

**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA
SOUZA**

ESCOLA TÉCNICA ESTADUAL DA ZONA LESTE

MTEC Desenvolvimento de Sistemas AMS

Erick Ferreira Lima

Gustavo Rodrigues Leite Da Silva

Hernandes Arthur Da Silva Santos

**ESTOK: automatização de controle de produtos em mercados
autônomos com RFID**

São Paulo

2025

Erick Ferreira Lima

Gustavo Rodrigues Leite Da Silva

Hernandes Arthur Da Silva Santos

**ESTOK: automatização de controle de produtos em mercados
autônomos com RFID**

Trabalho de Conclusão de Curso
apresentado à Etec da Zona Leste,
como requisito parcial para a obtenção
do título do curso de Análise e
Desenvolvimento de Sistemas, sob
orientação do Professor Jeferson
Roberto de Lima.

São Paulo

2025

ERICK FERREIRA LIMA
GUSTAVO RODRIGUES LEITE DA SILVA
HERNANDES ARTHUR DA SILVA SANTOS

**ESTOK: automatização de controle de produtos em mercados
autônomos com RFID**

**Trabalho de Conclusão de Curso apresentado à Etec da Zona Leste, como
requisito parcial para a obtenção do título do curso de Análise e
Desenvolvimento de Sistemas, sob orientação do Professor Jeferson Roberto
de Lima.**

Aprovado em ____/____/____.

Banca Examinadora:

Prof.(a) [Nome do Orientador(a)]

Etec [Nome da Etec]

Prof.(a) [Nome do membro 1 da banca]

Etec [Nome da Etec]

Prof.(a) [Nome do membro 2 da banca]

Etec [Nome da Etec]

DEDICATÓRIA

Dedicamos este trabalho aos nossos professores, por nos orientarem e dividirem seu conhecimento exercendo essa profissão tão nobre e aos nossos colegas pela colaboração ao longo do ano.

AGRADECIMENTOS

Erick: Agradeço a Deus, por me fortalecer em cada etapa desta jornada. Nos momentos de incerteza, cansaço ou baixa motivação, encontrei Nele a calma e a direção para seguir.

Aos meus pais, que sempre foram minha base. Sou grato pelo apoio constante, pela compreensão nos dias difíceis e pelo esforço — emocional e financeiro — que permitiu que eu continuasse estudando e avançando. Cada conquista minha carrega um pouco do que vocês plantaram.

À Letícia, pela presença leve e firme. Nos períodos mais intensos e desgastantes, foi ela quem trouxe paz, equilíbrio e me lembrou do propósito quando o ritmo parecia pesado demais. Sua companhia tornou essa caminhada mais possível.

Aos meus colegas de grupo, Gustavo e Hernandes, pela parceria construída ao longo de todo o processo. Foram muitas pressões, prazos e desafios, mas também união, respeito e momentos que tornaram o percurso mais leve.

A todos que, de alguma forma, fizeram parte desse trajeto e contribuíram para que este trabalho se concretizasse, deixo minha gratidão.

Gustavo: Gostaria de agradecer de forma primordial a Deus, que me deu força, serenidade e foco para persistir em meus estudos, além da direção e do apoio que me deu nos momentos de maior fraqueza, incerteza e tristeza. Nada do que foi construído hoje seria possível sem Seu apoio.

Aos meus pais, por todo o suporte emocional e financeiro que me permitiu continuar estudando, pela compreensão nos momentos mais difíceis de minha jornada e pela orientação em cada decisão importante. Tudo o que fizeram por mim hoje se transforma em conquistas ricas.

À minha irmã, Lorena Rodrigues, que me deu diversão, brincadeiras e risadas para que eu pudesse me tranquilizar nos momentos de maior pressão e me lembrar de que nem tudo é estudar — a diversão também tem seu papel.

À minha namorada, Ananda Holanda, que me trouxe paz, tranquilidade e felicidade, permitindo que eu tivesse um lugar seguro para descansar quando tudo

parecia pesado demais. Se hoje entrego o meu melhor, é fruto de toda a companhia e estabilidade emocional que ela me forneceu durante essa jornada. Em todos os momentos em que desacreditei do meu potencial, havia ela para me lembrar quem eu sou, ajudar-me a levantar e persistir no desafio, não importando o quão grande fosse. Sou grato para sempre por toda a paz que me trouxe nos momentos de pressão e dificuldades e, principalmente, pelo carinho, compreensão, atenção e cuidado que teve comigo.

Ao meu tio, Weslley Rodrigues, que desde o início de minha carreira me auxiliou e me deu suporte técnico, emocional e financeiro. Hoje, meus trabalhos e meu modo de agir carregam um pouco de cada ensinamento que me foi dado por ele durante todos esses anos.

Aos meus colegas de grupo, Hernandes Arthur e Erick Lima, com quem tive o prazer de construir uma amizade forte e duradoura, que passou por todos os momentos de complicações. Cada conquista foi fruto de muito trabalho em grupo. Quando tudo parecia impossível, nossa união foi mais forte, permitindo-nos superar cada desafio.

Sou grato a todos que participaram e contribuíram comigo de forma produtiva — seja técnica, emocional ou financeiramente. A todos que me ajudaram a alcançar mais essa etapa da minha vida, sou eternamente grato.

Hernandes: Agradeço a Deus por me manter forte, a minha família por me manter motivado a continuar minha caminhada acadêmica e aos amigos que me motivaram a continuar e dividiram os momentos difíceis.

Um agradecimento especial aos meus colegas Gustavo e Erick, a contribuição em grupo que possibilitou a criação desse trabalho não seria possível sem vocês. Me orgulho de compor esse time.

“Só se pode alcançar um grande êxito quando nos mantemos fiéis a nós mesmos.”

– Friedrich Nietzsche

ESTOK: automatização de controle de produtos em mercados autônomos com RFID

Resumo

O Estok é um sistema desenvolvido para automatizar o controle de produtos em prateleiras de mercados autônomos utilizando tecnologia RFID. A solução integra leitor RFID, microcontrolador ESP32, backend em Node.js e dashboard em React, permitindo registrar a movimentação de entrada e saída de itens em tempo real. O projeto aborda desde a diagramação UML até a construção do hardware e do software, evidenciando como a automação reduz a ruptura de estoque, aumenta a rastreabilidade e melhora a experiência do gerente. O estudo reforça a importância da automação logística para o avanço dos mercados autônomos no Brasil.

Palavras-chave: RFID; IoT; controle de estoque; mercados autônomos

ESTOK: Automating product control in autonomous markets with RFID

ABSTRACT

Estok is a system developed to automate product control on shelves in autonomous markets using RFID technology. The solution integrates a UHF reader, ESP32 microcontroller, Node.js backend, and a React dashboard, enabling real-time registration of product entries and exits. The project covers the entire structure, from UML modeling to hardware and software implementation, demonstrating how automation reduces stockouts, increases traceability, and improves managerial decision-making. This study highlights the relevance of IoT and logistics automation for the evolution of autonomous markets in Brazil.

