UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

Vitor Furtado Thomé

SISTEMA PARA MONITORAMENTO E RASTREIO DE EMBARCAÇÕES BASEADO EM IOT

Florianópolis
2023

Vitor Furtado Thomé

SISTEMA PARA MONITORAMENTO E RASTREIO DE EMBARCAÇÕES BASEADO EM IOT

Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia Eletrônica do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.
Orientador: Prof. Eduardo Luiz Ortiz Batista, Dr. Eng.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Thomé, Vitor
SISTEMA PARA MONITORAMENTO E RASTREIO DE EMBARCAÇÕES BASEADO EM IOT / Vitor Thomé ; orientador, Eduardo Luiz Ortiz Batista, 2023.
80 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2023.

Inclui referências.

1. Engenharia Eletrônica. 2. IoT. 3. Flutter. 4. GPRS. 5. Rastreamento. I. Batista, Eduardo Luiz Ortiz. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Vitor Furtado Thomé

SISTEMA PARA MONITORAMENTO E RASTREIO DE EMBARCAÇÕES BASEADO EM IOT

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia Eletrônica" e aprovado em sua forma final pelo Curso de Graduação em Engenharia Eletrônica.

Florianópolis, 06 de Dezembro de 2023.

ICP-Edu
Documento assinado digitalmente
Fernando Rangel de Sousa
Data: 13/12/2023 13:20:35-0300
CPF: ***.649.114-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Fernando Rangel de Sousa, Dr.
Coordenador do Curso

Banca Examinadora:

ICP-Edu
Documento assinado digitalmente
Eduardo Luiz Ortiz Batista
Data: 13/12/2023 13:46:09-0300
CPF: ***.521.889-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eduardo Luiz Ortiz Batista, Dr. Eng.
Orientador

ICP-Edu
Documento assinado digitalmente
Richard Demo Souza
Data: 13/12/2023 12:56:44-0300
CPF: ***.267.379-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Richard Demo Souza, Dr. Eng.
Avaliador UFSC/CTC/EEL

MARCELO CESAR DOS REIS:06604592995
Assinado de forma digital por MARCELO CESAR DOS REIS:06604592995
Dados: 2023.12.13 15:39:10 -03'00'

Marcelo César dos Reis, Eng.
Avaliador

Este trabalho é dedicado aos meus queridos pais e familiares,
aos meus amigos e a todos que fizeram parte desta jornada.

AGRADECIMENTOS

Quero agradecer a Deus por ter me dado saúde e uma família maravilhosa e as pessoas que estiveram ao meu lado, durante toda esta jornada, sempre me apoiando e abençoando com sua sabedoria.

Agradeço aos Professores Fernando Rangel por ter me acolhido como bolsista em seu laboratório, Prof. Orientador Eduardo Batista e ao Professor Richard Demo Souza, por todo o aprendizado e conselhos, vou lembrar para sempre com muita gratidão por terem feito parte de momentos tão especiais na minha trajetória pessoal e profissional.

Agradeço ao meu supervisor e sócio fundador da Navalcare, Marcelo César dos Reis por ter me proporcionado a oportunidade de desenvolver este trabalho, por todos os momentos que passamos juntos ao longo deste ano, que foi marcado por desafios, vitórias e aprendizados, graças a seus conselhos e seu direcionamento fui capaz de evoluir como profissional e como pessoa.

Agradeço ao sócio e Gerente Técnico da Navalcare, Ericson Meier pelo apoio, feedback e direcionamento ao longo do projeto, suas dicas foram valiosas e sem você a empresa jamais teria a qualificação e respeito que hoje conquistou. Também incluo aqui os demais funcionários da Navalcare, Lucas Barbosa, Marlon de Amorim, Andre Bueno, Cedric Gorczevski, Rubens Varani e Renata Selbach, pelo apoio e por todos os momentos que passamos juntos.

Agradeço ao designer Nilo Braga que esteve conosco durante todas as etapas deste projeto, auxiliando com suas ideias e com sua experiência, por ter nos reerguido quando mais precisávamos e aceitado fazer parte deste projeto.

Agradeço a todos os meus amigos, familiares e a todos os professores e colegas que tive a oportunidade de conhecer durante a passagem pela graduação e ao longo da vida.

Deixo aqui também um agradecimento ao professor Jacob Moura, por ter sido um marco central da minha carreira como desenvolvedor, também quero homenagear todos os membros da Masterclass Fluterando, turma 7, e aos professores Marco Tulio e Andre Hora, que estiveram junto comigo nesta jornada onde pude iniciar os aprendizados neste vasto campo que é a engenharia de software.

Por fim, quero também agradecer a Universidade Federal de Santa Catarina, seus fundadores e a todos que mantiveram em funcionamento toda a estrutura do campus ao longo dos anos, por ter me proporcionado essa experiência incrível que com certeza me transformou muito tanto como pessoa quanto como profissional.

"You can't connect the dots looking forward;
you can only connect them looking backwards.
So you have to trust that the dots will
somehow connect in your future."
Steve Jobs

RESUMO

O trabalho desenvolvido tem como foco o desenvolvimento de um protótipo de uma plataforma baseada em IoT para o sensoriamento remoto e manutenção inteligente de embarcações. Tal plataforma, denominada "Navalcare Safeboat" desenvolvida para a empresa Navalcare, visando uma solução moderna e integrada para seus clientes, além de uma série de sistemas secundários e uma aplicação móvel, denominada "Navalcare Connect", desenvolvida utilizando o framework Flutter e diversas funcionalidades da plataforma Google Cloud e Firebase, que tem como objetivo servir como ponte para conectar os dados coletados pela plataforma IoT aos clientes da empresa e a suas embarcações.

Palavras-chave: IoT, Flutter, Cloud, GPRS, Aplicações Móveis, Manutenção Inteligente.

ABSTRACT

The work developed focuses on the development of a prototype for an IoT platform for remote sensing and smart maintenance of vessels, named "Navalcare Safeboat," created for Navalcare. This solution aims to provide a modern and integrated option for its clients, in addition to a series of secondary systems and a mobile application named "Navalcare Connect." Developed using Flutter, and incorporating various features of the Google Cloud and Firebase platforms, such an application has the goal to act as a conduit to connect data collected by the IoT platform to the company's clients and their vessels.

Keywords: IoT, Flutter, Cloud, GPRS, Mobile Applications, Intelligent Maintenance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico-Digital
API	Application Programming Interface
BaaS	Backend-as-a-Service
BLoC	Business Logic Component
BoM	Bill of Materials
CI	Circuito Integrado
CI/CD	Continuous Integration / Continuous Delivery
CRUD	Create Read Update Delete
DDD	Domain-Driven Design
DRC	Design Rule Checking
FDD	Feature Driven Development
FSM	Finite State Machine
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communication
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things (internet das coisas)
JSON	JavaScript Object Notation
M2M	Machine to Machine
NB-IoT	InternetNarrowBand-Internet of Things
NMEA	National Marine Electronics Association
PCB	Printed Circuit Board
PDF	Portable Document Format
UI	User Interface
USB	Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

No contexto contemporâneo da engenharia eletrônica e da tecnologia náutica, o avanço da Internet das Coisas (IoT) oferece um terreno fértil para inovações. Este trabalho é o resultado de um esforço sistemático para explorar essa fronteira, conduzindo ao desenvolvimento de um protótipo de sistema IoT robusto para sensoriamento e monitoramento de embarcações - um reflexo da capacidade da engenharia eletrônica em responder aos desafios contemporâneos encontrados.

A problemática central abordada reside na aplicação de tecnologias como módulos GPRS e GNSS na concepção de um protótipo que servirá como base para a criação de um produto real para o mercado náutico, denominado "Navalcare Safeboat", complementado pelo aplicativo "Navalcare Connect". O projeto apresenta um sistema capaz de integrar hardware e software para oferecer uma solução de monitoramento compreensiva e confiável, visando não só a satisfação das necessidades atuais da empresa Navalcare, mas também a previsão e adaptação às demandas futuras de seus clientes.

A metodologia adotada para o desenvolvimento deste trabalho é holística e pragmática, combinando pesquisa de mercado detalhada com a aplicação de práticas de engenharia para o desenho e implementação de hardware e software. O projeto envolveu a criação de aplicativos móveis compatíveis com iOS e Android, desenvolvimento de uma base de dados eficiente e a implementação de uma API para previsão do tempo, culminando em uma plataforma versátil que oferece uma visão abrangente sobre dados relevantes para a segurança das embarcações.

Reconhecendo o papel central da experimentação e da validação em engenharia, foram realizados testes em bancada, abrangendo desde simulações até testes com o hardware e firmware desenvolvidos, API de previsão do tempo, e a funcionalidade dos aplicativos em telas de diversos tamanhos e aparelhos de diversos modelos (em múltiplos sistemas operacionais mobile), estes testes permitiram que o sistema proposto atendesse os elevados padrões de qualidade exigidos pelo setor, tendo em mente os recursos de tempo, equipe e capital disponíveis.

Este documento está estruturado de forma a refletir as diversas etapas do projeto, desde a justificativa teórica até os resultados práticos, oferecendo uma contribuição relevante ao campo da engenharia eletrônica e evidenciando a importância de uma abordagem integrada que combina hardware, software e dados em soluções de IoT.

1.1 Objetivos

Os objetivos centrais do projeto realizado podem ser divididos em dois grupos: o objetivo geral que explicam a finalidade e o escopo do projeto; e os objetivos específicos que propõe o cronograma e divisão das etapas principais envolvidas na conclusão dos requisitos funcionais propostos.

1.2 Objetivo geral

Desenvolver um protótipo de um sistema comercial e modular, de fácil instalação e baixo custo para o sensoriamento remoto de embarcações, obtendo dados relevantes para a segurança da embarcação, detecção preventiva de possíveis falhas, rastreios e leitura das tensões de diversos bancos de bateria utilizados pelos sistemas elétricos da embarcação, além de uma aplicação iOS e Android que os clientes possam utilizar para monitorar suas embarcações e interagir com outros sistemas oferecidos e criados pela empresa.

1.3 Objetivos específicos

1. Pesquisa de mercado.
2. Detalhamento completo das ferramentas, componentes e frentes do projeto.
3. Aquisição dos componentes e ferramentas necessários.
4. Testes iniciais e simulações com os componentes do hardware.
5. Escolha dos padrões de projeto e arquiteturais necessários para o desenvolvimento do software e hardware.
6. Desenvolvimento de mockups para as telas da aplicação.
7. Desenvolvimento do protótipo em hardware.
8. Desenvolvimento do firmware do sistema embarcado.
9. Desenvolvimento do fronend da aplicação.
10. Desenvolvimento do backend da aplicação.
11. Desenvolvimento do primeiro protótipo definitivo para o hardware safeboat.
12. Testes finais e ajustes.

1.4 Estrutura do trabalho

O trabalho realizado foi dividido em cinco tópicos centrais, referências bibliográficas, anexos e apêndices. Os principais pontos abordados são os seguintes:

Capítulo 1 - Introdução: Contextualização, objetivos gerais e específicos do projeto.

Capítulo 2 - Metodologia: Abordagens e processos estabelecidos para o desenvolvimento do projeto em cada setor.

Capítulo 3 - Fundamentação Teórica: Explicação sucinta sobre as principais ferramentas, padrões e detalhes relevantes abordados durante o desenvolvimento.

Capítulo 4 - Desenvolvimento: Atividades realizadas em cada frente para concepção dos protótipos, sistemas e aplicações que compõe o projeto.

Capítulo 5 - Conclusão: Resultados obtidos, testes e simulações, perspectivas para o futuro e considerações finais.

2 METODOLOGIAS

Dada a natureza multidisciplinar do projeto, diversas metodologias já reconhecidas e adotadas pelo mercado de trabalho foram implementadas para assegurar a escalabilidade futura e segurança, tendo em vista o contexto empresarial do mesmo. As metodologias adotadas para as diversas frentes do trabalho têm como foco realizar tarefas e resolver problemas de maneira sistemática e repetível, permitindo assim a criação de um ambiente de trabalho organizado, seguro, escalável e de fácil entendimento.

2.1 Metodologia para o desenvolvimento de hardware

Assim como todos os outros setores, a etapa de desenvolvimento do hardware faz uso de metodologias ágeis visando entregas contínuas e integração contínua do sistema (CI/CD)[1]. O ciclo usual para cada nova entrega de hardware definido consiste principalmente em:

1. **Análise de requisitos de sistema como:** objetivos centrais, tecnologias utilizadas, faixas de operação, interfaces, módulos e componentes necessários assim como o uso de placas de desenvolvimento para facilitar a ambientação com as tecnologias e os testes iniciais na etapa de prototipagem.

2. **Diagramação completa do sistema proposto:** visando explicitar os diversos blocos que abstraem o sistema para atender as demandas necessárias, bem como sua interconexão, servindo de apoio para a escolha dos componentes finais e a criação do firmware.

3. **Escolha e aquisição dos componentes:** a escolha correta e aquisição dos componentes é uma etapa fundamental para o desenrolar do projeto do hardware, fatores como disponibilidade, fornecedores, custo, precisão, confiabilidade, praticidade, entre outros são levados em consideração.

4. **Testes em bancada, simulações e estudos:** com o objetivo central de verificar o funcionamento do firmware e hardware propostos, componentes e robustez do protótipo, viabilizando a validação das primeiras especificações e delimitação de escopo para o projeto das versões iniciais do hardware dedicado.

5. **Criação do esquemático para o protótipo em PCB:** definição dos componentes individuais e limitações considerando as especificações que foram estabelecidas após o uso das placas de desenvolvimento e módulos na etapa inicial de testes.

6. **Avaliação final do esquemático:** buscando possíveis falhas e melhorias necessárias para atender os requisitos de projeto.

[1] Link para leitura: What is CI/CD?

7. **Criação do layout:** importação dos footprints e modelos 3D dos componentes utilizados, assim como a criação de trilhas e do layout completo.

8. **Avaliação final do layout e DRC:** conferência final das regras de design estabelecidas pelas limitações impostas pela empresa que irá se encarregar pela fabricação da PCB, bem como uma avaliação geral do layout.

9. **Envio do pedido para a fabricação:** utilizando a lista de materiais (Bill of Materials - BoM) produzida e os arquivos para posicionamento dos componentes é então requisitada a produção da primeira versão do hardware dedicado.

10. **Modelagem e impressão 3D do encapsulamento:** através do layout e dos modelos 3D desenvolvidos para a PCB, são criadas réplicas com impressão 3D para auxiliar no esboço da primeira versão do encapsulamento, enquanto a fabricante produz o produto final.

11. **Testes e validação da PCB:** após a entrega da PCB pela fabricante, conectores e outros componentes finais são soldados e então novos testes são realizados buscando a avaliação do encapsulamento e dos requisitos funcionais.

12. **Nova iteração:** por fim, repete-se o ciclo descrito acima desde as primeiras etapas para avaliar melhorias ou eventuais correções as quais estabelecem o escopo para o projeto da segunda versão.

2.2 Metodologia para o desenvolvimento de firmware

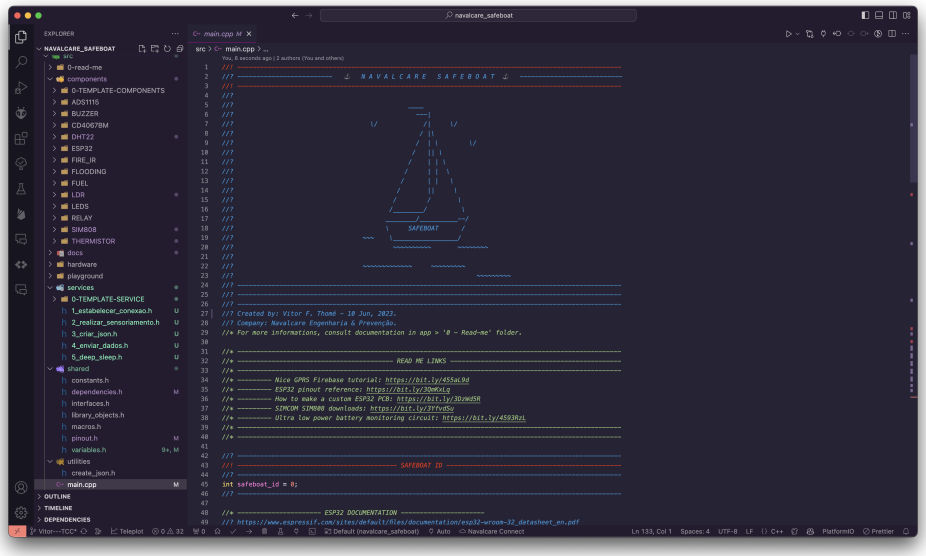
O processo para a criação do firmware segue também as diretrizes CI/CD estabelecidas pelos outros setores, boas práticas para produção de código baseado nos princípios S.O.L.I.D. (TV, 2015) e uma arquitetura criada para organização do código do projeto; a metodologia criada para este setor se baseia nas seguintes etapas:

1. **Produção dos requisitos:** a produção dos requisitos e métodos utilizados em cada um dos sub sistemas do protótipo.

2. **Criação da máquina de estados:** A criação da máquina de estados do tipo "Máquina de Mealy" onde o próximo estado é definido com base no estado atual e em variáveis de entrada.

3. **Criação da pasta "Components":** respeitando a organização estabelecidas pela arquitetura do projeto e outros conceitos como os princípios S.O.L.I.D., visando a organização do código criado e assegurando escalabilidade e organização do projeto desenvolvido.

Figura 1 – Captura de tela do software Visual Studio Code mostrando a estrutura de pastas da arquitetura de software desenvolvida para o firmware.



Fonte: Elaborado pelo autor (2023).

A pasta criada para cada um dos componentes tem como objetivo centralizar cada uma das implementações das funcionalidades relacionadas ao seu funcionamento adequado e operação, permitindo criar o código do firmware de maneira modular, facilitando testes unitários, integração e organização.

4. **Criação da pasta "Shared":** para a organização do código, bibliotecas utilizadas, mapeamento dos pinos, constantes, objetos, variáveis e interfaces do sistema embarcado.

5. **Desenvolvimento das funcionalidades de cada componente:** primeiramente é desenvolvido o código necessário para o funcionamento e operação de cada componente, seguido por testes unitários buscando aferir seu funcionamento.

6. **Desenvolvimento das rotinas da máquina de estados finitos (FSM):** dentro da pasta "Services" são criadas rotinas numeradas visando a implementação adequada da máquina de estados; para cada uma das etapas da máquina de estados são utilizadas diversas funcionalidades de cada um dos componentes utilizados no projeto.

7. **Integração dos componentes:** confecção dos loops principais para o funcionamento do protótipo do hardware, utilizando as funções "setup" e "loop" do framework Arduino, visando estabelecer o mapeamento de pinos, inicialização adequada de cada

um dos componentes utilizados e a implementação da máquina de estados finitos (FSM) no "loop" de execução do framework.

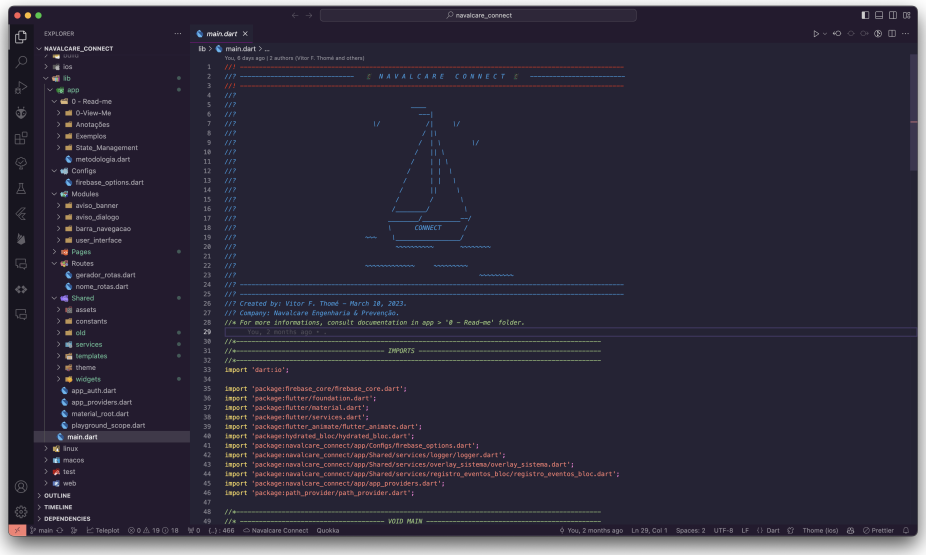
8. **Testes de integração final:** verificando o funcionamento adequado do firmware com o protótipo em bancada, correção de eventuais problemas e validando os requisitos funcionais previamente estabelecidos.

9. **Novos testes com o hardware dedicado:** o firmware produzido é validado com a PCB em mãos, então, são geradas novas demandas de requisitos para eventuais correções e melhorias observadas através dos testes práticos.

10. **Nova iteração:** após os testes finais com a versão produzida do hardware, as correções, melhorias e novos requisitos são gerados para o próximo ciclo.

2.3 Metodologia para o desenvolvimento de software - frontend

Figura 2 – Captura de tela do software Visual Studio Code mostrando a estrutura de pastas da arquitetura de software desenvolvida para o aplicativo.



Fonte: Elaborado pelo autor (2023).

