnIndex(Vistoria.KEY_EMBARCACAO))); //Insere no objeto vistoria um
atributo da entrada do cursor atual
        vistoria.put("tipo", cursor.getString(cursor.getColumnIndex(Vistoria.KEY_TIPO))); //Insere no
objeto vistoria um atributo da entrada do cursor atual

        //Configura o formato de data e hora
        DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy - HH:mm");
        String formattedDateString =
formatter.format(Long.valueOf(cursor.getString(cursor.getColumnIndex(Vistoria.KEY_DATA))));
        vistoria.put("data", "Salvo: " + formattedDateString); //Insere no objeto vistoria a hora
formatada

        vistoriaList.add(vistoria); //Insere o objeto vistoria na lista de vistoria

    } while (cursor.moveToNext()); //Avança dentro do cursor
}

cursor.close(); //Fecha o cursor
db.close(); //Fecha o banco de dados
return vistoriaList; //Retorna o objeto vistoriaList

}

/**
 * Retorna informações de uma vistoria pelo seu Id
 * @param Id
 * @return Um objeto vistoria

```

```

*/
public Vistoria getVistoriaById(int Id) {
    if (Id != 0) { //Se o Id recebido não for zero, a consulta no banco de dados é feita
        SQLiteDatabase db = dbHelper.getReadableDatabase(); //Instancia o banco dados em modo
        leitura

        //A string com a consulta ao banco de dados é montada
        String selectQuery = "SELECT " +
            Vistoria.KEY_ID + " , " +
            Vistoria.KEY_EMBARCACAO + " , " +
            Vistoria.KEY_TIPO + " , " +
            Vistoria.KEY_DATA +
            " FROM " + Vistoria.TABLE
            + " WHERE " +
            Vistoria.KEY_ID + "=?";

        Vistoria vistoria = new Vistoria(); //Um objeto Vistoria é instanciado

        Cursor cursor = db.rawQuery(selectQuery, new String[]{String.valueOf(Id)}); //O resultado da
        consulta é retornado em um objeto Cursor

        if (cursor.moveToFirst()) {
            do {
                vistoria.embarcacao =
                cursor.getString(cursor.getColumnIndex(Vistoria.KEY_EMBARCACAO)); //Insere no objeto vistoria um
                atributo da entrada do cursor atual
                vistoria.tipo = cursor.getInt(cursor.getColumnIndex(Vistoria.KEY_TIPO)); //Insere no objeto
                vistoria um atributo da entrada do cursor atual
                vistoria.data = cursor.getLong(cursor.getColumnIndex(Vistoria.KEY_DATA)); //Insere no
                objeto vistoria um atributo da entrada do cursor atual

            } while (cursor.moveToNext());
        }

        cursor.close(); //Fecha o cursor
        db.close(); //Fecha o banco de dados
        return vistoria; //Retorna um objeto Vistoria

    } else { //Se o Id recebido for zero, então valores padrão são retornados
        Vistoria vistoria = new Vistoria(); //Um objeto Vistoria é instanciado
        vistoria.embarcacao = ""; //O nome da vistoria recebe um texto vazio
        vistoria.data = System.currentTimeMillis(); //A vistoria recebe a hora atual
        return vistoria; //Retorna um objeto Vistoria
    }
}
}
}

```

ItemRepo.java

```
package com.example.adeju.jessica;
```

```
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
```

```
import java.util.ArrayList;
```

```
public class ItemRepo {
```

```
    private Context mContext;
    private DBHelper dbHelper;
```

```
    public ItemRepo(Context context) {
        mContext = context;
        dbHelper = new DBHelper(context);
    } //Construtor da classe
```

```
    /**
     * Obtem os item de um checklist de acordo com a categoria
     * @param cat
     * @param vistoria
     * @return
     */
```

```
    public ArrayList<ItemCheck> getItemList(int cat, int vistoria) {
```

```
        SQLiteDatabase db = dbHelper.getReadableDatabase(); //Instancia o banco de dados em modo de
        leitura
```

```
        //A string com a consulta ao banco de dados é montada
```

```
        String selectQuery = "SELECT " +
            Item.KEY_ID + "," +
            Item.KEY_TEXT + "," +
            Item.KEY_TIPO +
            " FROM " + Item.TABLE +
            " WHERE " + Item.KEY_TIPO + " = " + cat;
```