Keywords: RFID; IoT; inventory control; autonomous markets

LISTA DE ILUSTRAÇÕES

Figura 1 — Diagrama de casos de uso sistema de biblioteca.....	15
Figura 2 — Diagrama de atividade empréstimo de livro.....	17
Figura 3 — Diagrama de sequência empréstimo de livro.....	19
Figura 4 — Diagrama de classe Estok.....	21
Figura 5 — Leitor RFID e Microcontrolador ESP32.....	23
Figura 6 — Placa ESP32 com anotações de componentes.....	25
Figura 7 — Código que exibe um texto em C++.....	26
Figura 8 — Resultado do código que exibe um texto em C++.....	26
Figura 9 — Estrutura básica de uma página em HTML.....	28
Figura 10 — Exemplo de uma tela HTML para cadastro de produtos.....	29
Figura 11 — Resultado do código HTML de uma página de cadastro de produtos.....	30
Figura 12 — Exemplo de conexão entre HTML e CSS.....	31
Figura 13 — Exemplo de escrita na CSS.....	32
Figura 14 — CSS da tela de cadastro de produtos.....	33
Figura 15 — Resultado da tela de cadastro HTML e CSS em conjunto.....	34
Figura 16 — Componente React.....	36
Figura 17 — Renderização componente React.....	36
Figura 18 — Função juntar nome e sobrenome Node.js.....	38
Figura 19 — Resultado função Node.js.....	38
Figura 20 — Código SQL Criação e inserção de dados tabela de Produtos com consulta...	39
Figura 21 — Resultado consulta banco de dados tabela Produtos.....	39
Figura 22 — Diagrama casos de uso Estok.....	42
Figura 23 — Representação tridimensional Estok em prateleira.....	46
Figura 24 — Fluxograma código embarcado C++.....	48
Figura 25 — Diagrama sequência etapas da lógica de aplicação.....	49
Figura 26 — Lógica de retorno por necessidade de reposição Node.js.....	50
Figura 27 — Consulta de produtos otimizada.....	51
Figura 28 — Protótipo de dashboard Figma.....	52
Figura 29 — Tela principal dashboard React.....	53
Figura 30 — Página de histórico.....	54

LISTA DE QUADROS

Quadro 1 — Requisitos funcionais.....	40
Quadro 2 — Requisitos não funcionais.....	41
Quadro 3 — Documentação do Caso de Uso: UC01 Listar Produtos.....	43
Quadro 4 — UC01 Cenário Principal.....	44
Quadro 5 — UC01 Cenário Alternativo Listar Produtos Acabando.....	45
Quadro 6 — UC01 Cenário Alternativo Visualizar Histórico de Prateleira.....	45

LISTA DE ABREVIATURAS E SIGLAS

ASCII – American Standard Code for Information Interchange

CSS – Cascading Style Sheets

DNS – Domain Name System

FTP – File Transfer Protocol

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

I/O – Input/Output

IDE – Integrated Development Environment

IoT – Internet of Things

RFID – Radio-Frequency Identification

UHF – Ultra High Frequency

URL – Uniform Resource Locator

ESP32 – Módulo microcontrolador com Wi-Fi e Bluetooth

SQL – Structured Query Language

SPA – Single Page Application

MVC – Model-View-Controller

JSON – JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO.....	11
2 REFERENCIAL TEÓRICO.....	13
2.1 Automatização No Controle De Estoque.....	13
2.2 UML, Linguagem De Modelagem Gráfica.....	13
2.3 Diagrama de casos de uso.....	14
2.4 Diagrama de atividade.....	16
2.5 Diagrama de sequência.....	18
2.6 Diagrama de classe.....	20
2.7 Internet Das Coisas (IoT).....	21
2.8 Identificação por rádio frequênci.....	22
2.9 ESP32.....	23
2.10 Arduino IDE.....	24
2.11 C++.....	24
2.12 HTML.....	26
2.13 CSS.....	29
2.14 JavaScript.....	34
2.15 React.....	34
2.16 Node.js.....	36
2.17 Banco de dados.....	37
3 DESENVOLVIMENTO.....	39
3.1 Requisitos do Sistema.....	39
3.2 Diagramação.....	40
3.3 Dispositivo físico.....	43
3.4 Back-end.....	44
3.5 Front-end.....	44
3.6 Testes e validações.....	44
REFERÊNCIAS.....	45
GLOSSÁRIO.....	50
ANEXOS.....	54

1 INTRODUÇÃO

O Estok consiste em uma ferramenta que utiliza a tecnologia de identificação por radiofrequência (RFID) para aumentar a rastreabilidade de produtos em prateleiras de mercados autônomos, diminuindo perdas por ruptura de estoque e vencimento. O objetivo geral é desenvolver um dispositivo para detectar a entrada e saída de produtos da prateleira, otimizando a reposição e evitando a perda de produtos próximo por vencimento ou a falta de produtos, o que é chamado de ruptura de estoque, resultado de um produto indisponível na prateleira, porém existente no estoque.

Os objetivos específicos para a concretização deste projeto consistem em produzir um dispositivo capaz de identificar a quantidade de produtos em uma prateleira, essa identificação possibilitará notificar o gerente dos produtos prestes a acabar ou que estão perto da data de vencimento, notificações essas que deverão ser dispostas em uma página web para a visualização do gerente.

O aumento da popularidade de mercados autônomos no Brasil fez do país um dos protagonistas nesse setor. Esse aumento de popularidade é atribuído à facilidade e segurança promovidas por esse modelo de negócio, sendo um meio tanto de fonte de renda para o dono/franqueado, quanto uma fonte de comodidade para os clientes (LEÃO, 2024). No entanto, se essa automatização não for feita para garantir um real autonomia, problemas como dependência de verificação constante da mercadoria disponível nas prateleiras e da quantidade em estoque por parte do gerente do mercado autônomo podem surgir, afetando não somente a comodidade do gerente, mas também a dos clientes ao se depararem com a ruptura de estoque.

Problemas como esse devem ser contornados com o uso de novas tecnologias, buscando trazer um aumento de eficiência, principalmente na parte logística (ROVAROTO, 2024). Pesquisas serão feitas utilizando o rigor científico, pois, como afirma Lakatos e Marconi (2003), para estabelecer a realidade do nosso cenário e se de fato nosso projeto é eficiente, devemos seguir o tratamento científico para conhecer a realidade. Neste estudo, a principal hipótese se dá na produção e implementação de um aparelho que aumente a rastreabilidade dos produtos, buscando mitigar os problemas já mencionados, causados pela falta de rastreabilidade de produtos em mercados autônomos.

O presente estudo visa desenvolver um meio para mitigar a problemática dos clientes que sofrem com a ruptura de estoque e, portanto, responder como a tecnologia RFID pode aumentar a rastreabilidade e reduzir rupturas de estoque em mercados autônomos de São Paulo.