Os modelos (mockups) do frontend da aplicação foram desenvolvidos com o auxílio de um designer contratado pela empresa durante reuniões semanais, então foi criado em código a arquitetura de pastas e arquivos necessária para organizar o projeto, com base nos requisitos produzidos pelo design original, deste modo o fluxograma e a metodologia para criação das telas da aplicação seguiu a seguinte ordem:

1. **Criação da pasta principal da tela sob desenvolvimento:** dentro da pasta principal "app/pages/nome_da_pagina" delimitada pela arquitetura de software desenvolvida que contém todas as telas presentes na aplicação.

2. **Criação do arquivo para as reatividades da tela atual:** criação do arquivo "nome_tela.dart" na pasta principal da tela sob desenvolvimento, que tem como responsabilidade cuidar das reatividades provenientes de outras partes do backend da aplicação e instanciar o componente de regra de negócio (BLoC) responsável pela gestão de estados da tela atual.

3. **Criação do arquivo para os elementos gráficos da tela atual:** criação do arquivo nome_tela_page.dart que é responsável por encapsular os widgets (elementos gráficos) utilizados na tela em questão.

4. **Criação da rota para a tela atual:** adição da página em desenvolvimento nas rotas da aplicação sempre apontando para o componente de mais alto nível da página, o qual possui o mesmo nome da pasta principal.

5. **Uso de componentes compartilhados:** componentes comuns entre telas como botões, containers, backgrounds e estruturas com scroll e layout similares são adicionados na tela atual e uma vez que um mesmo componente possa ser utilizado por mais de uma tela, este deve respeitar a metodologia desenvolvida e os limites arquiteturais, sendo movido para a pasta da aplicação responsável por armazenar os widgets que são utilizados em duas ou mais páginas da aplicação.

6. **Desenvolvimento dos componentes únicos:** organizados em arquivos separados dentro da pasta "widgets" da tela atual. Os widgets podem representar diversos elementos gráficos como botões, divisórias, containers, imagens, dashboards, etc. Dependendo da complexidade de alguns dos elementos que compõe a página, pode ser necessário a criação de sub pastas à pasta principal dos "widgets", esta nova pasta deve possuir o mesmo nome do arquivo principal contido em seu interior.

   Por exemplo: um elemento gráfico é responsável por apresentar dados relacionados a um dos bancos de baterias da embarcação em uma determinada tela, que por sua vez, contém uma grade com vários destes elementos repetidos para cada um dos bancos existentes na embarcação, nomeamos então este componente "Container Bateria" explicitando da melhor forma possível seu papel.

   Seguindo os padrões arquiteturais, a tela deve então conter uma pasta "nome_da_pagina/widgets" e uma sub pasta "nome_da_pagina/widgets/container_bateria". Dentro da pasta "container_bateria" estão arquivos contendo o código dos sub elementos que compõe o container do banco de baterias, como o arquivo principal "container_bateria.dart" e a pasta "container_baterias/widgets" onde os

outros elementos gráficos auxiliares são armazenados, como mostradores circulares, icones, entre outros.

Caso o componente "Container Bateria" sob desenvolvimento possua baixa complexidade, teremos apenas um arquivo "container_bateria.dart" na pasta "nome_da_pagina/widgets".

7. **Gestão de telas com sub páginas:** caso a tela atual contenha sub páginas, estas devem estar localizadas dentro de sub pastas internas a pasta denominada "views" da página principal. Deste modo a tela "Safeboat Baterias" que é uma sub página da tela "Safeboat", deve estar contida na pasta "safeboat/views/safeboat_baterias" na pasta "views" da página "Safeboat". As sub páginas seguem a mesma estrutura de pastas das páginas principais, contendo pastas como "widgets", "blocs", entre outros.

8. **Criação gradual do design system:** as cores utilizadas na da aplicação sempre são obtidas através dos códigos hexadecimais originais provenientes do mockup criado na etapa de design, sendo então armazenadas no arquivo "app/shared/constants/colors.dart". Caso um código hexadecimal de cor já exista neste arquivo, deve-se utilizar a constante já existente criada, evitando assim a adição desnecessária de novas constantes e reforçando a padronização de cores, permitindo a criação gradual de um design system (DESIGN, 2023) para o projeto.

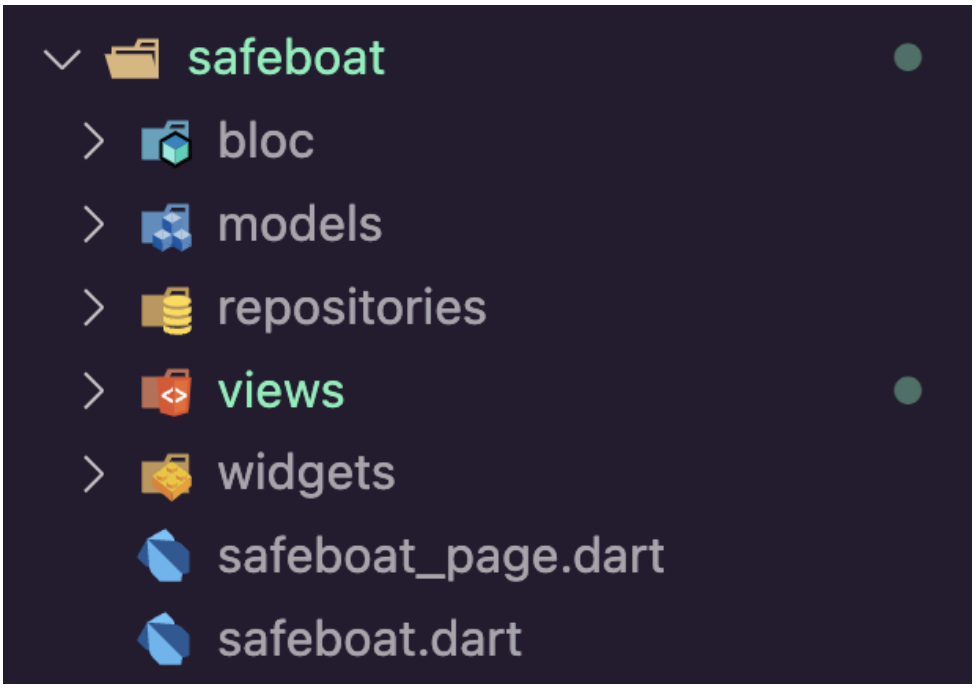
9. **Criação dos módulos principais:** por fim, alguns elementos fundamentais para a aplicação como a barra de navegação, banners de avisos, notificações, caixas de diálogos e popups são mantidos na pasta "app/modules/nome_do_modulo". Segundo a metodologia estabelecida e se diferenciando dos demais widgets pelo fato de possuírem estados e regras de negócio compartilhadas entre telas ao longo do funcionamento da aplicação.

2.4 Metodologia para o desenvolvimento de software - backend

A estruturação do backend da aplicação teve como base padrões de projeto, metodologias e arquiteturas de software reconhecidas pela indústria como Domain-Driven Design (DDD) (FOWLER, 2023), Feature Driven Development (FDD) (WIKIPEDIA, 2022) e Bussines Logic Component (BLoC) capítulo. Com o estudo das abordagens citadas, foi possível desenvolver uma organização para os arquivos e abstrações do projeto, seguindo a seguinte metodologia:

1. **Uso do pacote e padrão de projeto BLoC:** o padrão BLoC, baseado em streams de estados e eventos, foi utilizado para o gerenciamento dos estados da aplicação.

Figura 3 – Captura de tela do software Visual Studio Code mostrando a estrutura de pastas da arquitetura de software desenvolvida para o backend de cada uma das telas da aplicação.



Fonte: Elaborado pelo autor (2023).

2. **Uso de padrões de projeto:** dentre todos os padrões estudados, reconhecidos e utilizados na indústria de desenvolvimento de software, se destacam: Agregates, Entities, Repositories, Singletons, Interfaces, Adapters, CopyWith, entre outros (GURU, 2023).

3. **Organização do código:** estruturas de pastas que respeitem os padrões arquiteturais estabelecidos permitindo uma fácil distribuição, organização e escalabilidade do projeto, dentre as quais temos principalmente:

    - Pasta “Services” dentro da pasta primária “app/shared” que contém todos os serviços utilizados por diversos setores da aplicação.
    - Pasta “Bloc” dentro das pastas de alguns dos “serviços”, “módulos” e de todas as páginas da aplicação, com o foco na gestão e distribuição das regras de negócio, eventos e estados envolvidos no funcionamento da aplicação. Também é utilizado para a persistência dos estados quando o usuário estiver offline.
    - Pasta “Repositories” dentro da pasta de cada uma das telas da aplicação, responsável pelo acesso e Create Read Update Delete (CRUD) utilizando o serviço principal do banco de dados.

- Pasta “Models” dentro da pasta de cada uma das telas da aplicação, responsável por declarar as classes que são responsáveis pela modelagem dos dados que são apresentados em cada uma das telas.

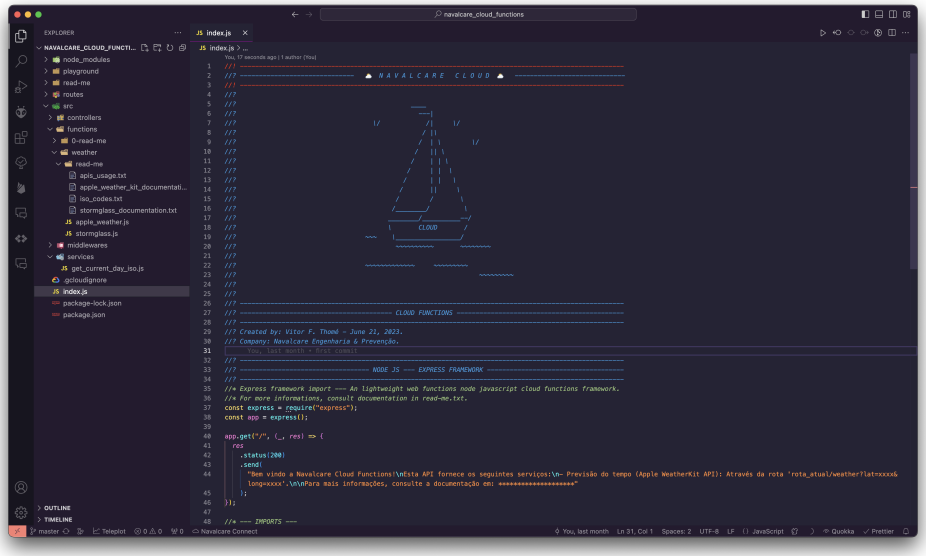
O banco de dados da aplicação foi desenvolvido utilizando o banco de dados não relacional Realtime Database (capítulo 3.5), que permite a conexão entre planilhas da empresa com a plataforma serverless “Firebase” da Google, que oferece diversos serviços como autenticação, testes A/B, analytics, entre outros.

A metodologia escolhida para a organização estrutural do banco de dados teve como foco facilitar ao máximo a consulta dos dados pela aplicação de maneira a espelhar a organização dos dados de forma similar a cada uma das telas dos modelos criados, recomendado entre desenvolvedores e arquitetos de software que fazem uso de bancos de dados de chave-valor ou documentos (HACKOLADE, 2021).

Deste modo a modelagem resultou em um banco de dados desnormalizados[2], para facilitar as consultas realizadas pela aplicação.

2.5 Metodologia para o desenvolvimento de software - API’s

Figura 4 – Captura de tela do software Visual Studio Code mostrando a estrutura de pastas da arquitetura de software desenvolvida para as API’s.



Fonte: Elaborado pelo autor (2023).

[2] A desnormalização é uma estratégia utilizada para aumentar o desempenho ao simplificar consultas realizadas pela aplicação por meio da normalização dos dados com os padrões de consulta ao invés da normalização dos dados em relação a sua estrutura original provenientes de fontes externas responsáveis por disponibiliza-los no sistema (WIKIPEDIA, 2021).

O fluxograma proposto pela metodologia para o desenvolvimento das API's utilizando o Google Cloud se baseia nas seguintes etapas:

1. **Criação dos requisitos:** para auxiliar a criação da API, inicialmente são estabelecidos os requisitos funcionais da mesma, bem como o ambiente de desenvolvimento utilizado. No caso do projeto realizado, o ambiente de desenvolvimento escolhido foi o Node.js 20 com o framework Express JS.

2. **Criação da estrutura de pastas do projeto:** pastas e arquivos dedicados para o código javascript das funções, nomeando o arquivo principal da função de maneira intuitiva e respeitando os padrões arquiteturais.

3. **Estudo e adição das dependências:** adição dos pacotes das bibliotecas e dependências necessárias para o funcionamento adequado das funções criadas no ambiente javascript do framework Express JS.

4. **Criação da rota para a função atual:** criação de uma rota dedicada (endpoint) para o uso da função nas rotas existentes do framework Express JS.

5. **Upload do código para a nuvem:** enviando a função para o projeto na Google Cloud para os testes preliminares.

6. **Testes unitários:** testes realizados com cada uma das funções criadas visando verificar o funcionamento individual adequado da API desenvolvida.

7. **Gestão dos segredos e chaves:** a adição dos secrets utilizados no projeto atual na plataforma Google Cloud (GOOGLE, 2023b) permitindo assim um armazenamento seguro de dados sensíveis sem a necessidade do uso de variáveis de ambientes ou valores acessíveis a invasores no código desenvolvido para a regra de negócio das funções criadas.

8. **Testes finais:** verificando o funcionamento e acesso aos segredos de forma segura e a validação final do funcionamento da API desenvolvida.

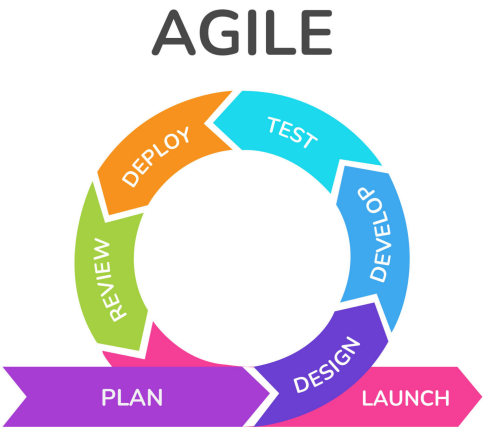
9. **Documentação:** a documentação funções e rotas criadas no projeto permitindo a fácil compreensão do código produzido e das respostas da API.

2.6 Metodologias para gestão de projetos

A execução do projeto respeitou as metodologias definidas para cada setor, incluindo definição dos requisitos, diagramação inicial, arquiteturas e organização, documentação, desenvolvimento, testes e melhorias constantes ao longo de novas iterações.

O objetivo central do setor de gestão de projetos é o controle de qualidade, implementação e integração contínuas de maneira ágil. Para tal, o estudo e adoção dos seguintes conceitos foram fundamentais:

Figura 5 – Imagem que apresenta o fluxograma de etapas comuns em metodologias ágeis.



Fonte: https://www.krasamo.com/agile-development-process.

1. **Estudo de metodologias ágeis:** estudo de práticas relacionadas a metodologias ágeis como Scrum, Kanban, Extreme Programming e Agile.

2. **Feedback constante:** o alinhamento constante e melhorias nas metodologias e requerimentos com as partes interessadas e colaboradores do projeto.

3. **Controle de versão:** criação de repositórios e controle de versão para cada uma das frentes do desenvolvimento do projeto.

4. **Testes unitários e de integração:** realizados com o intuito de validar o código desenvolvido, acelerar o tempo de desenvolvimento validando a integração de diferentes elementos que compõe cada uma das frentes de projeto.

5. **Refatoração:** a refatoração dos códigos produzidos busca manter o alinhamento com os padrões de projeto e arquiteturas de software permitindo maior clareza, organização e documentação.

6. **Controle de qualidade:** a participação de mais de 27 usuários na etapa de testes internos da aplicação Navalcare Connect, com diversos formatos de tela, modelos e versões de aparelhos iOS e Android permite avaliar o funcionamento correto e desempenho da aplicação em situações reais, facilitando a garantia dos padrões de qualidade e exigências das partes interessadas.

7. **Controle das tarefas:** a criação de listas de tarefa contendo “To Do”, “Doing”, “Done” para cada “Sprint” de cada uma das frentes do projeto, visando um controle rigoroso de cada uma das sugestões e feedbacks relevantes para o projeto.

Figura 6 – Logos do evento “The Ocean Race”.

Fonte: https://www.underconsideration.com.

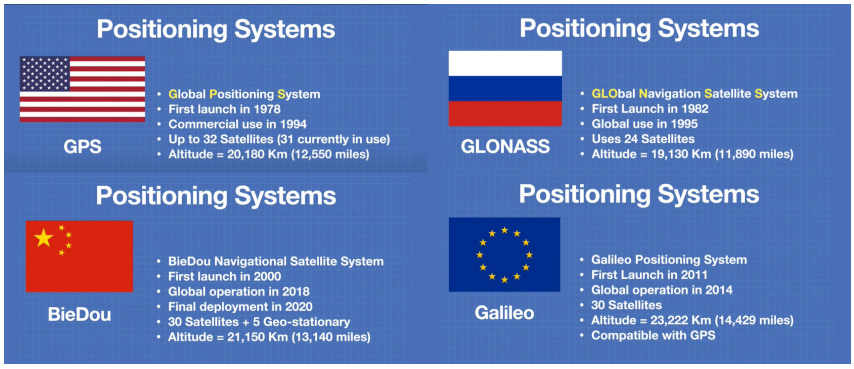
8. **Apresentação dos projetos em grandes eventos:** durante o período de desenvolvimento do projeto, a empresa participou de dois eventos internacionais do setor náutico, The Ocean Race (RACE, 2023) e 63ª Edição do Salão Náutico de Gênova (GENOVA, 2023). Tal participação contribuiu para entender a fundo o mercado atual do setor além de atrair atenção de possíveis clientes e investidores interessados no projeto.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar aos leitores um pouco dos conceitos, plataformas, tecnologias e serviços que foram utilizados para o desenvolvimento do projeto, bem como uma visão geral de seus pontos fortes e fracos.

3.1 Sistema global de navegação por satélite - GNSS

Figura 7 – Sistemas atuais e constelações GNSS.



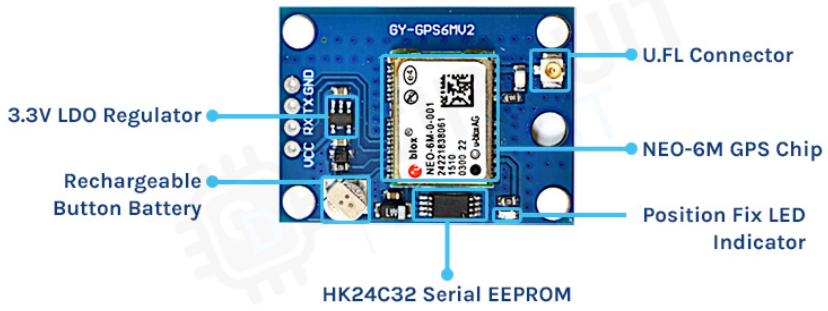
Fonte: https://www.youtube.com/watch?v=kwk3qzaIcCU.

O Sistema Global de Navegação por Satélite (Global Navigation Satellite System - GNSS) constitui-se como um agrupamento de satélites que proporciona sinais a partir do espaço exterior, permitindo a determinação do posicionamento e a cronometragem com precisão em escala global. A aplicabilidade mais conhecida dos sistemas GNSS é a navegação terrestre, marítima e aérea, fundamental para diversas atividades no contexto atual da sociedade.

Diversos sistemas GNSS estão presentes no mercado hoje, sendo o Global Positioning System (GPS), gerido pelos Estados Unidos da América, o GLONASS, desenvolvido pela Federação Russa, o Galileo, da União Europeia, e o BeiDou, administrado pela China, permeando assim uma infraestrutura de alcance mundial e de caráter estratégico para diversos países (WIKIPEDIA, 2023).

A comunicação entre os módulos GPS e plataformas de hardware como ESP32 e Arduino é tipicamente realizada através de uma interface serial, com o uso de protocolos como NMEA (National Marine Electronics Association) ou UBX da empresa u-blox. Este protocolo prescreve um padrão para os dados de GPS que são transmitidos pelos satélites e interpretados pelos módulos receptores.

Figura 8 – Visão geral de uma placa de desenvolvimento GNSS comum que utiliza o módulo NEO-6M da empresa U-blox.

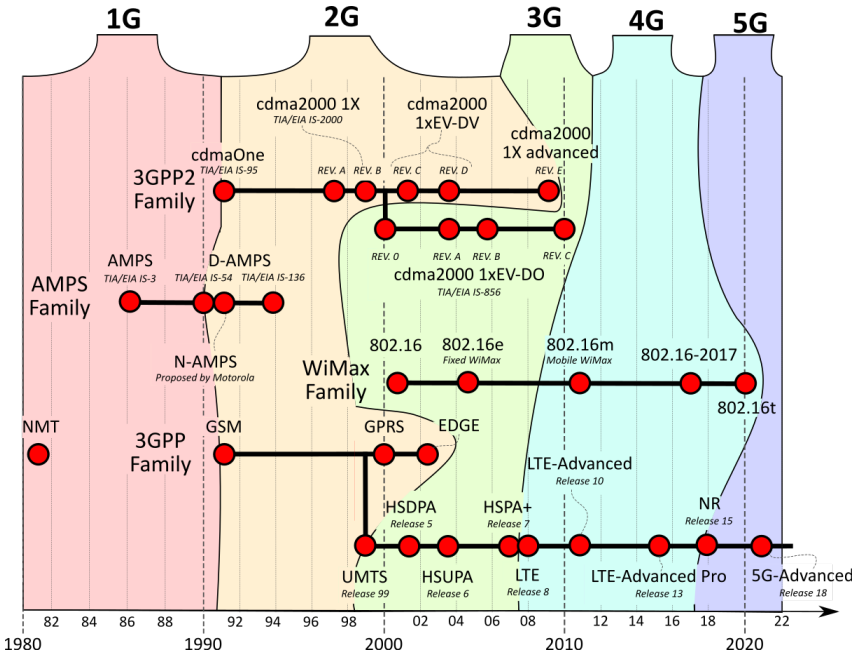


Fonte: https://circuitdigest.com/microcontroller-projects/interfacing-neo6m-gps-module-with-esp32.