```
        ArrayList<ItemCheck> itemList = new ArrayList<ItemCheck>(); //Instancia uma lista de itens vazia
```

```
        Cursor cursor = db.rawQuery(selectQuery, null); //Salvar o retorno da consulta ao banco de dados
        em um cursor
```

```
        if (cursor.moveToFirst()) { //Percorrer todas as entradas do cursor e adicionar a lista
            do {
                final CheckedItemRepo checkedItemRepo = new CheckedItemRepo(mContext);
                final CheckedItem row = new CheckedItem();
```



```

        row.id_vistoria = vistoria;
        row.id_item = cursor.getPosition();
        row.id_tipo = cat;

        //Verificar se o item do checklist já foi marcado como checado
        ItemCheck item = new ItemCheck(cursor.getString(cursor.getColumnIndex(Item.KEY_TEXT)),
checkedItemRepo.isChecked(row));
        itemList.add(item); //Adiciona o item a lista

    } while (cursor.moveToNext()); //Avança dentro do cursor
}

cursor.close(); //fecha o cursor
db.close(); //fecha o banco de dados
return itemList; //retorna a lista com os itens do checklist
}
}

```

CheckedItemRepo.java

```

package com.example.adeju.jessica;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class CheckedItemRepo {

    private DBHelper dbHelper;

    /**
     * Construtor da classe
     * @param context
     */
    public CheckedItemRepo(Context context) {
        dbHelper = new DBHelper(context);
    }

    /**
     * Salva no banco de dados um item do checklist (quando o usuário marca checado em uma linha)
     * @param checkedItem
     * @return
     */
    public int Check(CheckedItem checkedItem) {

```

```

        SQLiteDatabase db = dbHelper.getWritableDatabase(); //Instancia o banco de dados em modo de
escrita
        ContentValues values = new ContentValues(); //Instancia um objeto ContentValues a ser inserido no
banco de dados
        values.put(CheckedItem.KEY_ID_ITEM, checkedItem.id_item); //Insere no objeto values um atributo
recebido
        values.put(CheckedItem.KEY_ID_TIPO, checkedItem.id_tipo); //Insere no objeto values um atributo
recebido
        values.put(CheckedItem.KEY_ID_VISTORIA, checkedItem.id_vistoria); //Insere no objeto values um
atributo recebido

        long checked_id = db.insert(CheckedItem.TABLE, null, values); //Salva no banco de dados usando o
método INSERT
        db.close(); //Fecha o banco de dados
        return (int) checked_id; //Retorna o Id do objeto inserido
    }

```

```

/**
 * Verifica se um item do checklist já foi inserido no banco de dados
 * @param checkedItem
 * @return
 */
public boolean isChecked(CheckedItem checkedItem) {
    SQLiteDatabase db = dbHelper.getReadableDatabase(); //Instancia o banco de dados em modo de
leitura

    //A string com a consulta ao banco de dados é montada
    String selectQuery = "SELECT " +
        CheckedItem.KEY_ID_ITEM + "," +
        CheckedItem.KEY_ID_TIPO + "," +
        CheckedItem.KEY_ID_VISTORIA +
        " FROM " + CheckedItem.TABLE +
        " WHERE " + CheckedItem.KEY_ID_ITEM + "=" + checkedItem.id_item + " AND " +
        CheckedItem.KEY_ID_TIPO + "=" + checkedItem.id_tipo + " AND " +
        CheckedItem.KEY_ID_VISTORIA + "=" + checkedItem.id_vistoria;

    Cursor cursor = db.rawQuery(selectQuery, null); //O resultado da consulta é retornado em um objeto
Cursor

    if (cursor.getCount() > 0) {
        return true; //Se foi encontrado o item retorna verdadeiro
    } else {
        return false; //Se nada encontrado retorna falso
    }
}

```

```

/**

```

```

* Exclui do banco de dados um item do checklist (quando o usuário desmarca um item em uma linha)
* @param checkedItem
*/
public void unCheck(CheckedItem checkedItem) { //Exclui um item checkado do banco de dados -
acontece quando o usuário desmarca um item
    SQLiteDatabase db = dbHelper.getWritableDatabase(); //Instancia o banco de dados em modo de
escrita