A tecnologia de rádio frequência vem sendo utilizada no setor varejista, promovendo um aumento de eficiência atrelado ao aumento da rastreabilidade de produtos. Uma das pioneiras na implementação dessa tecnologia, as lojas Renner, que já desfrutam de 100% de suas lojas utilizando essa tecnologia, registrou melhoria de 64% na acuracidade dos estoques, agilizando o trabalho de reposição e evitando a ruptura de estoque (INFORCHANNEL, 2022).

Diante do cenário apresentado, o uso de tecnologias na área logística para o aumento da eficiência se faz crucial para um desenvolvimento contínuo do modelo de mercados autônomos e de outras evoluções no setor varejista.

2 REFERENCIAL TEÓRICO

A atual seção irá analisar o cenário atual de pesquisa sobre mercados autônomos. Apresentaremos a proposta da integração de um sistema baseado na tecnologia de identificação por rádio frequência (RFID), visando mitigar a falta de automatização no monitoramento de estoque em mercados autônomos.

2.1 Automatização no controle de estoque

No Brasil, o setor varejista enfrenta problemas de administração de estoque que acarretam prejuízos para as empresas e em perda de qualidade da experiência do consumidor, R\$34,9 bilhões de prejuízo foram registrados em 2024 (KPMG, 2024). Segundo Wenceslau (2024), esses problemas podem ser atribuídos à imprecisão de processos orientados à ação humana ou ao mau funcionamento de softwares.

A automação no controle de estoque se torna essencial para o desenvolvimento do setor e para diminuição de problemas de imprecisão. A seguir buscaremos explicar uma das possíveis soluções para esse problema. Com foco em mercados autônomos, a solução aumentará a autonomia na reposição de estoque, resultando na diminuição da ruptura de estoque que beneficiará os clientes dos estabelecimentos.

É importante notar também a contribuição da pesquisa nessa área para o incentivo à busca e desenvolvimento de soluções e inovações em mercados autônomos (TORTORELLI, 2018). Em vista do desenvolvimento e crescimento dessa modalidade no Brasil, e da iminente mudança na atual interação do cliente com o varejo causada pela implementação de novas tecnologias (LEÃO, 2024), pesquisas referentes a esse setor da indústria se tornam primordiais para a criação de uma base sólida de conhecimentos.

2.2 Diagramação

Para iniciar esse projeto, primeiro precisaremos de um bom planejamento para determinarmos quais as funcionalidades que o nosso sistema terá e como ele deve interagir com agentes externos. Uma boa forma de visualizar essa interação é por meio de diagramas, mais especificamente diagramas de Linguagem de Modelagem Unificada do inglês Unified Modeling Language (UML).

A UML é uma forma de descrever o seu projeto de forma gráfica para facilitar a compreensão de outras pessoas. Ela ajuda a definir os processos a serem seguidos no desenvolvimento do software (FOWLER, 2000).

Independente da complexidade do projeto, é importante desenvolver diagramas, pois ajudará na escalabilidade desse software. Os diagramas contemplarão tanto a descrição geral do projeto quanto detalhes específicos do sistema (GUEDES, 2018).

É importante ressaltar que os diagramas suportam desde uma visualização mais simples quanto uma visualização extremamente detalhada, aumentando sua complexidade.

A seguinte abordagem dos diagramas busca explicar o que utilizaremos, poupando o leitor de explicações de características, componentes e funcionalidades que não contribuíram para o real uso no nosso projeto. Dada a característica de esclarecer diferentes níveis de abstração, explicarei os diagramas em ordem de complexidade.

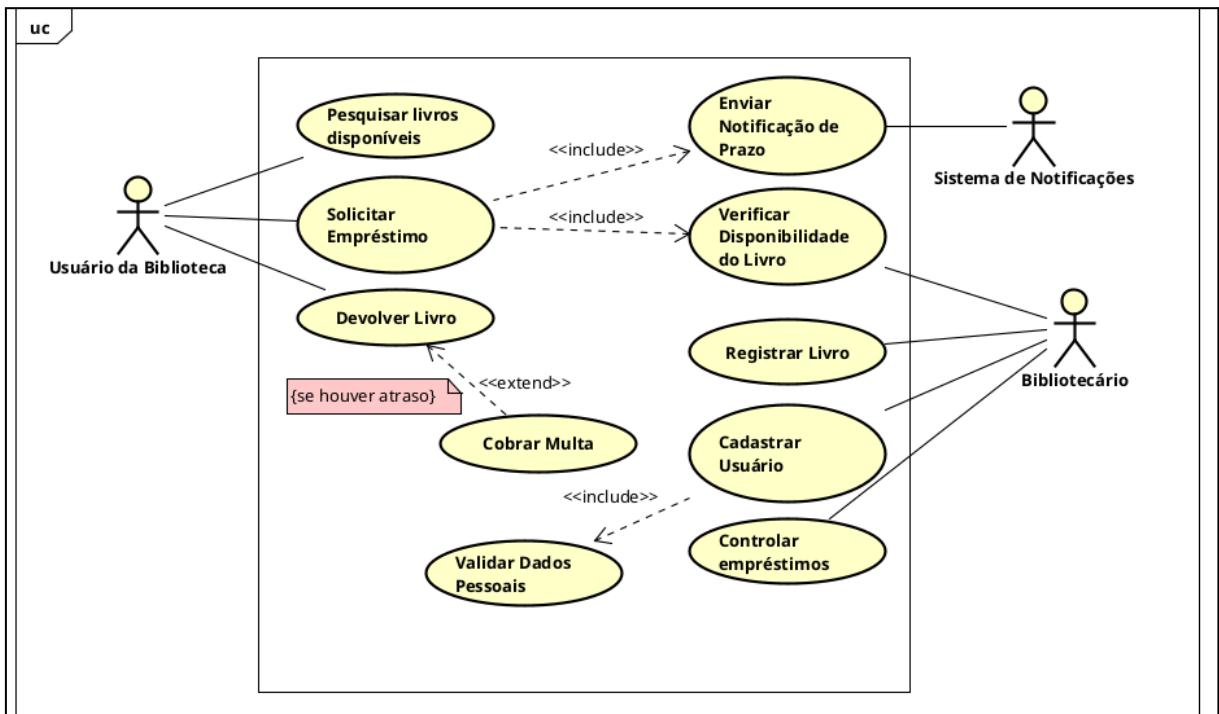
2.3 Diagrama de casos de uso

Um diagrama de caso de uso é a descrição dos cenários envolvendo o projeto em questão, sendo esses cenários as interações em que os usuários, ou outras entidades, terão com o seu sistema (FOWLER, 2000).

O diagrama de casos de uso serve para começar a visualizar as interações que existirão, possibilitando a delimitação e definição de como lidaremos com elas.

Para representar essas interações, utilizaremos os seguintes elementos. Atores, casos de uso e associações. Com esses três elementos é possível criar uma vasta possibilidade de interações, como demonstrado na figura 1.

Figura 1 — Diagrama de casos de uso sistema de biblioteca



Fonte: Do próprio autor, 2025

Para compreender esse diagrama de caso de uso, primeiro começaremos definindo cada componente dentro dele. Os principais componentes representados na figura são atores, casos de uso e associações de inclusão e de extensão.