3.2 Serviço de rádio de pacote geral - GPRS

O General Packet Radio Service (GPRS), também conhecido comercialmente como 2.5G (TARGET, 2023) é um padrão de tecnologia móvel que se situa na base da infraestrutura da telefonia móvel de segunda geração (2G), é possível observar a distribuição das diversas tecnologias e gerações em uma linha do tempo na figura 9. Este serviço introduziu a transmissão de dados por pacotes, representando uma significativa evolução em relação às redes anteriores, que se baseavam predominantemente na comutação de circuito, utilizada para transmissões de voz.

Figura 9 – Linha do tempo das tecnologias e gerações das redes de telefonia móvel.



Fonte: https://en.wikipedia.org/wiki/2G.

Os módulos GPRS disponíveis comercialmente variam em recursos e capacidades, e sua escolha depende do escopo e das exigências do projeto. Alguns dos mais comuns disponíveis no mercado incluem SIM900 e SIM800 da fabricante SIMCom, que oferecem funcionalidades como envio e recebimento de SMS, chamadas de voz e transferência de dados via GPRS (utilizado no projeto para o envio das informações do hardware para o banco de dados Realtime Database - capítulo 3.5).

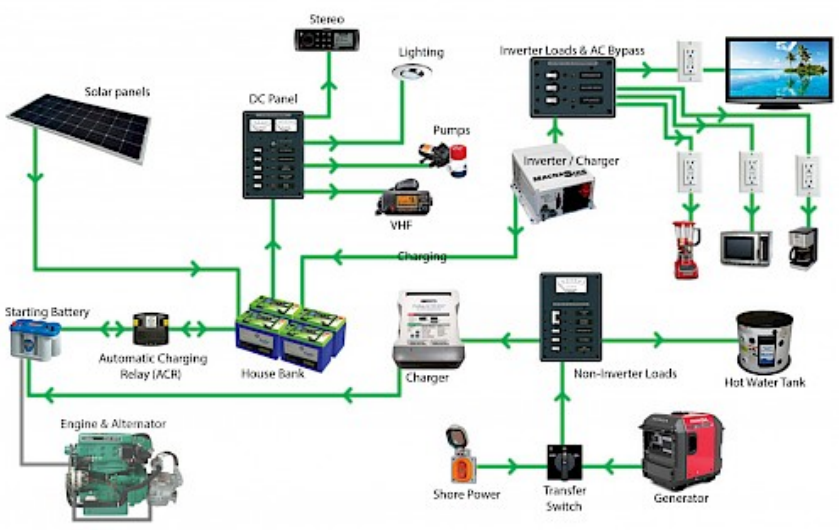
Quanto à eventualidade do desligamento das redes 2G, é indiscutível que a longo prazo os módulos devem estar preparados para funcionar com infraestruturas e tecnologias mais novas, garantindo assim a continuidade da operação dos sistemas que dependem da transmissão de dados em áreas onde as redes 2G podem não mais estar disponíveis. Deste modo, foram adquiridas novas versões de placas de desenvolvimento e módulos do modelo SIMA7670SA visando a futura adaptação do protótipo inicial para as redes 4G (SIMCOM, 2017).

3.3 Sistemas elétricos da embarcação

As embarcações, independentemente do seu propósito ou porte, possuem um componente essencial que garante o seu funcionamento operacional e a segurança de seus ocupantes: o sistema elétrico. Este sistema é responsável por fornecer energia a uma série de dispositivos e sistemas auxiliares.

O projeto do sistema elétrico deve considerar variáveis e requisitos diferentes para cada embarcação, um exemplo de sistema elétrico comum nas embarcações dos clientes atuais da empresa Navalcare pode ser visto na figura 10.

Figura 10 – Exemplo de sistema elétrico comum em yachts.



Fonte: https://www.pysystems.ca/site/assets/files/2407/py_nov18_64-65_tech_talk_2.pdf.

Os sistemas específicos comuns encontrados a bordo incluem, entre outros, os dedicados a sonorização, motorização, iluminação, bem como equipamentos de navegação e comunicação, que podem ser integrados via redes NMEA. Adicionalmente, sistemas de segurança e dispositivos de emergência são essenciais e devem ser eficientemente planejados para assegurar a integridade da embarcação e a segurança dos passageiros em caso de falhas ou condições adversas.

A infraestrutura desses sistemas normalmente é composta por múltiplos bancos de baterias, segmentados por suas respectivas funções. Esta segmentação tem por objetivo fornecer redundância e confiabilidade ao sistema como um todo. A tensão em corrente contínua (DC) existente nas embarcações tende a aderir aos padrões de 12 ou 24 volts, mas essa escolha pode variar conforme requisitos específicos estabelecidos pelos fabricantes e características individuais de cada embarcação.

Um componente crítico no projeto de sistemas elétricos náuticos é o sistema de aterramento. Essencial para a proteção contra choques elétricos, o aterramento adequado é responsável por fornecer um caminho seguro para a dissipação de correntes elétricas, evitando assim danos aos componentes eletrônicos e reduzindo riscos de incêndio.

O aterramento eficaz nos sistemas marítimos também envolve estratégias para eliminar possíveis correntes galvânicas, que podem resultar em corrosão acelerada de componentes metálicos submersos. Consequentemente, não somente o aterramento protege os sistemas elétricos, mas também contribui para a manutenção estrutural a longo prazo da embarcação.

Para o projeto desenvolvido, a compreensão profunda da organização e sub sistemas que compõe o sistema elétrico da embarcação como um todo foi fundamental, tanto para especificação de requisitos quanto para avaliar as diversas abordagens possíveis quanto a instalação, manutenção e acesso aos componentes da embarcação que estão relacionados ao projeto.

3.3.1 Componentes principais do sistema elétrico

Apesar da diversidade nas configurações dos sistemas elétricos em embarcações, há uma uniformidade quanto aos componentes primordiais. Estes incluem:

1. **Baterias:** armazenam a energia necessária para a operação dos diversos equipamentos elétricos e são o cerne do sistema de energia embarcado.

2. **Inversores:** converter a corrente contínua armazenada nas baterias em corrente alternada, alimentando equipamentos que assim o exigem.

3. **Carregadores de Baterias:** asseguram que as baterias sejam mantidas em um estado de carga adequado, utilizando fontes externas de energia ou geradores quando disponíveis.

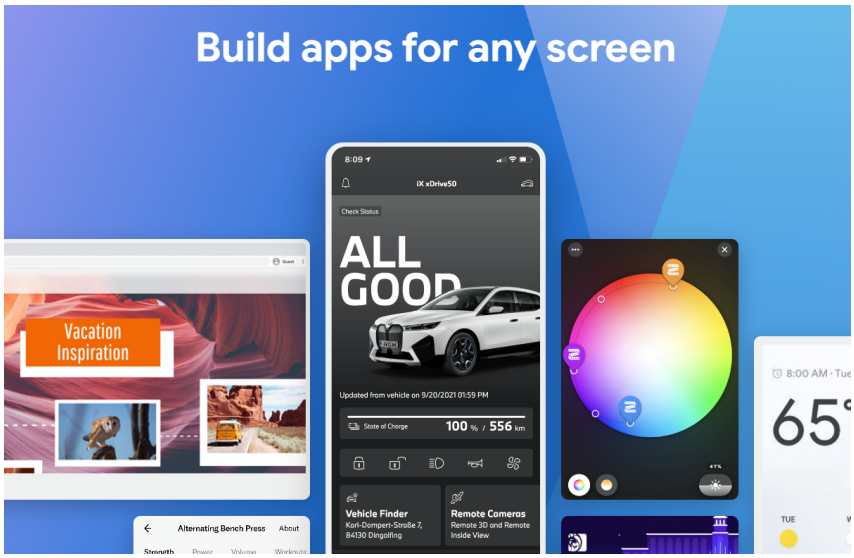
4. **Geradores:** oferecem uma fonte de energia alternativa e são essenciais para embarcações que realizam viagens prolongadas ou possuem grande demanda energética.

5. **Sistemas de Proteção e Distribuição:** incluindo disjuntores, fusíveis e relés, estes componentes são fundamentais para a segurança do sistema e para a prevenção de falhas ou danos decorrentes de sobrecargas ou curto-circuitos.

6. **Rede NMEA:** os protocolos de comunicação NMEA desempenham um papel crucial na integração e no compartilhamento de informações entre dispositivos eletrônicos marítimos, tais como GPS, sonares, radar, piloto automático, e outros instrumentos de navegação. Em particular, os protocolos NMEA0183 e NMEA2000 são amplamente adotados pela indústria marítima, possibilitando a troca de informações de forma confiável e em tempo real entre os vários sistemas a bordo.

3.4 Google Flutter

Figura 11 – Captura de tela da home page do framework Flutter da Google.



Fonte: https://flutter.dev.

O desenvolvimento de interfaces de usuário (UI - User Interface) vem evoluindo significativamente com o surgimento de novas ferramentas e frameworks que ampliam as possibilidades dos desenvolvedores. Neste contexto, o Google Flutter desponta como um framework de UI open-source que figura entre as escolhas mais inovadoras e eficientes da atualidade.

3.4.1 Características do Flutter

Criado e mantido pela Google, o Flutter é um framework de UI que permite o desenvolvimento de aplicações ricas visualmente e altamente interativas. Com uma abordagem moderna e eficaz, destaca-se por diversas características:

1. **Baseado em Dart:** a escolha da linguagem Dart, também desenvolvida pela Google, para o funcionamento do Flutter, não foi aleatória. Dart é projetada para compilação JIT (Just-In-Time) e AOT (Ahead-Of-Time), o que favorece tanto o desenvolvimento rápido (via hot reload) quanto a performance robusta de apps em produção.

2. **Abordagem "everything is a widget":** o conceito-chave do Flutter é que "tudo é um widget". Essa filosofia robusta simplifica a construção da UI, facilitando a criação de interfaces complexas a partir de blocos básicos de construção que englobam aspectos de layout e interatividade.

3. **Engine própria:** o Flutter possui uma engine de renderização inicialmente baseada no Skia e atualmente em processo de migração para o Impeller (nova engine), que é independente de plataformas. Isto permite que a aparência dos apps seja consistente em diferentes sistemas operacionais.

4. **Hot reload:** um dos recursos mais elogiados por desenvolvedores, o hot reload, permite a implementação quase instantânea de alterações no código, propiciando visualizar as mudanças de maneira ágil durante a fase de desenvolvimento.

5. **Vantagens competitivas como:** código aberto, desenvolvimento rápido, widgets personalizáveis, excelente desempenho, compatibilidades com múltiplas plataformas, suporte a diversos tipos e formatos de telas, comunidade ativa, ferramentas de desenvolvimento, cuppertino widgets, bibliotecas ricas, segurança, facilidade na realização de testes, integração nativa, fácil manutenção, documentação abundante, crescente adoção e suporte empresarial.

3.4.2 Adoção do Flutter por grandes empresas

O reconhecimento da capacidade e do potencial do Flutter se concretiza pelo seu acolhimento por grandes nomes no mercado, incluindo Alibaba, Google Earth, Nubank, Ifood, My BMW e eBay. O amplo espectro de aplicações, desde plataformas de comércio eletrônico até soluções financeiras, atesta a versatilidade e robustez do framework.

Figura 12 – Compilado das logos de grandes empresas que usam Flutter para o desenvolvimento de aplicações.

Fonte: Elaborado pelo autor (2023).

3.4.3 Benefícios do Flutter para as empresas

O Flutter oferece uma gama de vantagens estratégicas para as empresas como:

1. **Economia:** a capacidade de manter um código-fonte unificado para diversas plataformas garante uma redução significativa nos custos de desenvolvimento e manutenção da aplicação.

2. **Desenvolvimento acelerado:** a eficiência no desenvolvimento é potencializada pelas características como hot reload e a bem projetada biblioteca de widgets.

3. **Performance comparável ao nativo:** a experiência do usuário não é comprometida, pois a performance do Flutter é similar à de aplicativos nativos.

4. **Flexibilidade e personalização:** adapta-se facilmente à identidade visual de uma empresa, permitindo a criação de aplicações que respeitam os padrões de design estabelecidos.

3.5 Google Firebase

O Firebase é uma plataforma de Backend-as-a-Service (BaaS) que oferece uma gama abrangente de ferramentas e serviços para o desenvolvimento de aplicativos, destacando-se pela facilidade de uso e pela redução de tempo de desenvolvimento. A infraestrutura robusta do Google Cloud assegura a escalabilidade necessária para acompanhar o crescimento da aplicação, com um modelo de precificação que começa com um plano gratuito e se mantém competitivo a outras tecnologias mesmo com um grande número de usuários (GOOGLE, 2023f), alguns dos serviços oferecidos e utilizados pelo projeto foram:

- **Firebase Authentication:**

Figura 13 – Logo oficial da plataforma Firebase da Google.

Fonte: https://firebase.google.com.

  - Serviço de autenticação completo que suporta login através de múltiplos provedores e integração direta com o flutter por meio de um pacote dedicado.
  - Vantagens incluem a facilidade de implementação, especialmente com Flutter, e segurança reforçada.
  - Oferece um plano gratuito para 50.000 usuários mensais ativos, com escalonamento de preços conforme a demanda.

- **Firebase Realtime Database:**
  - Banco de dados não relacional que proporciona sincronização em tempo real entre clientes e integração direta com o flutter por meio de um pacote dedicado.
  - Possui vantagens como a persistência offline dos dados e escalabilidade automática caso haja até 200.000 usuários conectados ao mesmo tempo, além de escalabilidade manual com adição de múltiplos bancos de dados no projeto para comportar mais de 200.000 usuários.
  - Inclui um plano gratuito com 1 GB de armazenamento, 10 GB de download e até 100 instâncias simultâneas, com custos adicionais baseados no uso.

- **Firebase Cloud Messaging:**
  - Sistema para notificações push em plataformas Android, iOS e web que possui integração direta com o flutter por meio de um pacote dedicado.
  - Possui vantagens como configuração simplificada e compatibilidade com Flutter, pode ser utilizado com o Apple Push Notification Service (APNS) da Apple, e não necessita configurações adicionais para o funcionamento com dispositivos Android.
  - Serviço gratuito, sem custos adicionais.

3.6 Google Cloud Functions

Figura 14 – Ilustração representando alguns dos serviços da plataforma Google Cloud.

Fonte: https://www.ituniverse.com.br/noticias/quais-sao-as-vantagens-do-google-cloud-platform/.

O Google Cloud Functions é um serviço de computação sem servidor (BaaS) que oferece uma experiência de desenvolvimento simplificada e rápida. Através de sua arquitetura, permite que os desenvolvedores escrevam e executem trechos de código em resposta a eventos, sem a preocupação com a gestão de infraestrutura. Essa abordagem é ideal para aplicações que necessitam de automação em resposta a eventos e processamento de dados em tempo real, além de possuir outras vantagens como adoção pela comunidade, suporte as linguagens de programação mais utilizadas no mercado, assim como suporte ao uso de bibliotecas e frameworks baseados nas linguagens de programação suportadas pela plataforma.

- **Funcionalidades e Benefícios:**
  - **Desenvolvimento Acelerado:** Permite o desenvolvimento rápido de aplicações, simplificando a orquestração de serviços conectados e a transição do conceito à produção.
  - **Resposta a Eventos:** Executa código em resposta a eventos de uma variedade de serviços do Google Cloud, oferecendo uma ampla gama de possibilidades de automação e interconexão.
  - **Sem Gerenciamento de Servidores:** Abstrai toda a infraestrutura, permitindo que os desenvolvedores se concentrem no código e na lógica de negócios.
- **Precificação:**
  - **Modelo Pay-as-you-go:** Cobrança baseada no tempo de execução da função, com medição até o valor mais próximo de 100 milissegundos, e somente durante o período de execução.

  - **Nível Gratuito Permanente:** Inclui 2 milhões de invocações por mês, 400.000 GB-segundos, 200.000 GHz-segundos de tempo de computação e 5 GB de tráfego de saída de Internet.

- **Framework para Node.js:**

  - **Open-Source e Flexível:** Permite criar funções leves e portáteis para o Cloud Functions em Node.js, oferecendo desenvolvimento local e escalabilidade global.
  - **Capacidade de Diagnóstico Integrada:** Facilita o monitoramento e diagnóstico das funções, integrando-se com o Cloud Trace para observação total do aplicativo.

3.7 Apple WeatherKit REST API:

O Apple WeatherKit, conforme analisado nas fontes oficiais da Apple, emerge como um serviço sofisticado para a incorporação de informações meteorológicas em aplicativos e serviços digitais. Este serviço é notável pela sua abrangência de dados climáticos e pela flexibilidade de integração em diferentes plataformas, tanto da Apple quanto externas, através de APIs específicas.

- **Integração e Dados Meteorológicos:**

  - **APIs para Múltiplas Plataformas:** O WeatherKit disponibiliza uma API Swift para dispositivos da Apple, como iOS e macOS, e uma API REST para outras plataformas. Esta dualidade facilita a integração de informações climáticas em um amplo espectro de aplicações.
  - **Precisão e Diversidade de Dados:** Baseado no serviço Apple Weather, oferece desde condições climáticas diárias até previsões horárias para os próximos dez dias. Dados de temperatura, precipitação, vento, índice UV, entre outros, são disponibilizados, com a adição de previsões minuto a minuto e alertas de clima severo para certas regiões (ainda não disponível no Brasil).

- **Privacidade e Segurança dos Dados:**

  - **Foco na Privacidade do Usuário:** Em linha com a política de privacidade da Apple, o WeatherKit é projetado para entregar previsões detalhadas, sem comprometer a privacidade do usuário. As informações de localização são utilizadas exclusivamente para previsões climáticas e não são vinculadas a dados pessoais identificáveis.

- **Acesso e Estrutura de Custos:**

  - **Requisitos de Acesso:** A utilização do WeatherKit está atrelada à inscrição no Programa Apple Developer, o qual já seria fundamental para a eventual publicação do aplicativo desenvolvido no projeto.
  - **Precificação Variável:** A plataforma fornece inicialmente 500.000 chamadas de API mensais por assinatura do programa Apple Developper. Existem opções de planos adicionais baseados no volume de chamadas, com preços que variam a partir de US$ 49,99 para 1 milhão de chamadas mensais.

- **Ferramentas e Documentação Disponíveis:**

  - **Suporte de Desenvolvimento:** As API's Swift do WeatherKit exigem o uso do Xcode 14, além das versões mais recentes dos sistemas operacionais da Apple. Para outras plataformas, é disponibilizada a API REST.
  - **Recursos de Apoio ao Desenvolvedor:** Documentação abrangente e tutoriais estão disponíveis, oferecendo suporte ao desenvolvimento em diferentes plataformas.

No contexto da API REST do WeatherKit, a autenticação é um procedimento essencial, realizada por meio de um token de desenvolvedor gerado conforme a especificação JSON Web Token (JWT). Este token, construído com a utilização de bibliotecas como "jsonwebtoken", deve incluir informações como o algoritmo de assinatura ES256 e identificadores específicos obtidos através do Apple Developer Account. A inclusão desse token nas solicitações de API assegura uma camada de segurança e privacidade, em conformidade com as diretrizes da Apple.

Através desta análise, pode-se observar que o Apple WeatherKit representa uma ferramenta valiosa para o desenvolvimento de soluções meteorológicas em uma variedade de plataformas, com forte ênfase na privacidade e na disponibilização de dados detalhados e precisos. Este resumo reflete uma compreensão abrangente dos recursos e capacidades do WeatherKit, conforme descrito nas fontes oficiais da Apple.

3.8 Google App Scripts

O Google App Scripts, conforme descrito na documentação oficial (GOOGLE, 2023a), é uma plataforma de desenvolvimento baseada na nuvem para a criação de aplicações web. Completamente operado através de um navegador, este serviço é conhecido por sua capacidade de integrar-se com facilidade aos produtos Google, adotando o JavaScript como linguagem de programação.

- **Integração com APIs e Ferramentas Google:** A plataforma incorpora API's Google e ferramentas de interface de usuário como serviços embutidos. O editor de

Figura 15 – Ilustração representando alguns recursos do framework Google App Scripts.

Fonte: https://codeburst.io/automating-google-forms-sheets-using-apps-script-2c59db97966f.

scripts facilita o acesso a esses serviços, proporcionando recursos como a conclusão automática de código.

- **Opções de Armazenamento de Dados:** É possível armazenar dados em planilhas Google, conectar-se a bancos de dados externos ou utilizar o ScriptDb, que funciona como um armazenamento de objetos JavaScript.
- **Criação de Projetos:** Para começar um novo projeto no Google App Scripts, pode-se criar um script diretamente no Google Drive ou acessar o site script.google.com.

3.9 Express JS

O Express JS é um framework do Node.js, concebido para a criação rápida e eficiente de aplicações web e API's. Este framework é amplamente reconhecido por sua performance e versatilidade, tornando-o uma escolha popular para desenvolvedores que trabalham com JavaScript.