    //Deleta o item checkado do banco de dados
    db.delete(CheckedItem.TABLE, CheckedItem.KEY_ID_ITEM + "=? AND " +
        CheckedItem.KEY_ID_TIPO + "=? AND " + CheckedItem.KEY_ID_VISTORIA + "=?", new
String[]{String.valueOf(checkedItem.id_item), String.valueOf(checkedItem.id_tipo),
String.valueOf(checkedItem.id_vistoria)});
    db.close(); //Fecha o banco de dados
}

}

```

Vistoria.java

```

package com.example.adeju.jessica;

public class Vistoria { //Classe

    //Nome da tabela
    public static final String TABLE = "Vistoria";

    //Nome das colunas nas tabelas
    public static final String KEY_ID = "id";
    public static final String KEY_EMBARCACAO = "embarcacao";
    public static final String KEY_TIPO = "tipo";
    public static final String KEY_DATA = "data";

    //Propriedade dos atributos
    public int id;
    public String embarcacao;
    public int tipo;
    public long data;

}

```

Item.java

```
package com.example.adeju.jessica;

public class Item { //Classe

    //Nome da tabela
    public static final String TABLE = "Item";
    //Nome das colunas nas tabelas
    public static final String KEY_ID = "id";
    public static final String KEY_TEXT = "text";
    public static final String KEY_TIPO = "tipo";

    //Propriedade dos atributos
    public int id_item;
    public String text;
    public int tipo;
}
```

CheckedItem.java

```
package com.example.adeju.jessica;

public class CheckedItem { //classe

    //Nome da tabela
    public static final String TABLE = "CheckedItem";

    //Nome das colunas nas tabelas
    public static final String KEY_ID_VISTORIA = "id_vistoria";
    public static final String KEY_ID_ITEM = "id_item";
    public static final String KEY_ID_TIPO = "id_tipo";

    //Propriedade dos atributos
    public int id_vistoria;
    public int id_item;
    public int id_tipo;
}
```

ItemCheck.java

```
package com.example.adeju.jessica;

public class ItemCheck { //Classe

    private String title = "";
    private boolean checked = false;

    public ItemCheck(String title, boolean checked) {
        this.title = title;
        this.checked = checked;
    }

    public boolean isChecked() {
        return checked;
    }

    public void setChecked(boolean checked) {
        this.checked = checked;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
```

```

        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@color/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:backgroundTint="@android:color/holo_blue_dark"
        app:fabSize="normal"
        app:srcCompat="@android:drawable/ic_input_add" />

</android.support.design.widget.CoordinatorLayout>

```

content_main.xml

```

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">

    <ListView
        android:id="@+id/vistorialist"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </ListView>

</android.support.constraint.ConstraintLayout>

```

activity_vistoria_detalhe.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".VistoriaDetalhe">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include
        android:id="@+id/include"
        layout="@layout/content_vistoria_detalhe" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_open"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:backgroundTint="@android:color/holo_green_dark"
        app:fabSize="normal"
        app:srcCompat="@android:drawable/ic_menu_zoom" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_del"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/fab_margin"
        android:clickable="true"
        app:backgroundTint="@android:color/holo_red_dark"
```

```

app:fabSize="normal"
app:layout_anchor="@+id/include"
app:layout_anchorGravity="bottom|left"
app:srcCompat="@android:drawable/ic_menu_close_clear_cancel" />

```

```

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab_save"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="@dimen/fab_margin"
    android:clickable="true"
    app:backgroundTint="@android:color/holo_blue_dark"
    app:fabSize="normal"
    app:layout_anchor="@+id/include"
    app:layout_anchorGravity="bottom|center"
    app:srcCompat="@android:drawable/ic_menu_save" />

```

```

</android.support.design.widget.CoordinatorLayout>

```

content_vistoria_detalhe.xml

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".VistoriaDetalhe"
    tools:showIn="@layout/activity_vistoria_detalhe">

```

```

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```

<ImageView
    android:id="@+id/img_ship"
    android:layout_width="121dp"
    android:layout_height="108dp"
    android:layout_gravity="top"
    app:srcCompat="@drawable/ship" />

```

```

<TextView
    android:id="@+id/textViewEmbarcacao"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

```