De acordo com Guedes (2018), os atores representam os usuários do sistema, podendo também representar funções ou softwares específicos que interagem com o sistema. No caso do diagrama representado na figura, os atores são usuários da biblioteca, bibliotecário e sistema de notificação, sendo o primeiro ator primário, portanto representados do lado esquerdo do diagrama, e bibliotecário e sistema de notificação sendo atores secundários, tendo sua representação do lado direito.

Esses atores interagirão com o sistema no que chamaremos de casos de uso. Os casos de uso representam toda a ação que um ator tem para chegar em um determinado resultado observável. Cada caso de uso normalmente se inicia com um verbo e descreve a ação que o ator executa no sistema (IBM, 2021).

Na figura também pode-se observar que diferentes casos de uso se conectam. Essas conexões são chamadas de associações, sendo as representadas na figura as de inclusão e de extensão. Um relacionamento de inclusão, representado por uma seta aberta com linha tracejada indo do caso de uso principal para o caso de uso incluído, se dá quando o acionamento do caso de uso que está sendo incluído é obrigatório ao acionamento do caso de uso principal (base) (GUEDES, 2018).

Já o caso de uso de extensão, representado por uma seta aberta com linha tracejada indo do caso de uso estendido para o caso de uso principal, não tem a obrigatoriedade de ser executado quando o caso de uso principal é acionado, tendo sua execução atribuída a validações, consistências ou condições, esses podendo ou não ser acompanhadas de restrições em associações, que serão representadas em formato de folha com orelha dobrada contendo uma descrição entre chaves (GUEDES, 2018).

No exemplo, o usuário da biblioteca se conecta com o caso de uso “solicitar empréstimo”, que tem uma relação de inclusão com o caso de uso “verificar a disponibilidade do livro”. Isso significa que sempre que o usuário da biblioteca solicitar empréstimo, obrigatoriamente o bibliotecário terá que verificar a disponibilidade do livro.

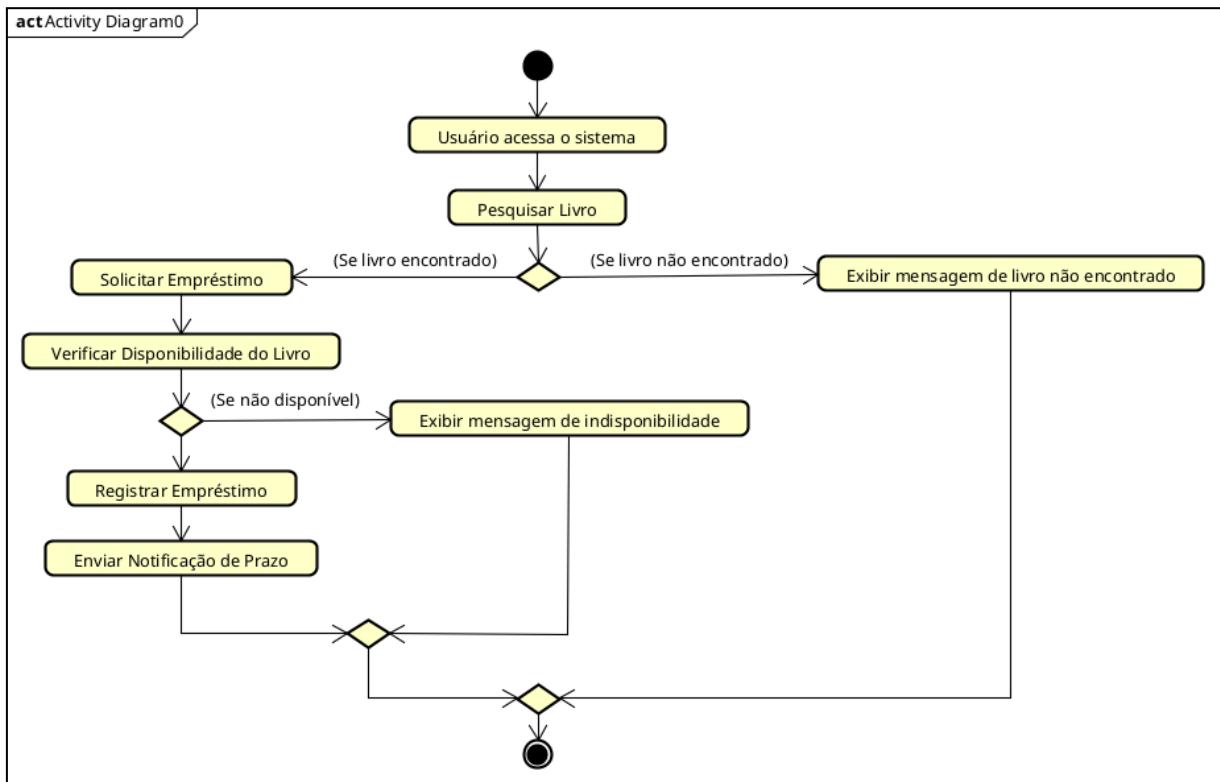
Já um exemplo de extensão se dá na figura quando o usuário da biblioteca aciona o caso de uso “devolver livro”, o caso de uso “cobrar multa” é estendido, mas somente se houver atraso, essa condição é descrita em uma restrição.

2.4 Diagrama de atividade

O diagrama de atividade, como o próprio nome já insinua, desbrincha as atividades, ou seja, as ações do usuário no sistema detalhadamente (FOWLER, 2000). Pode-se entender o diagrama de atividade como uma extensão dos principais casos de uso do projeto.

A figura 2 representa o mesmo sistema de biblioteca que vínhamos abordando. Nela podemos observar o detalhamento das principais interações que o usuário terá com o sistema.

Figura 2 — Diagrama de atividade empréstimo de livro



Fonte: Do próprio autor, 2025

O diagrama de atividade tem seu início no nó inicial, representado pelo símbolo de um círculo preto. Ele representa o início da interação do usuário, quando uma atividade é invocada (GUEDES, 2018). Esse nó inicial será ligado a uma ação, que nada mais é do que uma unidade funcional, representada por um retângulo com a ação descrita, identificando uma ação do usuário ou do sistema (IBM, 2025).

Já o símbolo do losango representa uma decisão ou ramificação que pode transformar uma única ação em mais de um estado paralelo (MICROSOFT, 2025).

O diagrama de atividade será finalizado com o nome de final de atividade representado por um círculo preenchido em um círculo vazio, todo diagrama deve culminar nele (GUEDES, 2018).

Na figura, o usuário acessa o sistema e em seguida pesquisa o livro, essa ação é ramificada em solicitar empréstimo, caso o livro seja encontrado, ou exibir mensagem de livro não encontrado. Ao solicitar empréstimo, o sistema terá que

verificar a disponibilidade do livro, prosseguindo para registrar o empréstimo do livro e enviar a notificação de prazo, caso o livro não estiver disponível o sistema exibirá a mensagem de indisponibilidade.