- **Características Técnicas:**
  - **Redução de Código e Tempo:** Comparado ao uso dos módulos nativos do Node.js, o Express JS diminui significativamente a quantidade de código e tempo necessário para desenvolver componentes de back-end, como API's.
  - **Middleware e Roteamento:** Implementa funções de middleware para processamento de requisições e respostas, e um sistema de roteamento para gerenciar respostas a diferentes solicitações de URL's.
  - **Flexibilidade e Performance:** A natureza não prescritiva do Express JS e sua rapidez são pontos fortes, embora a falta de uma estrutura rígida possa complicar a manutenção do código.

- **Uso no Setor Corporativo:**
  - **Empresas Utilizadoras:** Organizações como IBM, Uber, eBay e PayPal adotam o Express JS, evidenciando sua aplicabilidade e confiabilidade no desenvolvimento de aplicações web (BACK4APP, 2023).

3.10 Padrões de projeto utilizados

Os padrões de projeto podem ser vistos como soluções para problemas corriqueiros no desenvolvimento de software, oferecendo uma base sólida e universalmente aceita pela indústria, testada na prática inúmeras vezes.

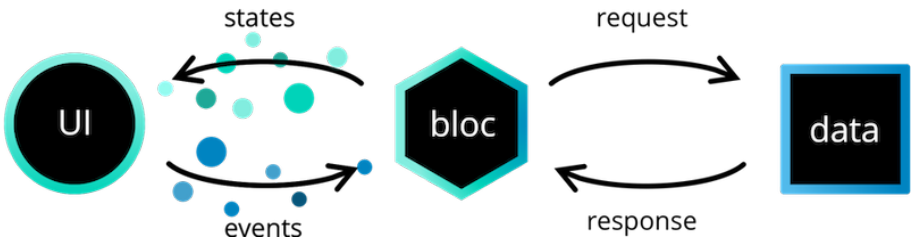
Um dos grandes potenciais dos padrões de projeto e do desenvolvimento de software personalizado é a adaptabilidade e liberdade que as linguagens de programação oferecem atualmente.

Porém, a versatilidade e flexibilidade das linguagens de programação pode ser também uma desvantagem quando o código se torna confuso ou mal redigido. Por isso, fazer uso dos padrões de projeto para a resolução dos problemas e demandas mais variadas é uma prática essencial para a produção de código de qualidade, uma vez que após a habituação com as ferramentas e padrões mais conhecidos, utiliza-los e seleciona-los de forma adequada se torna tão intuitivo quanto o uso de uma linguagem qualquer como a língua portuguesa.

Deste modo, ao longo do desenvolvimento do projeto, alguns dos padrões de projeto utilizados foram: Prototype, Adapter, Singleton, Factory Method, Builder entre outros.

3.11 Business Logic Component - BLoC

Figura 16 – Fluxograma ilustrativo da arquitetura BLoC.



Fonte: https://bloclibrary.dev/.

O acrônimo BLoC (Business Logic Component) pode ser traduzido para componente de regra de negócios, BLoC é tanto um padrão de projeto, pacote quanto uma arquitetura desenvolvida para simplificar o gerenciamento de estados por meio do uso de streams (fluxos de dados) de estados e eventos para estabelecer uma ponte entre as regras e a interface de aplicações.

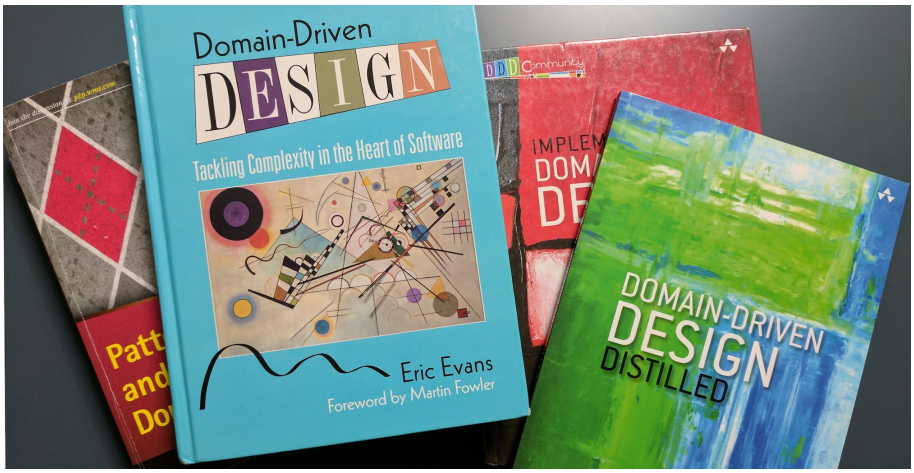
Com o crescimento e popularização do framework Flutter, muitas metodologias para o gerenciamento de estado surgiram, com diversas propostas, implementações, funcionalidades, vantagens e desvantagens.

A adoção do padrão BLoC juntamente do uso do pacote para o gerenciamento de estados da aplicação teve como foco a padronização do código desenvolvido e a agilidade para produção de reatividades, além de benefícios como alto desempenho quando comparado com demais concorrentes e funcionalidades como uso do armazenamento local com alta integração e praticidade, através do pacote "Hydrated BLoC" (DDD, 2023) que foi construído sobre o pacote para implementação de bancos de dados com Dart denominado "Hive", que atualmente é uma das soluções mais poderosas quanto a velocidade de acesso dos dados locais e segurança.

3.12 Domain Driven Design - DDD

Um dos pilares centrais do projeto, que realmente serviu como guia para um entendimento mais profundo dos conceitos e abstrações relacionados ao desenvolvimento de software, modelagem de regras de negócio, modelagem de objetos de dados, entre outras lições de extrema relevância foram as metodologias relacionadas difundidas por Erik Evans, nomeadas "Domain Driven Design" (KOSTYRA, 2023).

Figura 17 – Livros recomendados para o estudo do Domain Driven Design (DDD).

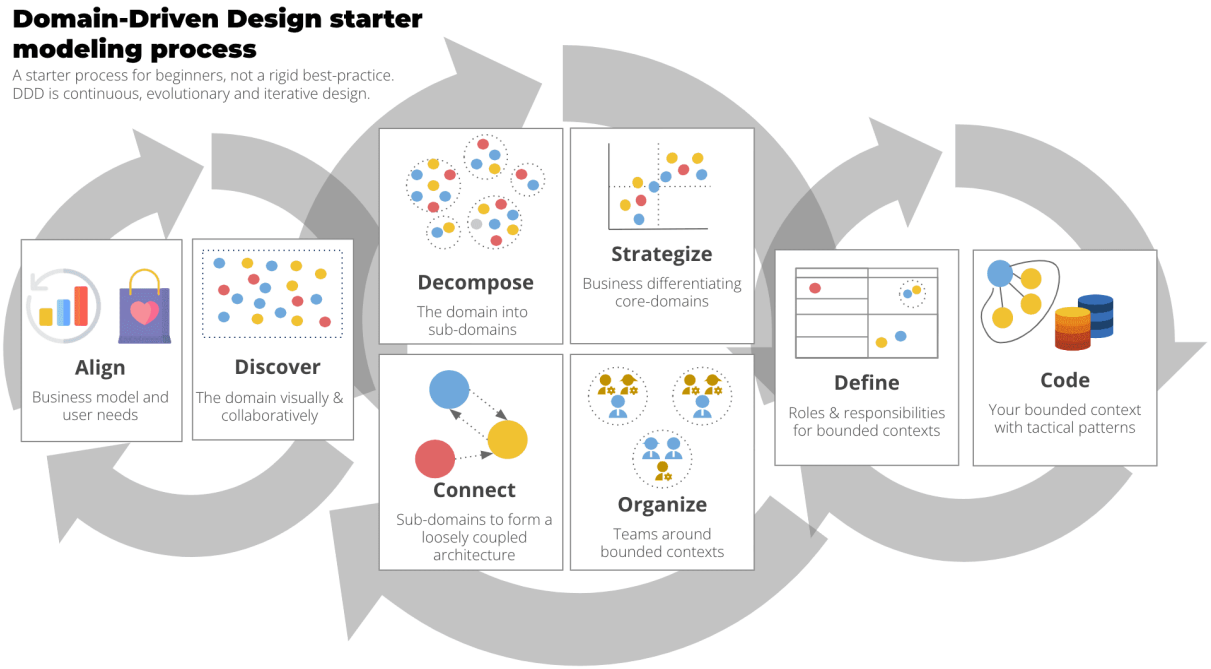


Fonte: https://www.tripled.io/02/07/2017/What-is-DDD/.

Foge do escopo do projeto adentrar a fundo nas metodologias e conceitos envolvidos no uso e entendimento das técnicas do Domain Driven Desgin, porém, de uma forma sucinta podemos citar alguns conceitos indispensáveis para o desenvolvimento do projeto.

- **Padrões Táticos (ou blocos de construção nos termos originais do DDD):**
  - **Entidade:** Semelhante aos objetos no design clássico orientado a objetos, mas com foco na identidade ao longo do tempo.
  - **Value object:** Entidade imutável identificada por seus atributos.
  - **Factory:** Padrão de projeto criacional que serve de interface para a criação de instâncias de uma super classe.
  - **Serviço:** Objeto sem estado que encapsula a lógica de domínio que não pode ser razoavelmente colocada em uma entidade ou objeto de valor.
  - **Agregado:** Encapsula várias entidades e objetos de valor.
  - **Repositório:** Biblioteca para obtenção de acesso aos agregados, tem como objetivo simular o acesso a memória local para as outras partes do código, mesmo quando contém internamente acessos a API.

Figura 18 – Fluxograma das etapas relacionadas a aplicação prática dos conceitos do Domain Driven Design.



Fonte: “How Domain-Driven Design Can Boost Your Product Development”.

- **Padrões e conceitos estratégicos:**
  - **Contexto limitado:** São limites estabelecidos em torno de diversos domínios com o objetivo de expor agrupamentos entre elementos que pertencem a um mesmo contexto do projeto.

- **Mapa de contexto:** É basicamente, um mapeamento dos diversos contextos, agregados, e domínios do sistema sob estudo.
- **Domínio principal:** É a parte da modelagem que envolve o maior valor comercial para as partes envolvidas. Deve ser abordado como o cerne de todo processo de desenvolvimento do sistema proposto.

4 DESENVOLVIMENTO

O projeto foi dividido em dois sistemas principais denominados “Navalcare Connect” e “Navalcare Safeboat”, criados com o objetivo de: desenvolver hardware proprietário para o sensoriamento remoto de embarcações, desenvolver uma aplicação móvel que permita visualizar em tempo real dados provenientes do hardware além de permitir escalabilidade futura e atendimento de outras demandas dos clientes e partes interessadas.

4.1 Criação dos requisitos

Após estudos iniciais, revisão do estado da arte e extensas pesquisas de mercado realizadas, foi possível criar o escopo original do projeto que teve como foco atender as demandas dos clientes e contratantes. Assim, os requisitos funcionais para o projeto foram:

1. Desenvolvimento de uma aplicação para os clientes, denominada Navalcare Connect, com foco original na validação dos protótipos de hardware e demandas constantes provenientes de outras funcionalidades do interesse da empresa.

2. Desenvolvimento de API’s para atender os requisitos funcionais da aplicação quanto a como previsão do tempo personalizada para o setor náutico.

3. Desenvolvimento de um protótipo em hardware para o envio de dados como: estado de sensores de alagamento e incêndio, latitude e longitude atual da embarcação, temperatura e umidade da casa de máquinas, entre outros.

4. Desenvolvimento de metodologias e boas práticas para os diversos setores envolvidos no projeto, com regras para padronização, organização e escalabilidade dos sistemas produzidos e times internos da empresa.

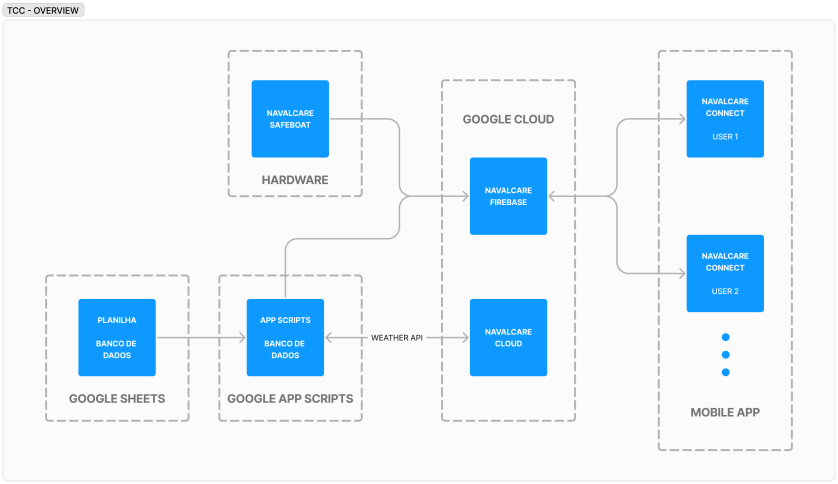
4.2 Visão geral do sistema proposto

Com base nos requisitos funcionais, passou-se a etapa de esboço do diagrama de blocos geral do projeto, após a escolha de tecnologias que seriam utilizadas para implementação de cada sub sistema envolvido no funcionamento adequado da infraestrutura proposta.

A figura 19 explicita o fluxo de dados entre os sistemas concebidos. Partindo de uma planilha do Google, os colaboradores e administradores do sistema podem fazer modificações no banco de dados utilizando a planilha como interface gráfica para visualização das informações e modificações dos dados.

Após o uso da planilha, as modificações feitas são então sincronizadas com o banco de dados Realtime Database da Google, através do uso do framework Google App Scripts,

Figura 19 – Diagrama de blocos do sistema proposto.



Fonte: Elaborado pelo autor (2023).

que é responsável por autorizar e realizar o envio dos dados para o banco de dados da plataforma Firebase.

Graças ao uso do banco de dados não relacional Realtime Database, todas as modificações realizadas podem ser sincronizadas com a aplicação caso seja especificado em código pelo desenvolvedor, com delay de aproximadamente alguns milisegundos (GOOGLE, 2016), permitindo aos usuários uma experiência fluída e segura, mantendo os dados de interesse sempre atualizados.

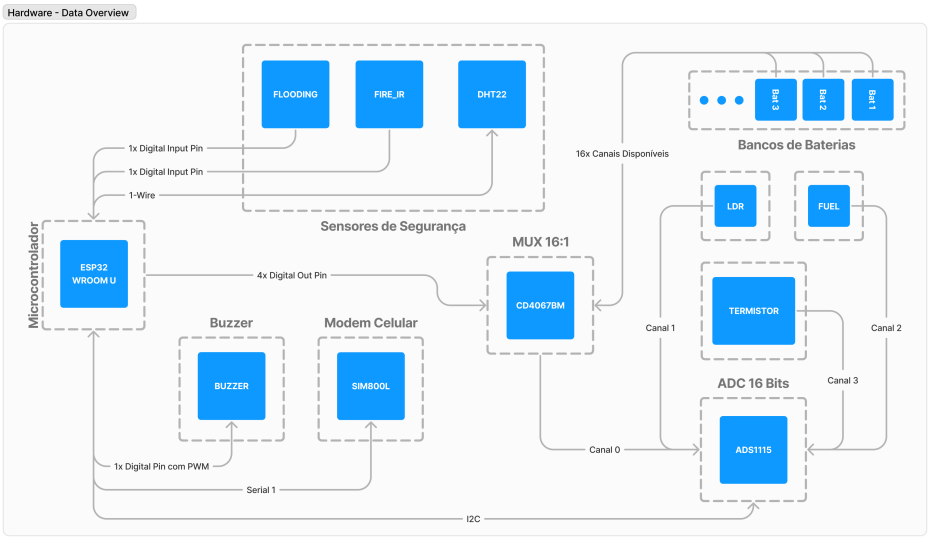
O hardware concebido teve como foco principal estabelecer a conectividade com as redes de telefonia móvel e o envio direto das informações para o banco de dados, validando a proposta original de visualização dos dados através da dashboard desenvolvida na aplicação.

Além de todas as funcionalidades e requisitos originais, foi do interesse das partes envolvidas a criação de outras interfaces para compor a aplicação como telas para autenticação, onboarding, previsão do tempo, lembretes de revisão, minha embarcação, meu perfil, visualização da carteira de habilitação e documentos da embarcação, entre outros, possibilitando o desenvolvimento de uma plataforma que permitisse a empresa moldar uma aplicação proprietária que será eventualmente publicada nas lojas de aplicativos e utilizada por seus clientes.

Deste modo, para atender as demandas e os requisitos envolvidos no projeto “Navalcare Connect”, foram criadas funções hospedadas em ambientes “Node.js 20” na infraestrutura do Google Cloud utilizando o framework Express JS para desenvolvimento de API’s, deste modo surgiu o sistema “Navalcare Cloud” com objetivo primário de servir como ponte para o consumo dos dados necessários para o funcionamento adequado da seção de previsão do tempo náutica do aplicativo “Navalcare Connect”.

4.3 Desenvolvimento do hardware

Figura 20 – Diagrama de blocos mostrando o fluxo de dados do protótipo desenvolvido.



Fonte: Elaborado pelo autor (2023).

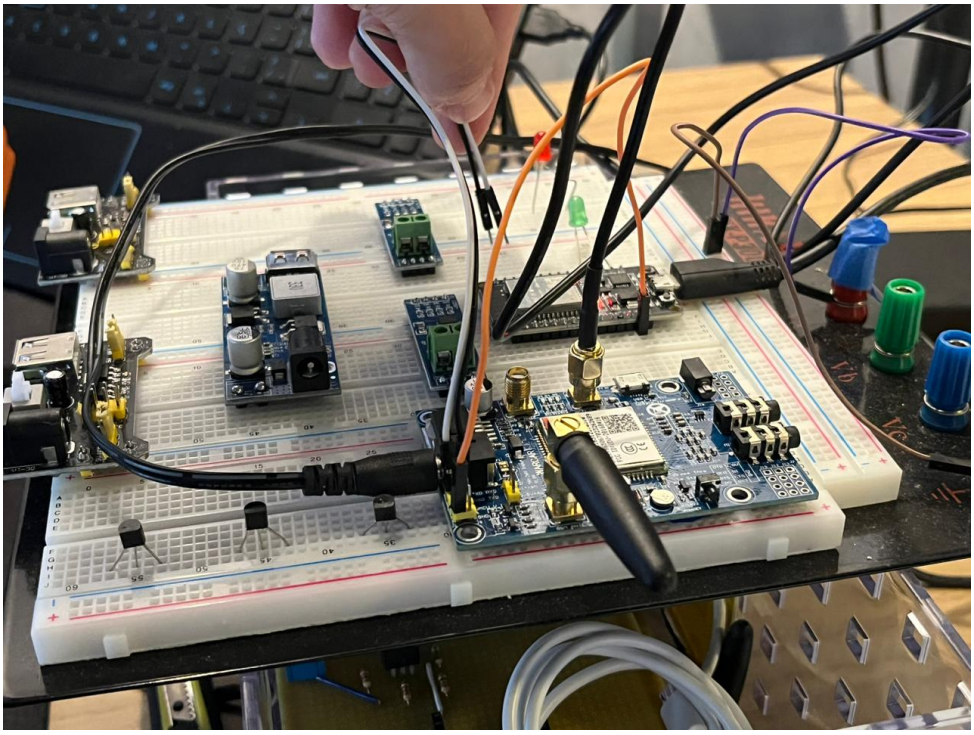
O protótipo do hardware desenvolvido consistiu no uso de diversas tecnologias como GPRS, GNSS, transdutores e sistemas embarcados para realizar a aquisição e envio de dados relevantes para o setor náutico como localização da embarcação, temperatura, sensores de segurança, entre outros, buscando validar o funcionamento e servir como base para criação de uma plataforma IoT com hardware dedicado e proprietário.

A placa de desenvolvimento selecionada para a criação do protótipo do projeto foi o ESP32 WROOM-32 Development Board. A escolha da placa de desenvolvimento baseada no microcontrolador ESP32 teve como foco a conectividade, custo, disponibilidade e simplicidade da mesma, além de fornecer vantagens estratégicas como Bluetooth para eventuais versões futuras e grande adoção pelo mercado, comunidade ativa e incontáveis bibliotecas de código aberto. Deve-se também observar que todas as etapas e metodologias utilizadas tiveram como base respeitar da melhor forma possível as etapas e nuances envolvidas na metodologia estabelecida pela seção 2.1.

4.3.1 Comunicação com as redes de telefonia móvel

Uma das etapas centrais do processo de criação do hardware proposto foi o sensoriamento remoto de embarcações, atendendo a demandas do setor náutico quanto a integração das embarcações com o universo Internet of Things (internet das coisas) (IoT), inicialmente foi proposto utilizar a rede Wi-Fi do Iate Club Santa Catarina, onde se situa a sede da empresa contratante. Porém após averiguar possíveis problemas quanto

Figura 21 – Testes com a placa de desenvolvimento SIM808 evb-v3.2.



Fonte: Elaborado pelo autor (2023).

ao deslocamento das embarcações e outras funcionalidades importantes para a vantagem competitiva do produto final, foi adotado o uso de modems GPRS SIM808 com chips M2M da empresa Arqia para o envio dos dados coletados para a internet.

Além disso, esses dispositivos adicionam a possibilidade de realizar o rastreamento e criação de novas funcionalidades embarcadas na aplicação "Navalcare Connect" como a "Cerca Virtual" que envia notificações aos proprietários da embarcação quando as notificações da aplicação são ativadas e sua embarcação entram ou saem de uma circunferência com centro e raios variáveis definidos pelos usuários na aplicação.

Deste modo, foi realizada a aquisição de placas de desenvolvimento baseadas no circuito integrado SIM808 da fabricante SIMCom. Com as placas em mão, após uma série de testes foi possível estabelecer comunicação entre o microcontrolador utilizado para a prototipagem (ESP32 Wroom-32 Development Board) e a internet.