```

    android:layout_below="@+id/img_ship"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:text="Nome da Embarcação:" />

```

```
<EditText
```

```

    android:id="@+id/editTextEmbarcacao"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/textViewEmbarcacao"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:ems="10"
    android:inputType="textPersonName" />

```

```
<TextView
```

```

    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editTextEmbarcacao"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:text="Tipo da vistoria"

```

```
/>
```

```
<Spinner
```

```

    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/textView"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"

```

```
/>
```

```
<TextView
```

```

    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/spinner"
    android:layout_marginBottom="16dp"

```

```

    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:text="Data" />

```

```

<EditText
    android:id="@+id/editTextData"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/spinner"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:ems="10"
    android:enabled="false"
    android:inputType="date"
    tools:clickable="false" />

```

```

</RelativeLayout>

```

```

</android.support.constraint.ConstraintLayout>

```

view_checklist_item.xml

```

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/item_Id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <TextView
        android:id="@+id/item_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.8"
        android:paddingLeft="6dip"
        android:paddingTop="6dip"

```

```
android:textSize="26dp" />
```

```
<CheckBox
```

```
    android:id="@+id/item_check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginRight="10dp" />
```

```
</LinearLayout>
```

view_vistoria_entrada.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
<ImageView
```

```
    android:layout_width="101dp"
    android:layout_height="66dp"
    android:layout_margin="8dp"
    android:layout_weight="0"
    app:srcCompat="@drawable/ship" />
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="vertical">
```

```
<TextView
```

```
    android:id="@+id/vistoria_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="gone" />
```

```
<TextView
```

```
    android:id="@+id/vistoria_embarcacao"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="6dip"
```

```

        android:paddingTop="6dip"
        android:textSize="26dp"
        android:textStyle="bold" />

<TextView
    android:id="@+id/vistoria_data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="false"
    android:editable="false"
    android:padding="6dp" />

</LinearLayout>

</LinearLayout>

```

perfil.xml

```

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome"/>

    <EditText
        android:id="@+id/nomeText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Empresa"/>

    <EditText
        android:id="@+id/empresaText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>

```

activity_checklist.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Checklist">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" >

            <Button
                android:id="@+id/rep_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="right|top"
                android:text="GERAR RELATORIO"
            />

        </android.support.v7.widget.Toolbar>

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_checklist" />

</android.support.design.widget.CoordinatorLayout>
```

content_checklist.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:context=".Checklist"
        tools:showIn="@layout/activity_checklist">

        <ListView
            android:id="@+id/checklist"
            android:layout_width="match_parent"
            android:layout_height="match_parent"

        />
    </android.support.constraint.ConstraintLayout>

```

colors.xml

```

<?xml version="1.0" encoding="utf-8"?>

<resources>
    <color name="colorPrimary">#FF5722</color>
    <color name="colorPrimaryDark">#F44336</color>
    <color name="colorAccent">#009688</color>
</resources>

```

dimens.xml

```

<resources>

    <dimen name="fab_margin">16dp</dimen>
</resources>

```

strings.xml

```

<resources>

    <string name="app_name">Vistorias Checklist App</string>
    <string name="file_provider_authority" translatable="false">
com.example.adeju.jessica.fileprovider</string>
    <string name="action_about">Sobre</string>
    <string name="action_profile">Perfil do usuário</string>
    <string name="title_activity_vistoria_detalhe">Dados da vistoria</string>
    <string-array name="tipos_vistoria">

```

```

    <item>Vistoria Inicial</item>
    <item>Vistoria Anual</item>
    <item>Vistoria Intermediária</item>
    <item>Vistoria de Renovação</item>
    <item>Vistorias Especiais - Prova de máquinas/navegação</item>
    <item>Vistorias Especiais - Nacional de borda livre</item>
    <item>Vistorias Especiais - Arqueação</item>
    <item>Vistorias Especiais - Laudo Pericial</item>
</string-array>
<string name="title_activity_checklist">Checklist</string>
</resources>

```

styles.xml

```

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>

    <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />

</resources>

```

file_provider_paths.xml

```

<?xml version="1.0" encoding="utf-8"?>

<paths>
    <external-path
        name="external_files" path="." />
</paths>

```