2.5 Diagrama de sequência

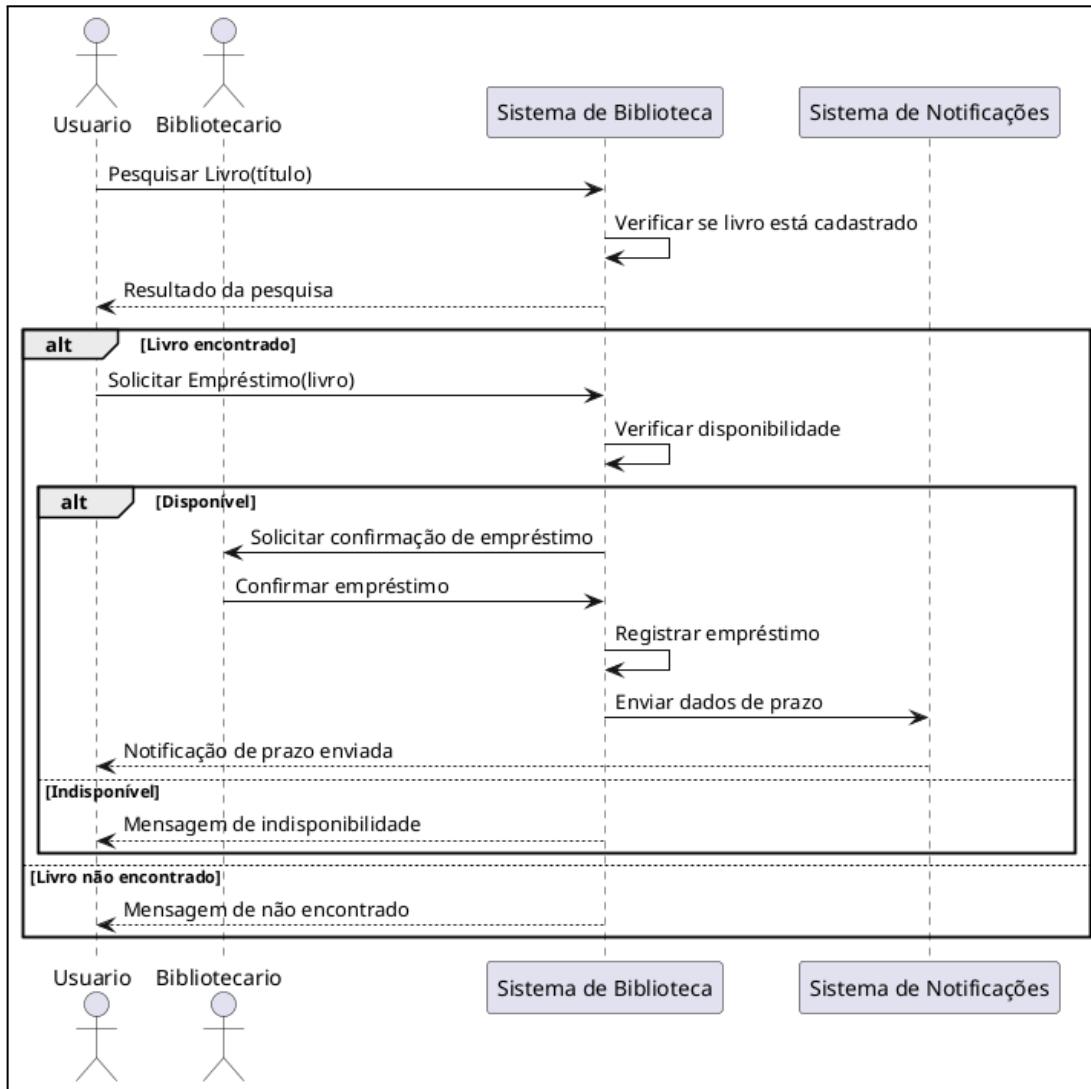
Uma maneira de representar as interações dos objetos que compõem o sistema de forma detalhada e cronológica é utilizando os diagramas de sequência. As interações são representadas por mensagens trocadas entre os objetos (MICROSOFT, 2025).

Todo o período da interação é atrelada a linhas de vida, representadas visualmente por linhas verticais tracejadas saindo do objeto. Dessas linhas de vida partem as mensagens trocadas entre os objetos, representadas por uma linha sólida com ponta preenchida. Mensagens de retorno, que respondem a uma mensagem anterior, são representadas por uma linha tracejada que retorna para a linha de vida a qual deverá receber a resposta, normalmente da direita para esquerda (GUEDES, 2018).

Para representar condições, o diagrama de sequência utiliza os fragmentos combinados, representados por um quadrado que engloba todo o período da condição. Anotações comuns nesses fragmentos combinados são os de alternatividade (alt) e de loop, que são para casos em que o diagrama se ramifica em duas direções condicionais, ou repete o ciclo descrito, respectivamente (IBM, 2021).

A figura 3 mostra um diagrama de sequência baseado no exemplo de biblioteca em que víhamos trabalhando.

Figura 3 — Diagrama de sequência empréstimo de livro



Fonte: Do próprio autor, 2025

O diagrama de sequência apresentado possui dois atores, sendo eles usuário e bibliotecário. Outros objetos também representados são de sistema de biblioteca e sistema de notificação.

A interação se inicia quando o usuário pesquisa livro, como parâmetro é passado o título do livro. Esses parâmetros, segundo Guedes (2018), são informações compartilhadas entre objetos, e devem ser utilizados para melhorar o entendimento do diagrama, podendo ser omitidos caso o autor entenda que beneficiará a compreensão.

A interação do diagrama prossegue com o sistema da biblioteca verificando se o livro está cadastrado e respondendo para o usuário o resultado da pesquisa. O usuário então, em uma condicional em que o livro foi encontrado, solicita o empréstimo do livro, ação essa prosseguida pela verificação de disponibilidade pelo sistema da biblioteca. Caso o livro esteja disponível o sistema da biblioteca solicitar a configuração do empréstimo para bibliotecário, que confirmará fazendo com que o sistema de biblioteca registre o empréstimo e envie os dados de prazo para o sistema de notificação, que por sua vez enviará essa informação para o usuário, caso o livro esteja indisponível o sistema de notificação retornará a mensagem de indisponibilidade do livro. O sistema da biblioteca informará o usuário caso o livro não seja encontrado na pesquisa inicial.

2.6 Diagrama de classe

Por último, explicaremos o diagrama de classes. Esse diagrama pode ser visto como o diagrama principal na construção de projetos orientados ao objeto.

Optamos em nosso projeto por utilizar a arquitetura de Modelo, Visão e Controlador, ou Model-View-Controller (MVC), uma arquitetura de software amigável às mudanças de paradigmas e de requisitos ao longo do projeto. Feita para funcionar em três camadas, o MVC lida com as interações primeiro recebendo-as na camada de controle (Controller), fazendo o processamento necessário na camada de modelo (Model), e por fim exibindo os elementos gráficos necessários para o usuário na camada de visualização (View) (LUCIANO; ALVES, 2011).