No inicio, isso foi realizado através de leitura dos dados de uma API PHP de terceiros e o uso de comandos AT pela comunicação serial com o microcontrolador, através do protocolo HTTP, então após ajustes no código, foi possível estabelecer o envio seguro das informações diretamente para o banco de dados Realtime Database, através do protocolo HTTPS.

A carência de boas referências e cursos para capacitação quanto ao entendimento e uso dos módulos de telefonia móvel em geral se mostrou uma grande barreira para novos

desenvolvedores, porém, após a persistência e estudo, além da participação em eventos como Embarcados (2023), foi possível compreender melhor a estrutura dos modems celulares, tecnologias existentes e problemas relacionados ao uso de redes móveis pertencentes a segunda geração (2G) a longo prazo, uma vez que esta possivelmente será substituída por tecnologias modernas como a LTE Cat-4, LTE Cat-1, LTE Cat-M1, NB-IoT, entre outras.

Pensando na escalabilidade do projeto e na demanda futura pela substituição de módulos 2G por módulos mais atuais com suporte a redes 4G, por sugestão do consultor da empresa SIMCom, foi especificado o uso de módulos do modelo SIMA7670SA, que suportam a tecnologia LTE Cat-1, GPRS e GSM. Porém, ainda será necessário averiguar na prática detalhes como fornecedores, custo, compatibilidade com as bibliotecas utilizadas Shymanskyy (2023) entre outros fatores para escolha definitiva do modem celular da versão final futura do projeto.

4.3.2 Aquisição dos dados de temperatura, umidade e sensores de segurança

Figura 22 – Detalhes técnicos do sensor DHT22 utilizado para aferir a temperatura e humidade.

3. Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Fonte: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

A aquisição dos dados de temperatura e umidade do protótipo foi feita através do uso do sensor DHT22 (especificado na figura 22 acima), apenas para a validação inicial do protótipo. Os dados dos sensores de segurança foram coletados através de dois sensores distintos, um sensor de chama infravermelho comum, configurado para o máximo de sensibilidade e um sensor de nível do tipo boia.

Tanto o sensor de nível quanto o sensor de chama infravermelho enviam um nível lógico baixo e um nível lógico alto, respectivamente, quando passam para o estado em que uma possível ameaça a integridade da embarcação é detectada. Deste modo, sua conexão

e uso são extremamente simples, porém testes práticos com o hardware dedicado devem ainda ser realizados para assegurar seu funcionamento adequado em situações reais.

4.3.3 Leitura das tensões das baterias

A leitura das tensões dos bancos de bateria inicialmente seria realizada através das portas analógicas e conversores nativos do módulo ESP32 WROOM-32.

Porém, após testes práticos em bancada com fontes de tensão e multímetro, foi averiguado que a solução não seria adequada para os requisitos atuais do projeto, apresentando leituras imprecisas devido a erros relacionados a linearidade do conversor analógico-digital interno do microcontrolador e também devidos ao divisor resistivo utilizado para o mapeamento da faixa de tensão de até 35 volts para aproximadamente 3.182 volts, utilizando resistores de valor 100k e 10k.

Devido aos problemas observados e a limitação no número de portas com conversão analógica-digital disponíveis, optou-se pelo uso do conversor Analógico-Digital (A/D) dedicado ADS1115, o qual apresentou resultados favoráveis durante a etapa de testes inicial.

O módulo ADS1115 por si só possui 4 canais para leitura diferencial ou pontual das tensões, porém para aumentar ainda mais as entradas disponíveis, atendendo a requisitos das partes interessadas, foi adquirido e adicionado no projeto original do hardware também um multiplexador de 16 canais que se baseia no Circuito Integrado (CI) CD4067BM (INSTRUMENTS, 2023).

Assim como todos os outros componentes adquiridos no projeto do protótipo do hardware, foram analisadas as classificações máximas absolutas de cada componente e avaliada a necessidade do uso de circuitos analógicos adicionais para proteção de fatores externos como, sobre tensão, polaridade reversa, faixa de operação do projeto, entre outros.

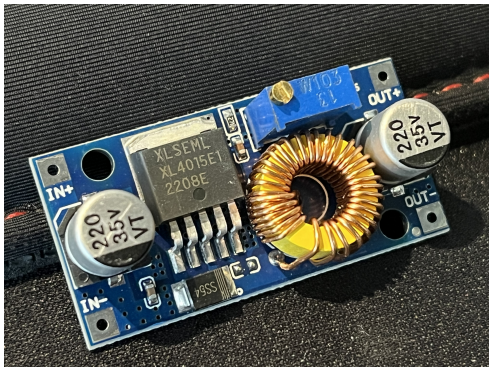
4.3.4 Fontes de alimentação

As fontes de alimentação utilizadas para os testes com o protótipo do hardware desenvolvido foram apenas dois reguladores de tensão do tipo DC-DC com capacidade para até 3A de corrente contínua (que pode ser visto na figura 23). Ambos reguladores foram alimentados por uma fonte 12V de 5A conectada na rede elétrica, sendo que um dos reguladores foi configurado para uma tensão de saída de 4V e outro para uma tensão de saída de aproximadamente 2V, simulando um valor de tensão lido pelo conversor A/D após atenuação do divisor resistivo.

A alimentação da placa de desenvolvimento ESP32 WROOM-32 foi realizada a partir do próprio cabo USB utilizado para o envio do firmware para o protótipo.

Porém, na versão final em hardware dedicado, é necessário fazer uso da tensão de um dos banco de baterias da embarcação para a alimentação do hardware, configurando

Figura 23 – Conversor DC-DC utilizado para alimentação dos módulos do protótipo.



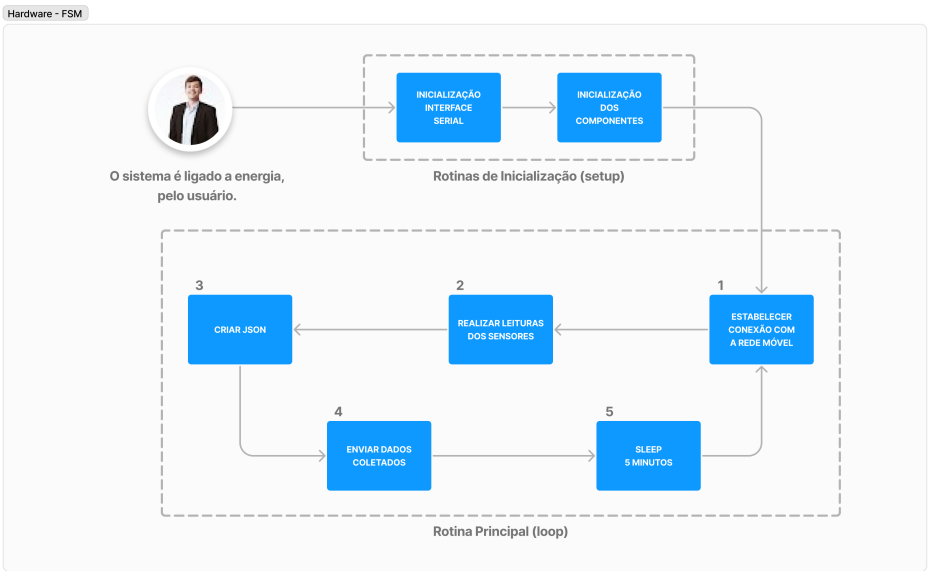
Fonte: Elaborado pelo autor (2023).

um dos conversores DC-DC para a tensão de 5 volts adequada para alimentar o regulador de tensão dedicado do microcontrolador.

4.4 Desenvolvimento do firmware

Para o desenvolvimento do firmware do projeto "Navalcare Safeboat", foram definidos requisitos funcionais simples para a primeira versão, buscando apenas validar na prática o envio dos dados o uso e integração de cada um dos sistemas que compõe o projeto de hardware.

Figura 24 – Máquina de estados finitos (FSM) do firmware do protótipo.



Fonte: Elaborado pelo autor (2023).

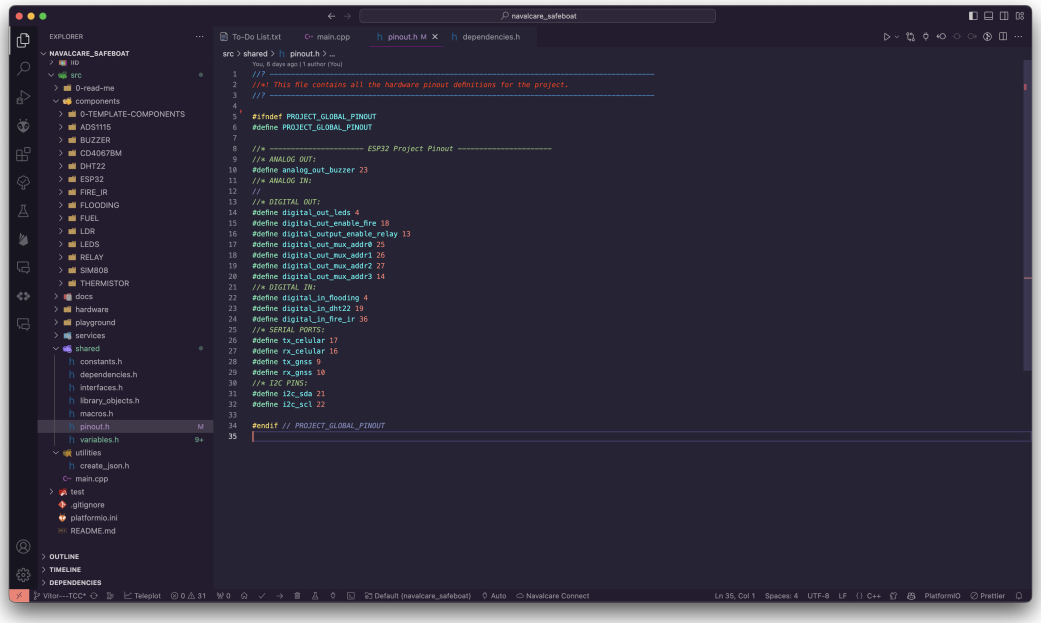
Com isso em mente, foi realizada a criação de uma máquina de estados finitos, do

tipo “Máquina de Mealy”, para explicitar cada um dos estados envolvidos no processo de funcionamento do hardware proposto. A Finite State Machine (FSM) desenvolvida pode ser vista na figura 24. Vale a pena ressaltar que as etapas ilustradas pela figura 24 contém apenas as macro etapas, abstraindo o funcionamento, chamada de sub rotinas e pormenores envolvidos na execução do código fonte do hardware “Navalcare Safeboat”.

Também foi isento na figura 24 o tratamento de erros de cada uma das funções criadas, um exemplo é a função “read_temperature” que realiza a leitura da temperatura utilizando o sensor DHT22, caso o sensor esteja desconectado ou com apresente outros problemas, os valores lidos através do uso das funções da biblioteca podem resultar em variáveis com valores NULL (nulos). Deste modo, no apêndice A podemos ver o tratamento de erros em ação exemplificando como podemos prevenir valores nulos no firmware desenvolvido para cada funcionalidade.

4.4.1 Mapeamento de pinos utilizados no projeto

Figura 25 – Captura de tela do software Visual Studio Code mostrando o mapeamento de pinos implementado no código fonte do firmware.



Fonte: Elaborado pelo autor (2023).

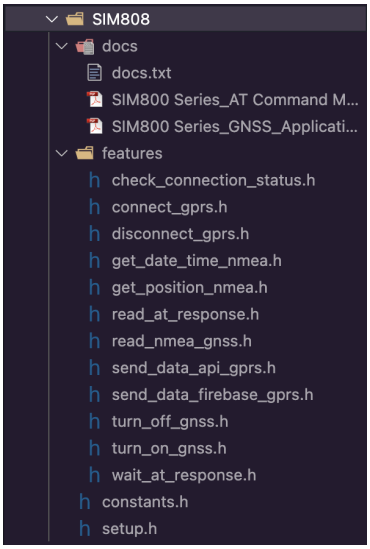
O mapeamento final dos pinos utilizados pelo projeto do sistema embarcado desenvolvido é uma etapa fundamental para a produção de um código seguro, organizado e escalável. O mapeamento dos pinos utilizados pelo firmware, figura 25, foi feito inicialmente a partir dos testes com o protótipo em bancada, então utilizou-se os esquemáticos

desenvolvidos para o hardware dedicado, visando testar o funcionamento adequado de cada componente e das escolhas para a pinagem do projeto final, desde a etapa de prototipagem.

4.4.2 Desenvolvimento das funcionalidades de cada componente

O desenvolvimento do firmware para o sistema embarcado do projeto "Navalcare Safeboat" foi realizado em um primeiro momento através da definição das metodologias e arquiteturas envolvidas no projeto, descritas na seção 2.2.

Figura 26 – Captura de tela do software Visual Studio Code mostrando a divisão de pastas e arquivos para um dos componentes utilizados no projeto do protótipo inicial do sistema embarcado "Navalcare Safeboat".



Fonte: Elaborado pelo autor (2023).

Após analisar os requisitos e elementos envolvidos no projeto, o código necessário para efetivar o teste unitário de cada componente foi então desenvolvido de maneira modular dentro da pasta "components" da arquitetura de software desenvolvida para o firmware. Na figura 26 podemos ver, por exemplo, os arquivos utilizados para implementar todo o funcionamento adequado do componente SIM808.

Pode-se constatar assim que, o código de cada componente consiste em um arquivo denominado "setup.h" responsável por centralizar a inclusão de cada uma das features desenvolvidas para o componente em questão, além de conter o método "setup_SIM808()" que é responsável pela execução das rotinas e configurações iniciais que fazem parte do processo de inicialização correto do componente "SIM808".

Também é possível observar que dentro da pasta de cada um dos componentes do projeto, foi criada uma sub pasta denominada "docs" que é responsável por centralizar informações relevantes ao desenvolvimento correto do firmware do componente, bem como links e outras informações úteis de maneira geral para o projeto do firmware e hardware.

4.4.3 Desenvolvimento da rotina de inicialização principal

Após a conclusão da etapa de mapeamento dos pinos utilizados pelo sistema embarcado e da criação do código fonte individual de cada um dos componentes utilizados no projeto, foi então criada a rotina de inicialização principal do projeto, seguindo a máquina de estados finitos proposta 24 e as boas práticas adotadas.

A rotina "Setup" é responsável por inicializar as interfaces seriais e executar as rotinas individuais de cada um dos componentes, que são por sua vez responsáveis pelo mapeamento dos pinos utilizados pelo componente em questão além da execução de comandos necessários para sua inicialização adequada.

Tanto no início quanto no final da rotina de inicialização do projeto são emitidos avisos sonoros utilizando o componente "Buzzer", também são enviados avisos alertando o início e término da configuração individual dos componentes e da rotina de inicialização principal através da interface serial USB para o monitor serial. A função completa para a inicialização se encontra disponível para consulta no apêndice B.

4.4.4 Desenvolvimento da rotina para implementação da FSM.

Por fim, foi então desenvolvido o código para a implentação da máquina de estados finitos criada para o projeto do firmware, fazendo o uso de diversas sub rotinas que implementam as funcionalidades dos componentes utilizados no projeto. Podemos ver o código fonte utilizado para a validação de um protótipo inicial no Apêndice C.

A pasta "services" presente na arquitetura desenvolvida para o firmware, tem como propósito armazenar cada uma das rotinas que deverão possuir nomenclaturas similares a nomenclatura dada a cada um dos estados da FSM, seguindo o padrão "1_nome_do_primeiro_estado.h".

4.5 Desenvolvimento do frontend da aplicação

O frontend desenvolvido para a aplicação "Navalcare Connect" foi produzido com base nos padrões arquiteturais e metodologias previamente estabelecidas no capítulo 2.3, tendo como objetivo central o uso de metodologias ágeis, padrões de projeto e boas práticas para a criação de uma base de código escalável e organizado, além de permitir entregas semanais de diversas telas da aplicação e validação com o time interno da empresa, testes e correção de eventuais problemas.

Deste modo, nas próximas seções, serão abordadas as etapas envolvidas no desenvolvimento do projeto da aplicação móvel, bem como os testes e versões iniciais e o uso do framewok Flutter.

4.5.1 Criação do layout final para as telas da aplicação

O layout das telas da aplicação inicialmente foi realizado pelo autor, com o uso de ferramentas como photoshop e rascunhos em papel, após a implementação inicial de uma versão alfa da aplicação, foi possível se ambientar melhor com as ferramentas do framework utilizado e foi constatada a barreira técnica e temporal envolvida na criação dos mockups (referências) para as telas da aplicação.

Com foco em facilitar o processo de desenvolvimento e mudar o tom do projeto, foi contratado pelas partes interessadas um designer que passou então a fazer parte do projeto da aplicação móvel "Navalcare Connect". Assim, foi possível acelerar muito o desenvolvimento das telas e também criar inúmeros layouts de novas funcionalidades que foram sendo desenvolvidas durante as reuniões semanais com o designer contratado e os membros do projeto.

Deste modo, atendendo aos pedidos internos provenientes dos contratantes, boa parte do foco do projeto foi direcionado na reprodução fiel das diversas telas da aplicação "Navalcare Connect", bem como a correção de problemas, testes com as diversas plataformas e telas dos usuários participantes da versão beta de testes internos, visando a publicação final do aplicativo desenvolvido nas lojas de aplicativos da Apple e Google.

4.5.2 Desenvolvimento das telas da aplicação a partir dos mockups

Após as reuniões semanais, foram desenvolvidas as telas iniciais de autenticação, onboarding, home page, safeboat entre outras. As referências para a criação do código da aplicação foram compartilhadas na plataforma Figma, com o intúito de facilitar o acesso e acompanhamento em tempo real compartilhado entre os membros do projeto, bem como permitir a visualização inicial de um sistema de navegação emulando a navegação real da aplicação ao clicar nos botões interativos das telas desenvolvidas. Deste modo, a criação das telas a partir dos mockups seguiu fielmente as etapas descritas na seção 2.3. Com isso, foi possível adquirir uma estrutura sólida, escalável e acessível para o código da aplicação.

Para o funcionamento correto da aplicação em diversos formatos de tela e sistemas operacionais, foi utilizada uma metodologia desenvolvida pelo autor, onde é possível inserir elementos visuais com tamanhos proporcionalmente idênticos aos do design, posiciona-los e também adapta-los de maneira automática para qualquer tamanho e proporção de tela.

A lógica para conversão do design aos diversos tamanhos de talas funciona através do uso de funções personalizadas que realizam operações matemáticas com base nas informações do dispositivo atual, como: altura da tela do dispositivo atual, largura da tela do dispositivo atual, altura e largura (dimensões em pixels) do elemento a ser adicionado no código da aplicação no software Figma.

Também foi necessário criar outras funções e classes personalizadas ao longo do desenvolvimento do frontend. Essas classes representam elementos que vão desde sombras,

efeitos de animação, rotação, zoom, scroll entre outros.

Além dos componentes que são responsáveis pelo funcionamento das animações e adaptação para os diversos formatos de tela existentes, foram também desenvolvidos cada um dos elementos visuais que compõe as diversas telas da aplicação "Navalcare Connect" como botões, gráficos, tabelas, containers personalizado para diversas funcionalidades e elementos gráficos, listas, divisórias, planos de fundo, entre outros.

4.5.3 Validação das telas desenvolvidas a partir do design

Semanalmente, após o término de diversas versões da aplicação móvel, era realizada a publicação do códgio contendo as atualizações nas plataformas "App Store Connect" e "Google Play Console", permitindo assim o acompanhamento das novas implementações pelo time de testes internos da empresa.

Com os lançamentos de cada versão da aplicação, uma mensagem personalizada foi criada para anunciar as novas modificações, correções e adições com relação a versão anterior da aplicação. Atualmente a aplicação se encontra na versão 1.5.7.

Com o download da aplicação e testes nas plataformas iOS e Android, foi possível validar na prática o desempenho e funcionamento corretos de cada nova versão da aplicação, permitindo entregas e integração constantes no projeto, buscando sempre o feedback das partes envolvidas e do designer.

4.5.4 Adição de reatividade e regras de negócio para cada uma das telas

As regras de negócio de cada das telas da aplicação foi desenvolvida em conjunto com sua concepção, visando criar uma navegação fluída pela aplicação e animações otimizadas que são executadas de maneira fluída mesmo nos aparelhos com capacidade de processamento limitada.

Para cada tela, primeiramente foram criados componentes individuais para representar cada um dos elementos gráficos como o "Container de Baterias" citado no capítulo 2.3. A estrutura das classes dos widgets criados permitiu apresentar nas telas da aplicação dados enviados aos elementos gráficos em código através dos construtores de suas classes.

Também foram adicionadas animações, reatividades e navegação em cada um dos botões e elementos interativos das telas desenvolvidas, possibilitando visualizar nas atualizações semanais realizadas as modificações e o funcionamento correto de cada tela, utilizando dados temporários que estavam solidificados no código, prontos para a etapa de desenvolvimento do código responsável pela gestão e atualização das informações com o banco de dados, através de pacotes nativos como "firebase_database" (GOOGLE, 2023e).

4.6 Desenvolvimento do backend da aplicação

O desenvolvimento do backend da aplicação envolveu diversas etapas que foram necessárias para estabelecer o funcionamento adequado da infraestrutura utilizada para a gestão, controle e visualização dos dados utilizados pela aplicação "Navalcare Connect". Nesta seção, serão mencionadas algumas das etapas e dos sistemas desenvolvidos para cumprir este propósito, de forma a respeitar ao máximo as metodologias e requisitos iniciais traçados.