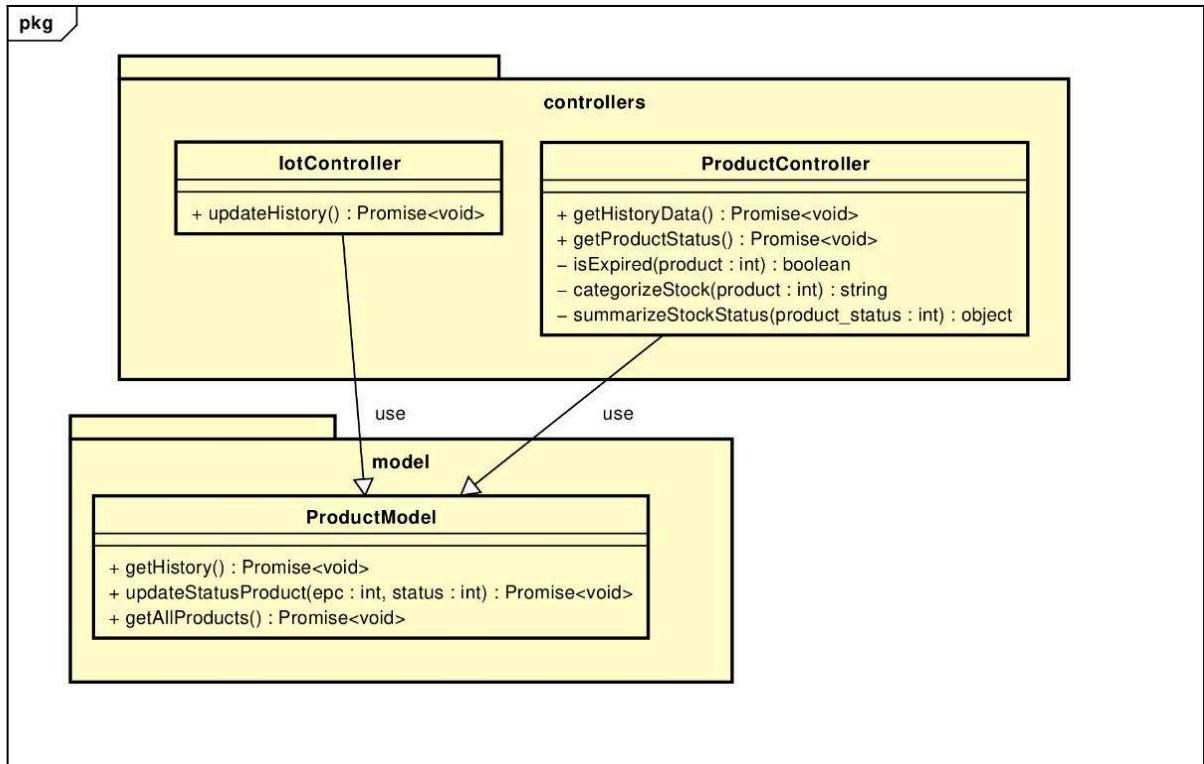
Embora a arquitetura do nosso sistema não seja orientada ao objeto, a representação gráfica com um diagrama de classes continua cabível ao projeto devido a sua versatilidade. É comum vermos o diagrama de classe sendo utilizado em mais de uma etapa na criação de um projeto. Como iniciamos o entendimento dos requisitos com o Diagrama de Casos de Uso, detalharei uma abordagem mais orientada ao modelo de domínio com Diagrama de Classe, ou seja, focando na solução do problema incluindo os detalhes de métodos e naveabilidade do nosso projeto.

Os principais componentes de um diagrama de classe são as classes, aqui lidas como as entidades que estamos identificando; os atributos dessa classe, que armazenam dados diversos; a visibilidade, que indica como essas diferentes classes

podem interagir entre si; e os métodos, sendo as funções que essas classes podem executar.

Na figura podemos observar o diagrama de classe referente ao projeto em questão, o Estok. Nele podemos observar como as classes vão interagir já no protótipo final.

Figura 4 — Diagrama de classe Estok



Fonte: Do próprio autor, 2025

O diagrama apresenta os pacotes Controllers, que reúne as classes IoTController e ProductController, e Model, que contém a classe ProductModel.

A comunicação ocorre em direção ao modelo, pois ambos os controladores dependem de seus serviços. O IoTController aciona diretamente o ProductModel para consultar ou atualizar dados de histórico por meio de updateHistory(). Já o ProductController utiliza o modelo ao executar as funções getHistoryData() e getProductStatus(), recebendo do ProductModel as informações necessárias para aplicar suas lógicas internas, como verificação de validade e categorização.

Essa relação de uso reforça a separação de responsabilidades, onde o pacote Controllers coordena regras de negócio enquanto o pacote Model concentra o acesso e manipulação dos dados.

2.7 Internet Das Coisas

O Estok interagirá tanto com o meio digital quanto com o meio físico, identificando e registrando interações com a prateleira do mercado autônomo, essa interação resultará na obtenção de dados, essa característica faz com que o projeto seja classificado como parte da Internet das Coisas, do inglês Internet of Things (IoT).

Segundo Magrani (2018), pode-se afirmar que dispositivos IoT possuem características de conectividade com a Internet e compartilhamento de informações entre si, possibilitando cenários de integração entre dispositivos e serviços.

Essas características, portanto, possibilitam a criação de dispositivos voltados para a melhoria de processos cotidianos, conectando estes ao contexto geral da internet.

Todas essas informações acumuladas com as leituras geram dados que, além de gerarem informações relevantes aos gerentes dos mercados autônomos, geram também dados que podem ser utilizados para estudar padrões de consumo específicos, contribuindo para um conceito conhecido como Big Data, que constitui esse grande volume de informações que produtos como o Estok podem produzir (Magrani, 2018).

Produtos como esse fazem parte da Indústria 4.0, sendo essa a grande nova transformação digital, tendo potencial de movimentar grandes alterações tanto em setores logísticos como industriais. Esses produtos são baseados na integração entre dispositivos físicos e a nuvem, onde os dados são armazenados e analisados para mapear padrões de uso (BOMJORNO; HERINGER; MADUREIRA, 2025).

2.8 Identificação por rádio frequência

Os padrões que poderão ser analisados são os de entrada e saída, que no nosso sistema será possibilitado por meio da tecnologia de identificação por radiofrequência (RFID).