4.6.1 Autenticação dos usuários na aplicação

O passo inicial para o funcionamento adequado da aplicação é o cadastro e a autenticação de novos usuários na plataforma, a autenticação dos usuários na aplicação foi realizada através da plataforma Firebase, pelo serviço "Firebase Authentication" (GOOGLE, 2023c), que é conhecido por sua escalabilidade, baixo custo e facilidade para integração ao framework Flutter, utilizando pacotes criados e mantidos pela própria Google.

O Firebase Authentication dá acesso a inúmeras formas de realizar a autenticação dos usuários na aplicação, como por exemplo: e-mail e senha, número de celular, login anônimo, além de permitir o login utilizando serviços de terceiros como contas do Google, Apple, Facebook, GitHub, Twitter, Yahoo, Microsoft, entre outros.

Para a validação do funcionamento inicial da aplicação e seu uso durante a etapa de testes internos, foi criada um cadastro padrão utilizando um endereço de e-mail oficial da empresa e uma senha compartilhada, com o propósito de avaliar o funcionamento adequado do software desenvolvido e do acesso a aplicação, além da experiência de usuário durante as etapas de "Login" e "Onboarding". Portanto, inicialmente se teve como meta o cadastro gradual e manual dos primeiros usuários no sistema, para os testes abertos conforme a aplicação se aproxima suas etapas finais de desenvolvimento, porém em um segundo momento, é do interesse das partes envolvidas avaliar uma possível versão aberta da aplicação para usuários que não fazem parte do grupo de clientes internos atual da empresa.

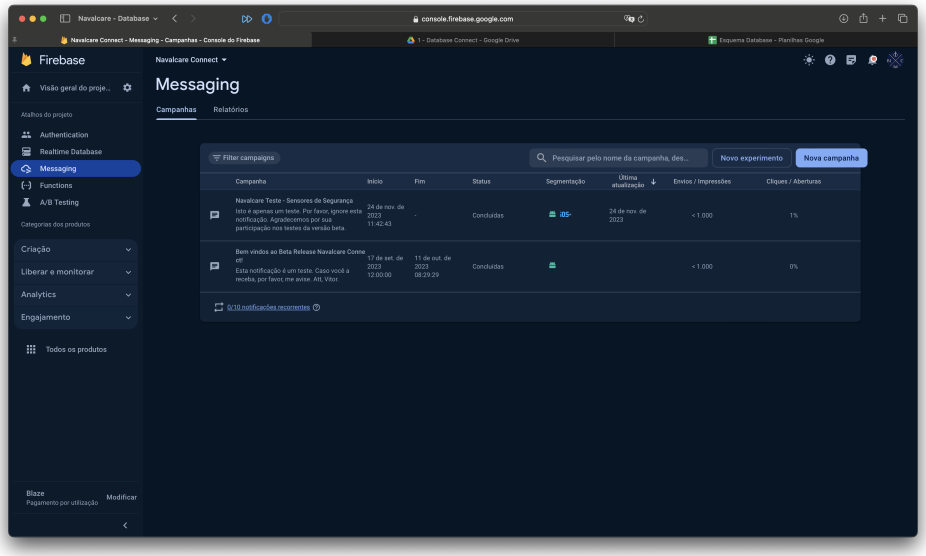
4.6.2 Envio de notificações personalizadas

Após a validação do funcionamento da autenticação, foi possível realizar o envio de notificações através do uso da plataforma "Firebase Cloud Messaging" (GOOGLE, 2023d), etapa essencial para o funcionamento adequado do sistema, uma vez que os alertas de segurança, lembretes de revisão e documentação assim como divulgação de notícias aos clientes será estabelecida por meio de notificações na aplicação "Navalcare Connect".

As notificações exigiram um processo maior para configuração nas plataformas da empresa Apple, porém, após diversos testes e pesquisa foi possível emitir e utilizar uma chave para autenticação da infraestrutura APNS (Apple Notification Service), compatível

com o serviço utilizado “Firebase Cloud Messaging” capítulo 3.5. Deste modo, a emissão de alertas de notificação pode ser realizada de duas formas, uma delas é através da segregação do público alvo, pela plataforma Firebase, com testes A/B e através da criação de novas campanhas na plataforma, a outra é através do uso de um token único de usuário atrelado a cada um dos dispositivos que fazem uso da aplicação.

Figura 27 – Captura de tela da plataforma Firebase mostrando as campanhas criadas na plataforma Firebase Cloud Mesaging para testar o funcionamento das notificações.



Fonte: Elaborado pelo autor (2023).

Os tokens para o envio das notificações são obtidos assim que o usuário efetua a autenticação na aplicação, permitindo assim, através de uma rotina personalizada, enviar ao banco de dados, no caminho relacionado ao usuário atual o token do novo dispositivo adicionado. Deste modo, no banco de dados existe uma lista que contém todos os dispositivos cadastrados e permite ao backend do sistema fazer uso dos mesmos e reagir de maneira adequada a eventos como alagamentos, incêndios, vencimentos de documentação, entre outras funcionalidades de interesse que serão gradativamente adicionadas na versão final da aplicação.

4.6.3 Criação do banco de dados

Na grande maioria das arquiteturas de software estudadas durante as etapas iniciais do projeto (capítulo 3.12), o código da aplicação possui três camadas principais: “Camada Lógica”, “Camada Visual” e a “Camada dos Dados”. Deste modo, vemos que a engenharia de dados é de fato um dos ramos de maior relevância para o funcionamento adequado, escalabilidade e utilização do projeto, viabilizando assim a eventual comercialização e

expansão do grupo de usuários para centenas de milhares ou até mesmo milhões, caso haja a necessidade. Com tudo isso em mente, foram realizados estudos extensos quanto a vantagens, desvantagens, custos, funcionamento, escalabilidade, entre outros fatores para a escolha da plataforma correta para a criação do banco de dados da aplicação. Assim, foi decidido que o banco de dados utilizado para o projeto seria o Realtime Database, também presente na plataforma Firebase e mantido pela empresa Google.

Podemos ver detalhes relacionados a escolha do banco de dados no capítulo 3.5 de referencial teórico. Porém alguns dos fatores importantes foram a escalabilidade, tempo de atraso para o acesso aos dados, integração ao framework e custos mensais relacionados a leitura, escrita e manutenção da infraestrutura.

Outras vantagem da escolha deste banco de dados foi a atualização automática (sincronização) dos dados na aplicação com o banco de dados através de "Streams" que disparam eventos na aplicação quando os dados são atualizados. Permitindo assim visualizar mudanças em tempo real (FIREBASE, 2022) e uso mensal gratuito dos serviços com até 100 instâncias conectadas ao mesmo tempo (capítulo 3.5).

4.6.4 Modelagem do banco de dados

A modelagem adotada para a arquitetura do banco de dados foi realizada com base em recomendações de especialistas, análise profunda do escopo do projeto e das possibilidades para implementação de um sistema para gestão dos dados, visto que a visualização e atualização dos dados pela interface gráfica disponível na plataforma Firebase é extremamente limitada e pouco intuitiva, devido a seu caráter não relacional e sua estrutura em formato JavaScript Object Notation (JSON). Com isso em mente, foram desenvolvidas uma série de planilhas que serviriam tanto para a gestão do banco de dados quanto para a visualização e integração dos dados com os outros sistemas utilizados pela empresa em seu contexto atual.

As planilhas foram separadas em dois grupos, planilhas de controle e planilhas desnormalizadas para a consultas na aplicação, as quais buscaram apresentar uma organização intuitiva e familiar aos membros da empresa quanto aos dados utilizados pela aplicação, permitindo assim conectar o banco de dados criado com os dados reais de cadastros dos clientes e embarcações já existentes com o uso de fórmulas, instruções personalizadas e do framework Google App Scripts (capítulo 3.8).

Para a criação destas planilhas e a organização dos dados da aplicação foram utilizadas metodologias conhecidas para modelagens de software, amplamente conhecidas e utilizadas pela indústria, como os conceitos abordados pelo Domain-Driven Design (capítulo 3.12), podemos ver um esquema inicial produzido para documentar as diversas informações e dados presentes na aplicação no PDF anexado ao apêndice D.

Essa abordagem permite aos gestores visualizar e modificar os dados de maneira prática evitando o desenvolvimento de uma interface gráfica dedicada para esta finalidade,

acelerando o tempo de produção do projeto, através da integração da planilha com o banco de dados pelo uso do framework App Scripts para a sincronização dos dados com o Realtime Database, utilizada pela a aplicação para as consultas realizadas.

A metodologia desenvolvida para a estrutura do banco de dados (utilizado pela aplicação) teve como base facilitar ao máximo o consumo dos dados de maneira a espelhar a organização dos dados de forma similar a cada uma das telas do mockup criado, recomendado entre desenvolvedores e arquitetos de software que fazem uso de bancos de dados de chave-valor ou documentos, não relacionais, consistindo em bancos de dados desnormalizadas e estruturadas para facilitar as consultas realizadas pela aplicação.

A modelagem completa de um banco de dados para todos os setores da empresa e de todas as funcionalidades propostas pela aplicação não pode ser concluída para a apresentação deste trabalho dado o tempo de projeto e o tamanho da equipe (apenas o autor, que também não possuía experiências prévias com o tópico). Porém, é possível vislumbrar que a criação modular de um banco de dados unificado para os demais setores pode e deve ser iniciada logo após a conclusão das diversas funcionalidades que envolvem o funcionamento e publicação da aplicação "Navalcare Connect" nas lojas de aplicativos.

4.6.5 Desenvolvimento da API Navalcare Weather para previsão do tempo

A previsão do tempo é um fator de grande relevância para o dia a dia dos clientes atuais da empresa, pensando nisso diversas API's de terceiros foram testadas e levadas em consideração para a escolha definitiva do serviço a ser utilizado. Dentre todas as opções analisadas o serviço Apple WeatherKit REST API apresentou grande robustez e desempenho superior aos demais concorrentes durante os testes realizados, comparando suas previsões com dados coletados em situações reais ao longo do desenvolvimento do projeto.

A REST API da Apple teve como papel fornecer conjuntos de dados como previsão do tempo diária dos próximos 7 dias e previsão do tempo ao longo das horas dos próximos 7 dias. Dentro todos os dados disponíveis em cada conjunto, foram utilizados pela aplicação foram: velocidade do vento, direção do vento, porcentagem e volume esperado de precipitação, nascer e por do sol e lua, fases da lua, visibilidade e nebulosidade. Já a API marítima da empresa stormglass.io, foi utilizada para coletar informações como temperatura da água e altura das marés ao longo das horas dos próximos 3 dias.

Por fim, com o uso do framework Express JS e das Cloud Functions e App Scripts da empresa Google, serão realizadas consultas a cada uma das API's através de uma API central denominada "Navalcare Weather", que faz parte do projeto "Navalcare Cloud" e é responsável por providenciar um acesso seguro e prático a ambos serviços realizando todo o processo de autenticação e armazenamento seguro das chaves de uso.

A escolha de uso de Cloud Functions teve como foco principal controlar o consumo mensal de requisições as API's que possuem limites de uso como o WeatherKit API da

Apple (APPLE, 2023) ou a API de marés e temperatura da água STORMGLASS.IO (STORMGLASS.IO, 2023), além de fazer o uso de bibliotecas para implementar o mecanismo de segurança e autenticação JSON Web Token necessário para o uso da API WeatherKit da Apple (JS, 2023).

4.6.6 Armazenamento e atualização dos dados com a aplicação

O acesso ao banco de dados pela a aplicação se deu através do uso de pacotes oficiais da Google para a integração da plataforma Firebase e do serviço Realtime Database com o framework utilizado.

Buscando adotar metodologias e boas práticas para padronização do acesso e gestão dos dados por cada uma das telas da aplicação, utilizou-se uma metodologia conhecida baseada no uso de classes para: modelar os dados presentes na camada visual, acessar os dados através de consultas ao banco de dados remoto ou a cache local em caso de perdas de conectividade, gestão e distribuição dos dados para as telas respeitando as regras de negócio de cada uma das funcionalidades da aplicação.

Todos os passos executados para a consulta, armazenamento e distribuição são abordados a fundo no capítulo 2.4, e se baseiam em padrões de projeto como: repositories, agregate roots, entities, value objects, BLoC's, facades, adapters, entre outros.

Como o pacote e o banco de dados Realtime Database utilizado já tem suporte para a persistência automática dos dados no dispositivo do usuário, mesmo após reinicializar a aplicação, outras metodologias como o uso de pacotes como "Hydrated BLoC" e a criação de regras de negócio personalizadas com o foco na persistência local dos dados não foram necessárias.

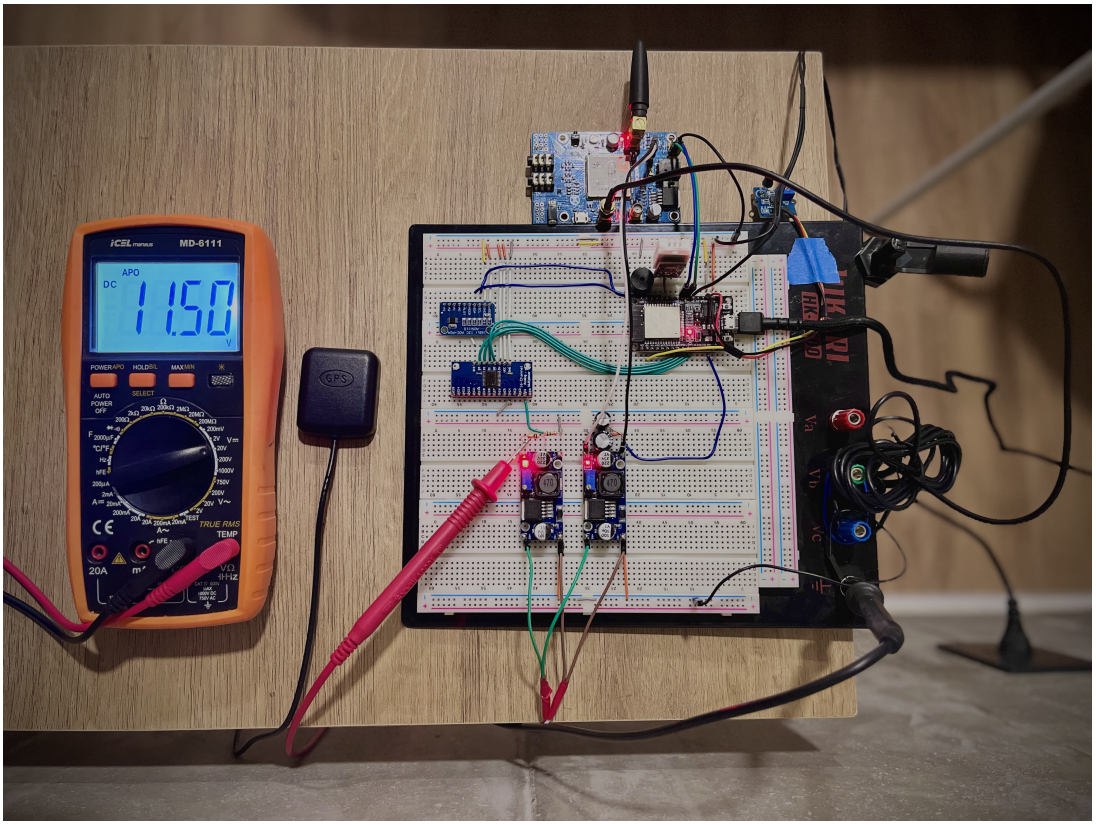
5 RESULTADOS

Este capítulo irá apresentar o funcionamento dos sistemas e alguns dos resultados obtidos com o projeto através dos testes realizados, bem como as considerações finais relacionadas ao aprendizado, experiência adquirida e perspectivas para o futuro dos projetos "Navalcare Connect" e "Navalcare Safeboat".

5.1 Testes de hardware

Para validar o funcionamento adequado do hardware e do firmware proposto, através da intercomunicação dos diversos blocos que compõe o projeto, foi desenvolvido um protótipo em bancada que fez uso dos principais sensores e componentes para coletar e enviar os dados para o banco de dados "Realtime Database".

Figura 28 – Foto do protótipo do hardware montado em bancada.



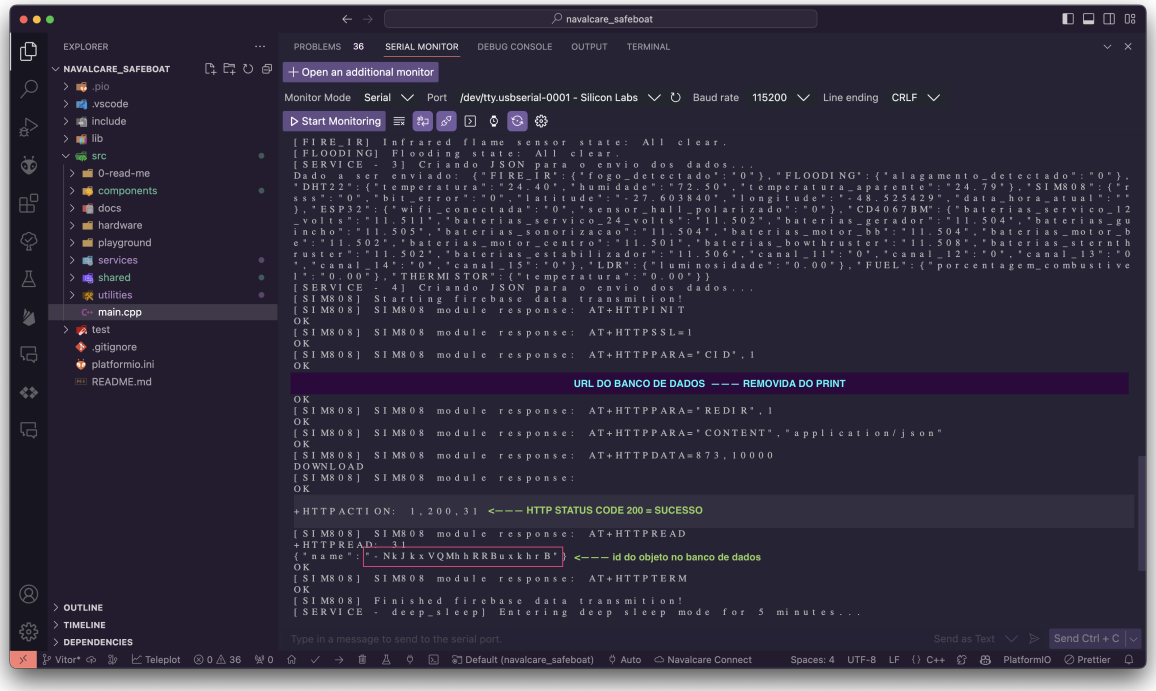
Fonte: Elaborado pelo autor (2023).

O hardware, visto na figura 28, fez uso de módulos como o modem celular SIM808, o multiplexador CD4067BM, o conversor ADS1115, sensores de chama, temperatura, sensor de nível com boia, um "buzzer" e reguladores de tensão, além de antenas necessárias para o funcionamento adequado dos sistemas GNSS e GPRS.

Na figura 28 podemos ver a tensão de 11.50V em uma das entradas do multiplexador CD4067BM, depois de passar pelo divisor resistivo composto por dois resistores de valores comerciais 100K e 10K, com atenuação de aproximadamente -90.91% do valor de entrada (-20.819172 dB), simulando um banco de serviços da embarcação.

Após a montagem adequada de todos os componentes, foi então testado na prática o funcionamento adequado do firmware para o projeto, através da interface serial USB conectada ao computador e do uso do monitor serial da extensão PlatformIO no software Visual Studio Code, da Microsoft.

Figura 29 – Captura de tela do software Visual Studio Code mostrando o monitor serial com o firmware em funcionamento.



Fonte: Elaborado pelo autor (2023).

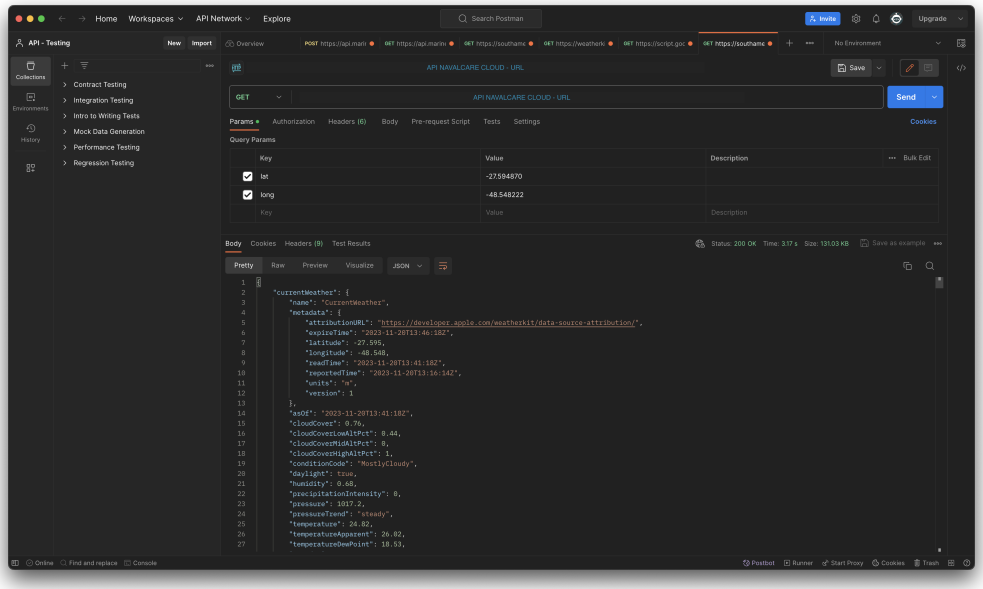
Deste modo, foi possível observar um erro de 0.011 volts na leitura do banco de baterias simulada no teste da figura 29, porém o erro apresentado se encontra dentro dos limites aceitos para o projeto, também é possível adicionar o fato de que ainda seria possível realizar a calibração do sistema possivelmente reduzindo o erro observado para limites ainda menores.