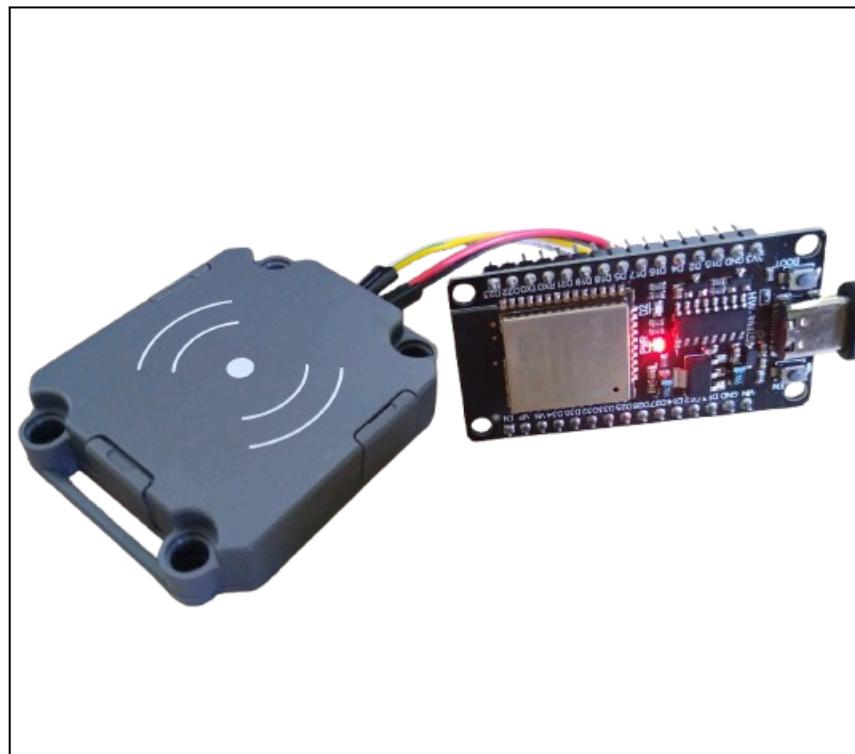
Conforme os estudos de Ferreira (2021), a etiqueta RFID possui uma antena que armazena um código único de identificação, tornando cada tag única. Essa etiqueta contém as informações necessárias para identificação do produto,

possibilitando toda a dinâmica de reconhecimento exato do produto que entrou na prateleira (COSTA, 2018).

Conforme analisa Campos (2021), a etiqueta RFID passiva não possui alimentação própria e, por conta disso, a emissão de dados e a própria alimentação da etiqueta vêm das ondas magnéticas emitidas pela antena do leitor.

Como abordado por Ferreira (2017), as etiquetas RFID possuem um circuito integrado de memória digital, que além de armazenar o código de identificação único, podem armazenar outros caracteres. No nosso sistema essa antena fica localizada nas proximidades da prateleira, identificando as interações da entrada e saída de produtos. Na Figura 5 pode-se observar o leitor que utilizamos em nosso projeto, ao seu lado se encontra a placa microcontroladora que detalharemos a seguir.

Figura 5 — Leitor RFID e Microcontrolador ESP32



Fonte: Do próprio autor, 2025

De acordo com Gonçalves (2025), ocorre a emissão de ondas de radiofrequência por meio da antena do leitor RFID. Essas ondas uma vez em contato com a etiqueta energizam a mesma fazendo com que ela envie o sinal e as suas informações de volta ao leitor.

Conforme Costa (2018), o leitor recebe os dados quando as etiquetas entram da área de alcance do mesmo. Devido a essa característica, a posição do leitor deverá ser analisada para obtermos a melhor leitura possível dentro do espaço delimitado da prateleira.

A tecnologia de identificação por radiofrequência foi escolhida pela sua eficácia comprovada no caso de uso das lojas Renner. Nessa implementação, a ruptura de estoque caiu 87%. Já a acurácia dos estoques aumentou em 64% (SENSORMATIC, 2025). Números como esse ressaltam a importância da implementação de tecnologias que promovam o aumento da rastreabilidade nos estoques, comprovando também a eficácia da tecnologia.

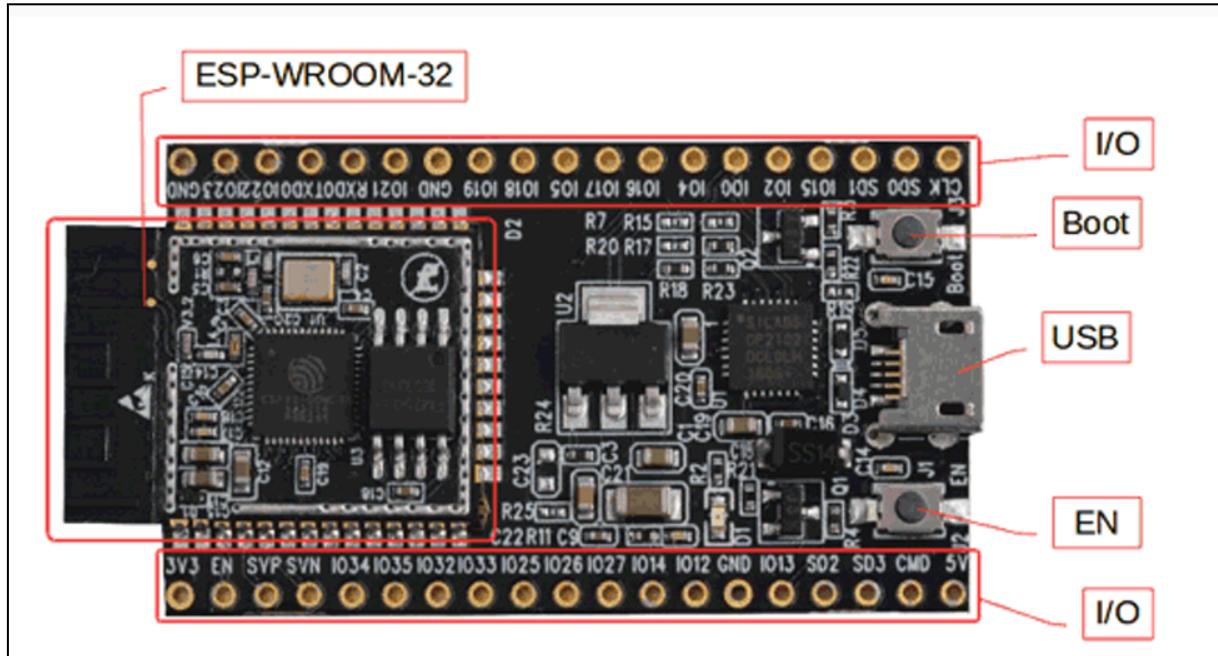
2.9 ESP32

Após as leituras das informações da etiqueta serem feitas pelo leitor RFID, o mesmo precisará comunicar essas informações para um microcontrolador para que possamos tratá-las e enviá-las para a interface que será vista pelo gerente do mercado autônomo.

A comunicação entre o leitor e o sistema será feita pela placa ESP32-WROOM-32, sendo esse um microcontrolador desenvolvido pela empresa chinesa Espressif Systems, capaz de executar o processamento de dados e se comunicar com outros meios via Wi-Fi ou Bluetooth (PET, 2021). Essa característica é importante, pois possibilita maior escalabilidade na infraestrutura de leitores.

A figura 6 mostra o ESP32 utilizado visto de cima, onde é possível visualizar seus componentes de conexão.