É possível observar também na imagem 29 o envio dos dados para o servidor sendo realizado através do uso do modem celular, que apresenta no buffer serial a resposta "+HTTPACTION: 1, 200, 31", onde "200" que seria o código de status padrão para operações Hypertext Transfer Protocol (HTTP) bem sucedidas (DOCS, 2023).

5.2 Testes da API Navalcare Weather

Para verificar o funcionamento adequado das API's de previsão do tempo, foi utilizado o software "Postman" para realizar as requisições HTTP GET nos links expostos para o uso de cada API. Desta forma, foram obtidas as respostas desejadas comprovando o funcionamento de cada API, na figura 30 é possível observar que a requisição feita a API também inclui coordenadas para a aquisição do dataset enviado pelas API's terceiras (Apple WeatherKit API e Stormglass.io). Na figura 30 também é possível observar que a resposta obtida foi bem sucedida, proporcionando o acesso a inúmeras informações de previsão do tempo, tanto ao longo de cada hora quanto resumos diários, tudo feito através de autenticação, gestão de segredos e requisições seguras na plataforma Google Cloud.

Figura 30 – Captura de tela do software Postman mostrando os testes das API's de previsão do tempo desenvolvidas para o projeto.



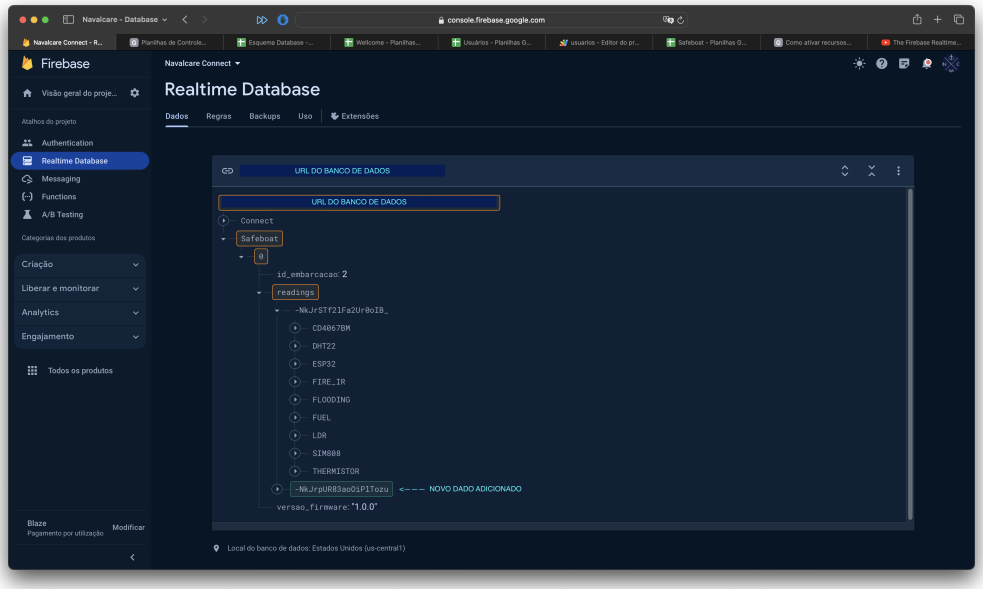
Fonte: Elaborado pelo autor (2023).

5.3 Testes do banco de dados

Para a realização dos testes com o banco de dados, inúmeros fatores foram abordados como: funcionamento adequado das planilhas de controle, funcionamento adequado do envio dos dados através dos secrets e API REST, leitura dos dados pela aplicação, atualização em tempo real dos dados com a aplicação, adição e remoção de dados através da aplicação, regras de segurança para leitura escrita e validação dos dados, entre outros.

Na figura 31 podemos observar a atualização do banco de dados após a adição de um novo dado na chave relacionada ao armazenamento das leituras realizadas pelo

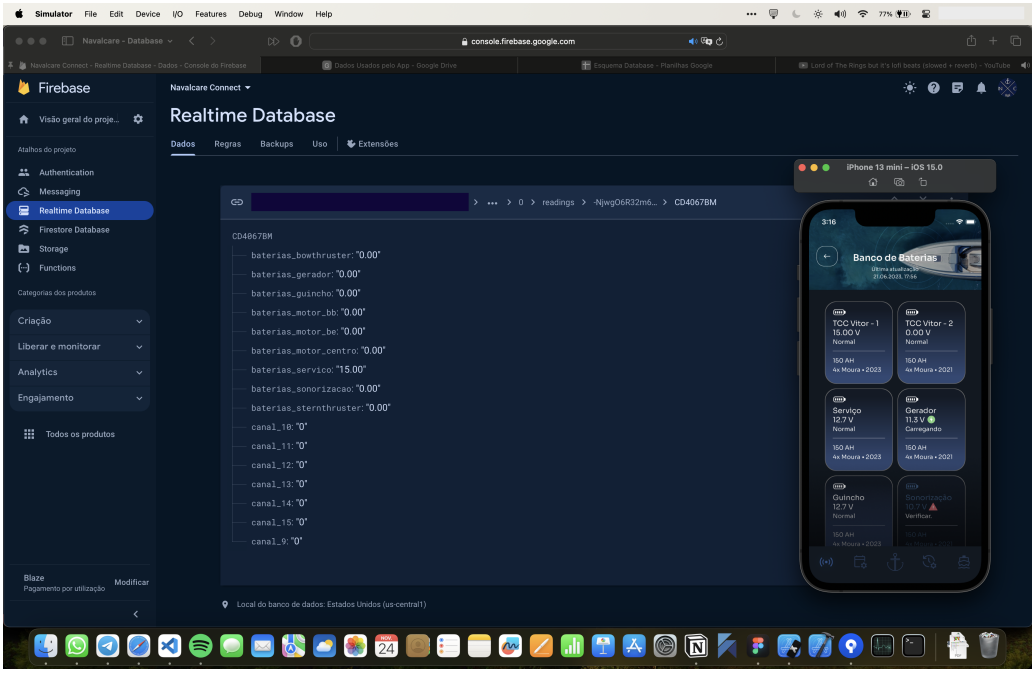
Figura 31 – Captura de tela da plataforma Firebase mostrando os testes com o banco de dados e o envio de dados a partir do hardware.



Fonte: Elaborado pelo autor (2023).

hardware “Navalcare Safeboat”.

Figura 32 – Captura de tela da plataforma Firebase mostrando os testes com o banco de dados e o envio de dados a partir do hardware.



Fonte: Elaborado pelo autor (2023).

Também é possível visualizar na figura 32 um teste realizado para a leitura em tempo real dos dados através do uso do pacote do banco de dados "Realtime Database" no framework Flutter, a tela apresenta um dashboard que foi desenvolvido para a apresentação dos diversos bancos existentes nas embarcações dos clientes, ao todo foram catalogados 11 tipos de bancos de baterias até o momento, como: banco de baterias de serviços 12V, bancos dos motores, banco de sonorização, entre outros.

5.4 Testes da aplicação Navalcare Connect

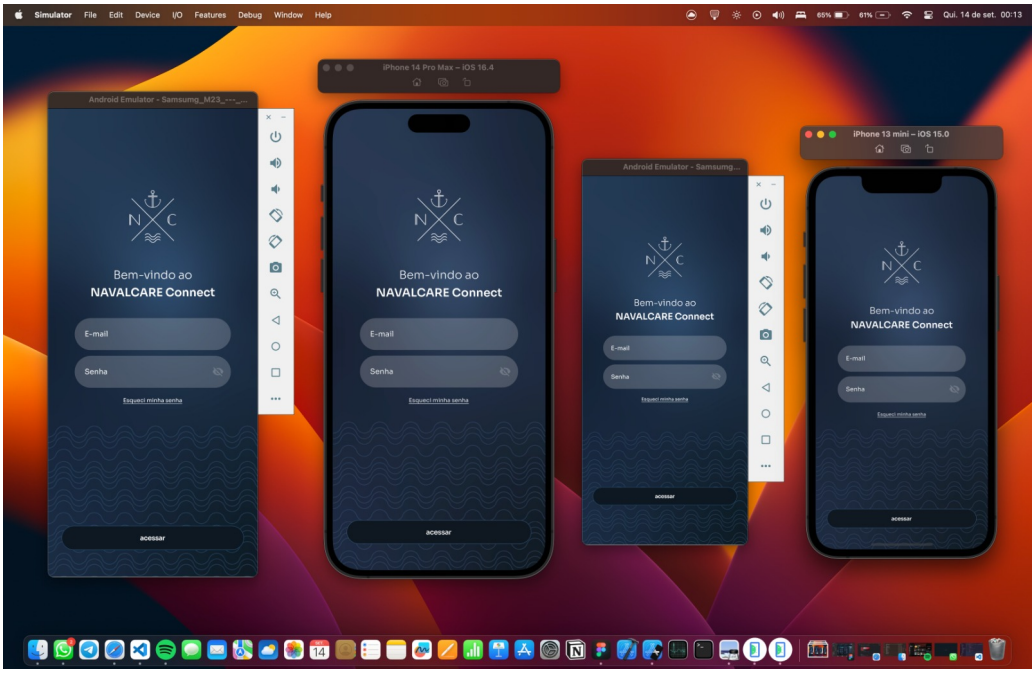
Durante o período de desenvolvimento do projeto, diversas versões da aplicação foram publicadas através das plataformas App Store Connect e Google Play Console, ambas em testes internos, para a validação do desempenho da aplicação e do funcionamento adequado da interface (frontend) de todas as telas desenvolvidas para a aplicação.

Boa parte do foco do projeto se deu no desenvolvimento do frontend de todas as telas da aplicação, visto que esta é uma das etapas que mais levam tempo para ser desenvolvidas e são o cerne de todas as outras modelagens dos sistemas que compõe o projeto. Deste modo, podemos ver nas figuras desta seção algumas das telas criadas para aplicação, que se iniciou com este trabalho de conclusão de curso e está cada dia mais próxima de se tornar um produto comercial para os clientes da empresa e futuramente para livre acesso, com monetização por meio de anúncios ou publicidades.

5.5 Hardware desenvolvido

Por fim, buscando acelerar o processo de desenvolvimento do hardware, visto que ainda existem muitas demandas para o setor de desenvolvimento de software, foi do interesse da empresa contratante, adicionar um colaborador para o time de desenvolvimento de hardware do projeto "Navalcare Safeboat", que foi supervisionado e direcionado pelo autor que atualmente ocupa o cargo de CTO (Chief Technology Officer) e desenvolvedor fullstack na empresa Navalcare. O apoio na etapa do layout da PCB foi de grande ajuda para o paralelismo no desenvolvimento do hardware embarcado, possibilitando assim o lançamento de uma versão beta da aplicação e do hardware para testes abertos aos clientes em breve. Todo o circuito do esquemático da placa de circuitos impresso foi feito em conjunto durante reuniões e boa parte do projeto já estava solidificada após os testes com o protótipo realizado em bancada antes da entrada do novo membro para o time, porém deve-se reconhecer que dada a autoria conjunta desta etapa, foi omitido detalhes aprofundados ou especificações quanto ao funcionamento do hardware dedicado.

Figura 33 – Captura de tela apresentando diversos emuladores com a tela “login” da aplicação “Navalcare Connect”.



Fonte: Elaborado pelo autor (2023).

Figura 34 – Captura de tela apresentando diversos emuladores com a tela “onboarding” da aplicação “Navalcare Connect”.

Fonte: Elaborado pelo autor (2023).

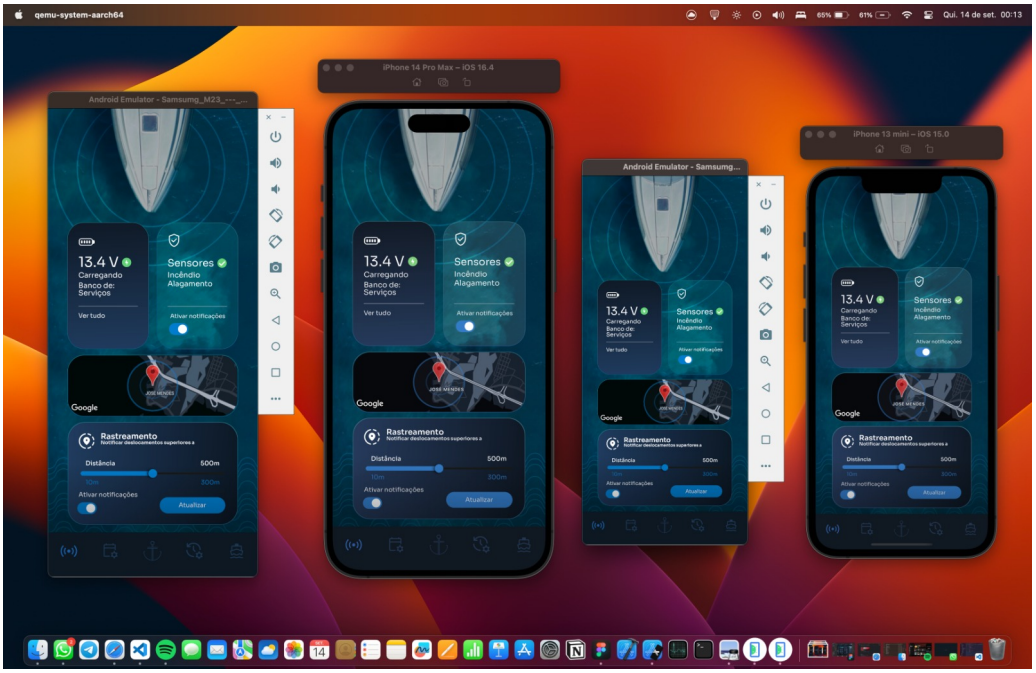
Figura 35 – Captura de tela apresentando diversos emuladores com a tela “home” da aplicação “Navalcare Connect”.

Fonte: Elaborado pelo autor (2023).

Figura 36 – Captura de tela apresentando diversos emuladores com a tela “previsão do tempo” da aplicação “Navalcare Connect”.

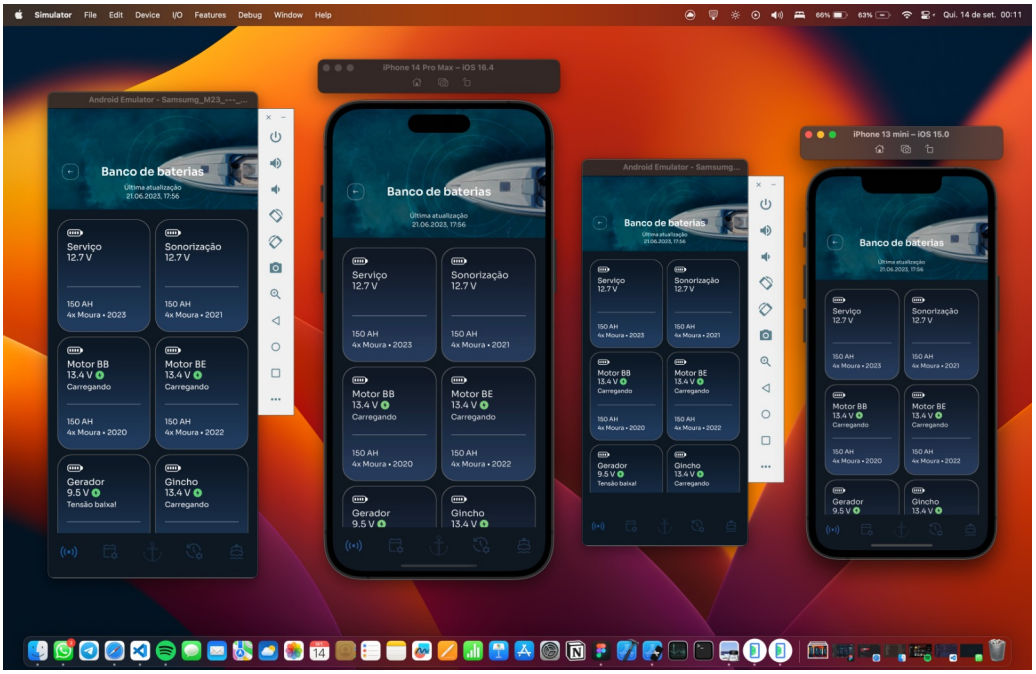
Fonte: Elaborado pelo autor (2023).

Figura 37 – Captura de tela apresentando diversos emuladores com a tela “Safeboat” da aplicação “Navalcare Connect”.



Fonte: Elaborado pelo autor (2023).

Figura 38 – Captura de tela apresentando diversos emuladores com a tela “Safeboat baterias” da aplicação “Navalcare Connect”.

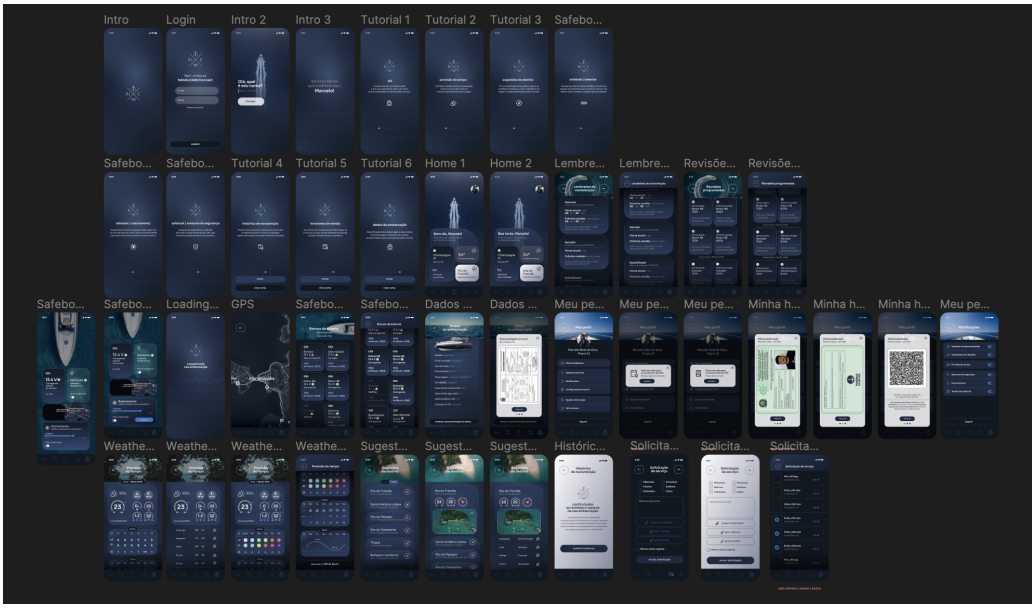


Fonte: Elaborado pelo autor (2023).

Figura 39 – Captura de tela apresentando diversos emuladores com a tela “Safeboat tracking” da aplicação “Navalcare Connect”.

Fonte: Elaborado pelo autor (2023).

Figura 40 – Modelos (mockups) de todas as telas criadas pelo designer para a aplicação “Navalcare Connect”.



Fonte: (NILO, 2023).

6 CONSIDERAÇÕES FINAIS

A proposta original deste trabalho, sempre possuiu no seu cerne a conectividade da empresa com seus clientes, de seus clientes com suas embarcações e dos bancos de dados da empresa com os sistemas buscando a concepção de "plataformas" sólidas e escaláveis que pudessem ser utilizada ao longo dos anos futuros pela empresa para o desenvolvimento de tecnologia de ponta e a entrada em diversos setores e mercados antes inexplorados. Com este objetivo em mente, a escolha correta das ferramentas utilizadas e os padrões adotados além de um estudo aprofundado sobre o estado da arte na engenharia e arquitetura de software, eletrônica e telecomunicações foi fundamental para todo o processo de desenvolvimento.

Durante a produção deste trabalho, o autor concluiu 4 cursos de capacitação de ponta: "STM32 : Internet Of Things with 4G LTE Modem - Hardware" (pela Udemy), "Engenharia de Software Moderna" (pela UFMG), Masterclass Flutterando (ministrado e criado por Jacob Moura, fundador e CEO da Flutterando maior empresa de desenvolvimento em Flutter do Brasil), Flutter & Dart - The Complete Guide [2023 Edition] (pela Udemy). Deste modo, foi possível produzir interfaces e aplicações confiáveis e fluídas, hardware dedicado e proprietário, protótipos e testes para cada um dos componentes do sistema que será futuramente comercializado, entre outras conquistas como metodologias e aprendizados documentados para os setores de software e hardware da empresa.

Com isso, inúmeras possibilidades futuras vão se abrindo, possibilitando a inclusão de diversos módulos e sistemas no projeto original com o foco na criação de um produto cada vez mais completo para manutenção inteligente (SERVICES, 2023) de embarcações.

Quanto à aplicação móvel, graças ao conteúdo produzido durante o estágio e a produção deste trabalho, será possível muito em breve realizar o lançamento oficial da plataforma "Navalcare Connect" para alguns dos clientes e eventualmente a adoção do mesmo para inúmeros processos internos e demandas dos contratantes.

Apesar do projeto ter sido capaz de validar o funcionamento correto de todos os sistemas e funcionalidades, não foi possível realizar a publicação da aplicação móvel até o presente momento, dada a escala do trabalho realizado, diversos sistemas e campos do conhecimento envolvidos, demandas adicionais, o tamanho da equipe, capital e o tempo disponível. Desta forma, temos uma perspectiva promissora para a aplicação móvel e o protótipo do hardware, os processos de desenvolvimentos internos da empresa tem se tornado cada vez mais refinados com o fortalecimento dos hábitos e conhecimentos adquiridos além do crescimento da equipe.

REFERÊNCIAS

APPLE. **Get started with WeatherKit**. 2023. Disponível em: https://developer.apple.com/weatherkit/get-started/.

BACK4APP. **As 10 principais startups que usam o Express.js**. 2023. Disponível em: https://blog.back4app.com/pt/startups-usando-express-js/.

DDD, Explore. **Flutter The BLoC Pattern: hydrated$_{b}$loc**. [*S.l.*: *s.n.*], mar. 2023. Disponível em: https://www.youtube.com/watch?v=kIKwPNKXaLU.

DESIGN, Invision. **A comprehensive guide to design systems**. 2023. Disponível em: https://www.invisionapp.com/inside-design/guide-to-design-systems/.

DOCS, mdn web. **HTTP response status codes**. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. Acesso em: 9 nov. 2023.

EMBARCADOS. **Webinar Gravado: Desligamento de redes 2G e 3G: por que você deve migrar para Cat. 1 e Cat. 1bis**. 2023. Disponível em: https://embarcados.com.br/webinar-desligamento-de-redes-2g-e-3g-por-que-voce-deve-migrar-para-cat-1-e-cat-1bis/.

FIREBASE. **Make your app the best it can be with Firebase**. [*S.l.*: *s.n.*], abr. 2022. Disponível em: https://www.youtube.com/watch?v=XHvWx1F3S4A.

FOWLER, Martin. **Domain-Driven Design Posts**. 2023. Disponível em: https://martinfowler.com/tags/domain%20driven%20design.html.

GENOVA, Salone Nautico. **Salone Nautico Genova Homepage**. 2023. Disponível em: https://salonenautico.com.

GOOGLE. **Automate extend Google Workspace with simple code.** 2023. Disponível em: https://developers.google.com/apps-script. Acesso em: 9 mai. 2023.

_____. **Criar e acessar um secret usando o Secret Manager**. 2023. Disponível em: https://cloud.google.com/secret-manager/docs/create-secret-quickstart?hl=pt-br.

_____. **Firebase Authentication**. 2023. Disponível em: https://firebase.google.com/docs/auth?hl=pt-br.

_____. **Firebase Cloud Messaging**. 2023. Disponível em: https://firebase.google.com/docs/cloud-messaging?hl=pt-br.

GOOGLE. **firebase$_database$**. 2023. Disponível em: https://pub.dev/packages/firebase_database.

_____. **Introducing Firebase Realtime Database**. [*S.l.*: *s.n.*], mai. 2016. Disponível em: https://www.youtube.com/watch?v=U5aeM5dvUpA&list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL.

_____. **Planos de preços, Firebase**. 2023. Disponível em: https://firebase.google.com/pricing?hl=pt-br. Acesso em: 9 mai. 2023.

GURU, Refactoring. **What's a design pattern?** 2023. Disponível em: https://refactoring.guru/design-patterns/what-is-pattern.

HACKOLADE. **ER2020 conference: NoSQL Data Modeling in Practice**. [*S.l.*: *s.n.*], jan. 2021. Disponível em: https://www.youtube.com/watch?v=1Z2Tfc46t8c&lc=UgyGOWB4i9k18kPFk1l4AaABAg.

INSTRUMENTS, Texas. **Data sheet CD4067BM**. 2023. Disponível em: https://www.ti.com/lit/ds/symlink/cd4067b.pdf?ts=1701168929783&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCD4067B%252Fpart-details%252FCD4067BM%253Futm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Docb-tistore-promo-asc_opn_en-cpc-storeic-google-wwe%2526utm_content%253DDevice%2526ds_k%253DCD4067BM%2526DCM%253Dyes%2526gad_source%253D1%2526gclid%253DCjwKCAiAvJarBhA1EiwAGgZl0B23gDYxYKsfbAKOctNEUercaiD-27iYhPUIaGL7UpTwAKaMVYKxjxoCzEIQAvD_BwE%2526gclsrc%253Daw.ds.

JS, NPM. **JSON Web Token**. 2023. Disponível em: https://www.npmjs.com/package/jsonwebtoken.

KOSTYRA, Philipp. **How Domain-Driven Design Can Boost Your Product Development**. 2023. Disponível em: https://medium.com/@philippkostyra/how-domain-driven-design-can-boost-your-product-development-58103623357b.

NILO. **CORE DGTL**. 2023. Disponível em: https://coredgtl.agency. Acesso em: 10 dez. 2023.

RACE, The Ocean. **The Ocean Race Homepage**. 2023. Disponível em: https://www.theoceanrace.com.

SERVICES, Advanced Technology. **Smart Maintenance**. 2023. Disponível em: https://www.advancedtech.com/blog/what-is-smart-maintenance/.

SHYMANSKYY, Volodymyr. **TinyGSM**. 2023. Disponível em: https://github.com/vshymanskyy/TinyGSM. Acesso em: 9 nov. 2023.

SIMCOM. **SIMCom SIM7600X**. 2017. Disponível em: https://www.simcom.com/product/SIM7600X.html. Acesso em: 9 nov. 2023.

STORMGLASS.IO. **Plans Pricing, Stormglass.io**. 2023. Disponível em: https://stormglass.io/pricing/.

TARGET, Tech. **What is GPRS (General Packet Radio Services)?** 2023. Disponível em: https://www.techtarget.com/searchmobilecomputing/definition/GPRS. Acesso em: 9 jul. 2023.

TV, TDD. **The S.O.L.I.D. Principles of OO Agile Design - Uncle Bob Martin**. [*S.l.*: *s.n.*], 2015. Disponível em: https://www.youtube.com/watch?v=t86v3N4OshQ&t=1s.

WIKIPEDIA. **Denormalization**. 2021. Disponível em: https://en.wikipedia.org/wiki/Denormalization. Acesso em: 9 nov. 2023.

_____. **Feature Driven Development**. 2022. Disponível em: https://en.wikipedia.org/wiki/Feature-driven_development. Acesso em: 9 nov. 2023.

_____. **Satellite navigation**. 2023. Disponível em: https://en.wikipedia.org/wiki/Satellite_navigation. Acesso em: 9 jul. 2023.

APÊNDICE A – TRATAMENTO DE ERROS

Figura 41 – Código fonte da rotina “Read Temperature” do componente DHT22.

```
#if !defined(READ_TEMPERATURE_DHT22)
#define READ_TEMPERATURE_DHT22

#include "shared/dependencies.h"

float read_temperature_dht22(bool isCelcius = true)
{
    float default_value = 0.0;

    //* Reading temperature or humidity takes about 250 milliseconds!
    //* Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    //* Read temperature as Celsius (the default)
    float t = dht.readTemperature();

    delay(250);

    //* Read temperature as Fahrenheit (isFahrenheit = true)
    float f = dht.readTemperature(true);

    delay(250);

    //* Check if any reads failed and exit early (to try again).
    if (isnan(t) || isnan(f))
    {
        serialMon.println(F("[DHT22] Failed to read DHT sensor!"));
        return default_value;
    }

    //* Debug.
    serialMon.print(F("[DHT22] Temperature - Celcius: "));
    serialMon.print(t);
    serialMon.println(F("°C "));
    serialMon.print(F("[DHT22] Temperature - Farenheit: "));
    serialMon.print(f);
    serialMon.println(F("°F"));

    //* Returns temperature in Celcius or Farenheit depending on the isCelcius flag.
    //* In case of NaN, returns default_value.
    return isCelcius ? t : f;
}

#endif // READ_TEMPERATURE_DHT22
```

Fonte: Elaborado pelo autor (2023).

APÊNDICE B – ROTINA DE INICIALIZAÇÃO PRINCIPAL

Figura 42 – Código fonte da rotina de inicialização principal do projeto.

```
1  //* ------------------------------------------------------
2  //* ---------------------- Void Setup ----------------------
3  //* ------------------------------------------------------
4  void setup()
5  {
6
7    delay(5000);
8
9    //* --- SERIAL BEGIN ---
10   //* Begin serial communication with Serial Monitor.
11   serialMon.begin(115200);
12   //* Begin serial communication with SIM808 and esp32.
13   serialCelular.begin(9600, SERIAL_8N1, rx_celular, tx_celular);
14   //* Begin serial communication with Serial2 and esp32.
15   // serialGNSS.begin(9600, SERIAL_8N1, rx_gnss, tx_gnss);
16
17   serialMon.println("\n-------------------------------------------------");
18   serialMon.println("---------- Void Setup STARTED! ----------");
19   serialMon.println("-------------------------------------------------\n");
20
21   //* Beginning startup routine.
22   make_beep_noise_buzzer(1);
23
24   //* --- INIT COMPONENT SETUP ---
25   setup_ADS1115();
26   setup_BUZZER();
27   setup_CD4067BM();
28   setup_DHT22();
29   setup_ESP32();
30   setup_FIRE_IR();
31   setup_FLOODING();
32   setup_FUEL();
33   setup_LEDS();
34   setup_LDR();
35   setup_RELAY();
36   setup_SIM808();
37   setup_THERMISTOR();
38
39   serialMon.println("\n-------------------------------------------------");
40   serialMon.println("---------- Void Setup Finished! ----------");
41   serialMon.println("-------------------------------------------------\n");
42
43   //* Finished startup routine.
44   make_beep_noise_buzzer(3);
45 }
```

Fonte: Elaborado pelo autor (2023).

APÊNDICE C – ROTINA PARA IMPLEMENTAÇÃO DA FSM.

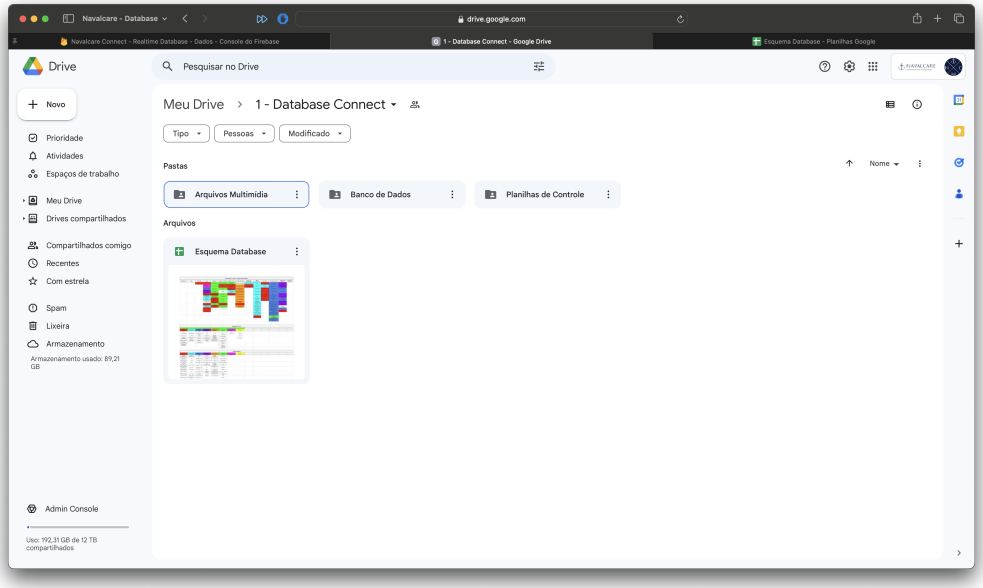
Figura 43 – Código fonte da rotina principal para implementação da máquina de estados.

```
1  //* ------------------------------------------------------
2  //* ---------------------- Void Loop ----------------------
3  //* ------------------------------------------------------
4  void loop()
5  {
6    serialMon.println("\n------------------------------------------------");
7    serialMon.println("---------- Void Loop COMEÇOU! ----------");
8    serialMon.println("------------------------------------------------\n ");
9
10   //? ---------------------- FSM IMPLEMENTATION ----------------------
11
12   //? 1. Estabelecer conexão com a rede móvel.
13   estabelecer_conexao();
14
15   //? 2. Realizar sensoriamento da embarcação.
16   realizar_sensoriamento();
17
18   //? 3. Criar JSON para o envio dos dados.
19   criar_json();
20
21   //? 4. Enviar dados coletados para o servidor.
22   enviar_dados();
23
24   //? 5. Deep Sleep for 5 min.
25   deep_sleep();
26
27   //? Repeat.
28   serialMon.println("\n------------------------------------------------");
29   serialMon.println("---------- Void Loop TERMINOU! ----------");
30   serialMon.println("------------------------------------------------\n ");
31 }
```

Fonte: Elaborado pelo autor (2023).

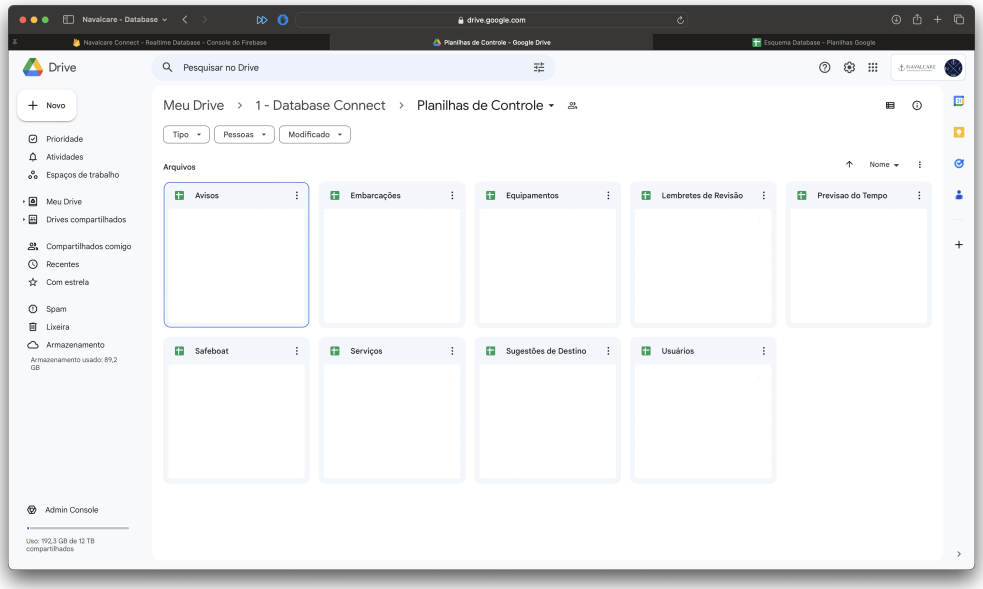
APÊNDICE D – ESQUEMA DO BANCO DE DADOS.

Figura 44 – Visualização geral do banco de dados no Google Drive da empresa.



Fonte: Elaborado pelo autor (2023).

Figura 45 – Planilhas de controle para gestão dos cadastros e dados do sistema.



Fonte: Elaborado pelo autor (2023).

Esquema Database

Navalcare Connect - Dados das Páginas

Splash Page	Login	Onboardig	Home	Safeboat	Safeboat - Baterias	Safeboat - Mapa	Próximas Revisões	Histórico de Manutenções	Minha Embarcação	Meu Perfil	Previsão do Tempo	Sugestões de Destino	Solicitar um Serviço
-	-	Nome	Nome	Tensão Bat	Data Ult. Atualização	LatLong Embarcação	~~Data Ult. Atualização~~	URL PDF Histórico	Foto	Nome	Temp. Atual	Nome destino	
			Mensagem de Aviso	Status Bat	Nome	LatLong Ancoragem Virtual	Horas de Uso Atuais	**Id Embarcação**	Modelo	Modelo	Temp. Máx. dia	Direção do vento média prox 6h algoritmo	
			Temperatura Atual	Nome Bat	Tensão	**Id Embarcação**	Data de Vencimento da Revisão		Nº Inscrição	Pdf Habilitação	Temp. Min. dia	Temp. Máx. dia	
			Nome Melhor Sugestão de Destino	LatLong Embarcação	~~Status~~	Nome da Embarcação	Horas de Uso Para o Vecimento da Revião		Ano do Casco	Destinos Favoritos	Nascer do Sol	Temp. Min. dia	
			Estado Safeboat	LatLong Ancoragem Virtual	Capacidade	Data Ult. Atualização	Nome Equipamento		Comprimento	Status Notificações	Por do Sol	Lista URL Fotos	
			Modelo da Embarcação	Status Sensores Segurança	Quantidade		Data Ult. Revisão		Boca	Dados Usuário Editar	Máxima da maré e horário	Descrição	
			Nome da Embarcação	Status Notificação Sensores Segurança	Marca		Hora Ult. Revisão		Motorização e Gerador	**Id Usuário**	Mínima da maré e horário	Intensidade do vento ao longo das horas	
			~~Horário do Dia~~	Status Notificação Ancoragem Virtual	Ano		Responsável Ult. Revisão		Capacidade Combustível		Temperatura da água	LatLong Destino	
			Foto Perfil	Sensores Instalados	**Id Embarcação**		**Id Usuario**		Capacidade Água Doce		Intensidade do vento ao longo das horas	Ângulo de Cobertura	
			Estado da Embarcação	Data Ult. Atualização	Número de bancos		**Id Equipamento**		Tensão Nominal do Sistema DC		Intensidade de rajada ao longo das horas	~~Distância~~	
			Id Embarcação						Tensão Nominal do Sistema AC		Direção do vento ao longo das horas	Temp. Atual	
			Id Usuário						HIN Number do Chasi		Clima ao longo das horas	**Id Embarcação**	
									PDF Documento da Embarcação		Precipitação em mm/h ao longo das horas		
									Validade do TIE		Maré ao longo das horas		
									Id Embarcação		LatLong Embarcação		
											Data Atual		
											Temperatura ao longo das horas		

Agregate Roots

Usuário	Embarcação	Previsão do Tempo	Sugestão de Destino	Lembretes de Revisão	Safeboat	Avisos	Serviços
id	id	id	id	id	id	id	id
id_embarcacao	disponibilidade	data_ultima_atualizacao	nome	nome	FIRE_IR	mensagem	descricao
id_firebase_auth	id_safeboat	CLIMA	VENTOS	ULTIMA REVISAO	FLOODING		url_arquivos
DADOS CADASTRAIS	url_historico_manutencao	ASTROS	COORDENADAS	PRÓXIMA REVISAO	DHT22		data_adicao
CERCA VIRTUAL	DADOS CADASTRAIS	MARES	DADOS TURISMO	ATUALIZACOES PREVENTIVAS	THERMISTOR		
STATUS NOTIFICACOES		VENTOS			SIM808		
DESTINOS FAVORITOS					ESP32		
					CD4067BM		
					ADS1115		

Value Objects

Usuário	Embarcação	Previsão do Tempo	Sugestão de Destino	Equipamento	Safeboat	Avisos	Serviços
STATUS NOTIFICACOES	DADOS CADASTRAIS	CLIMA	VENTOS	ULTIMA REVISAO	FIRE_IR		
status_notificacao_seguranca	nome	temperatura_atual	angulo_inicial_cobertura	data_ultima_revisao	fogo_detectado		
status_notificacao_documentacao	url_foto	temperatura_maxima	angulo_final_cobertura	horas_ultima_revisao	FLOODING		
status_notificacao_revisoes	modelo	temperatura_minima	direcao_vento_algoritmo	responsavel_ultima_revisao	alagamento_detectado		
status_notificacao_previsao_tempo	numero_inscricao	temperatura_horas	COORDENADAS	PRÓXIMA REVISAO	DHT22		
status_notificacao_rastreamento	ano_casco	temperatura_h2o	lat	data_proxima_revisao	temperatura		
status_notificacao_tensao_baterias	comprimento	nebulosidade_horas	long	horas_uso_proxima_revisao	humidade		
token_notificacao	boca	precipitacao_horas	DADOS TURISMO	ATUALIZACOES PREVENTIVAS	temperatura_aparente		
CERCA VIRTUAL	dados_motor	ASTROS	urls_fotos	data_ultima_atualizacao	THERMISTOR		
lat_cerca_virtual	dados_gerador	horario_nascer_sol	descricao	horas_uso_atuais	temperatura		
long_cerca_virtual	capacidade_combustivel	horario_por_sol			SIM808		
raio_cerca_virtual	capacidade_agua_doce	horario_nascer_lua			rsss		
DESTINOS FAVORITOS	tensao_nominal_dc	horario_por_lua			bit_error		
id1	tensao_nominal_ac	MARES			lat		
id2	hin_number	mare_maxima			long		
id3	url_documento	mare_minima			data_hora_atual		
...	validade_tie	VENTOS			ESP32		
DADOS CADASTRAIS		velocidade_vento_horas			wifi_conectada		
nome		velocidade_rajada_horas			sensor_hall_polarizado		
url_foto_perfil		direcao_vento_horas			CD4067BM		
url_habilitacao					tensao_canal_0		
					tensao_canal_1		
					tensao_canal_2		
					tensao_canal_3		
					tensao_canal_4		
					tensao_canal_5		
					tensao_canal_6		
					tensao_canal_7		
					tensao_canal_8		
					tensao_canal_9		
					tensao_canal_10		
					tensao_canal_11		
					tensao_canal_12		
					tensao_canal_13		
					tensao_canal_14		
					tensao_canal_15		
					ADS1115		
					tensao_canal_1		
					tensao_canal_2		
					tensao_canal_3		
					OBS: canal_0 dedicado ao CD4067BM		