Figura 6 — Placa ESP32 com anotações de componentes



Fonte: PET. Introdução à Programação Embarcada. Minas Gerais, 2021

2.10 Arduino IDE

Para que possamos passar as instruções de como deve ser feito esse primeiro processamento dos dados lidos pelo leitor RFID para o microcontrolador ESP32, utilizaremos um Ambiente de Desenvolvimento Integrado (IDE). Segundo Carvalho (2023), o ESP32 é compatível com o Arduino IDE, ambiente primariamente utilizado para placas da família Arduino, porém tendo compatibilidade com a placa ESP32.

Essa IDE foi escolhida devido à sua interface clara e de bom entendimento, e a compatibilidade com extensões e bibliotecas que facilitaram o andamento do projeto.

2.11 C++

As instruções passadas para o ESP32 pelo Arduino IDE são feitas na linguagem de programação C++. Conforme a documentação oficial do C++ (2023), a linguagem desde o seu início se destaca fortemente pela sua capacidade de ser

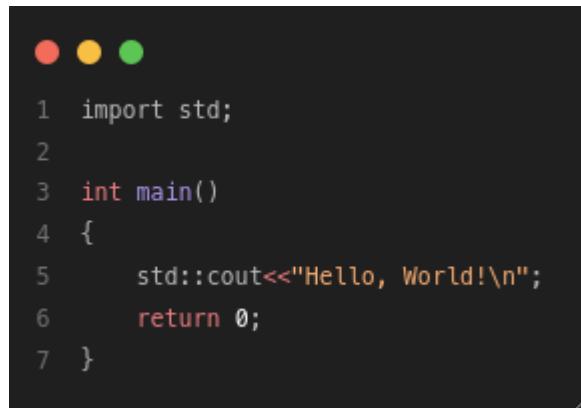
eficiente e poder atuar em diversos meios, como aplicativos de jogos, drivers e diversos programas.

A escolha de C++ com ESP32 se deve à sua alta velocidade de processamento, utilizando pouca memória. Uma forte qualidade da linguagem quando usada em dispositivos IoT.

Como Wiener e Pinson (1991) afirmam, C++, por ser orientada a objeto, é uma linguagem de programação extremamente eficaz quando se pensa em soluções mais humanizadas, que facilitam o entendimento do código. A linguagem também tem um melhor acesso ao hardware, parte física, possibilitando a criação de melhores softwares, parte lógica.

Segundo Stroustrup (2025), o código feito em C++ é bem popular pela sua alta compatibilidade entre diferentes máquinas. No exemplo a seguir, demonstrado na figura 7, podemos observar um código em C++ que exibe a mensagem “Hello world” no terminal.

Figura 7 — Código que exibe um texto em C++



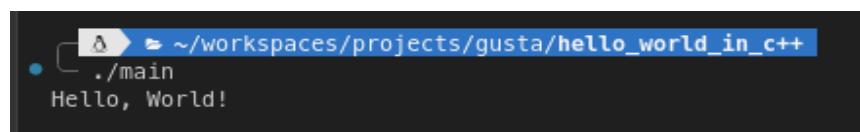
```

1 import std;
2
3 int main()
4 {
5     std::cout<<"Hello, World!\n";
6     return 0;
7 }
```

Fonte: Do próprio autor, 2025

O detalhamento dos comandos utilizados na figura 7 estão detalhados no Anexo A. A figura 8 ilustra a mensagem que o programa retorna após a sua execução:

Figura 8 — Resultado do código que exibe um texto em C++



```

~/workspaces/projects/gusta/hello_world_in_c++
./main
Hello, World!
```

Fonte: Do próprio autor, 2025

2.12 HTML

A interação com o usuário em nosso sistema RFID, é constituída pela dashboard, ferramenta visual que apresenta informações de forma concisa e organizada,. A principal base para a estrutura de código que utilizamos é a tecnologia de Linguagem de Marcação de Hipertexto.

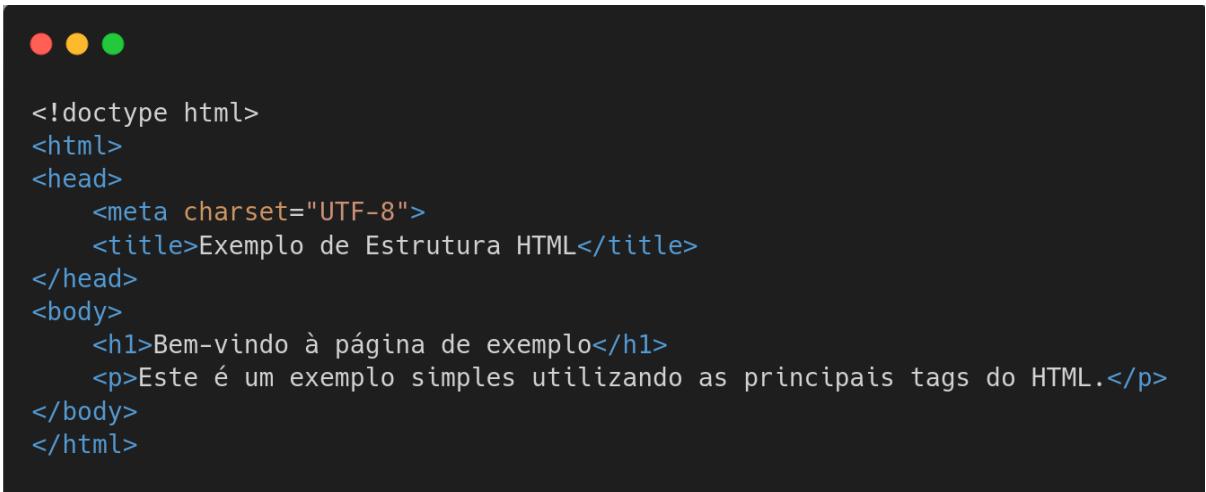
Segundo Silveira e Prates (2001), a Linguagem de Marcação de Hipertexto, do inglês Hypertext Markup Language (HTML) é a tecnologia para criação da estrutura de WebSites. A navegação de forma dinâmica é possível por conta do HyperText que por links conecta as páginas entre si, como explica Duckett (2016). Além disso, também destaca que Websites são compostos por diversos conteúdos como textos, links, imagens ou vídeos.

A partir disso, Cardoso (1999) afirma que para acessar aplicações web pelo navegador basta digitar o endereço (URL) na barra de pesquisa. Na sequência, Duckett (2016) ressalta que ao acessar um site, o navegador tem a função de traduzir o código HTML e projetar de forma visual para o usuário.

Todo esse processo só é possível graças aos elementos HTML, cuja construção se baseia nas *tags* (etiquetas), nomes colocados entre colchetes angulares (< >), indicando o início e o fim do elemento. Eles são fundamentais na criação de telas, onde cada elemento desempenha uma função específica na estrutura e organização dos conteúdos na página.

Segundo Silveira e Prates (2001), as principais tags usadas para construção da estrutura de um documento HTML básico são as seguintes apresentadas na figura 9.

Figura 9 — Estrutura básica de uma página em HTML



The image shows a terminal window with three colored window control buttons (red, yellow, green) at the top. The main area contains the following HTML code:

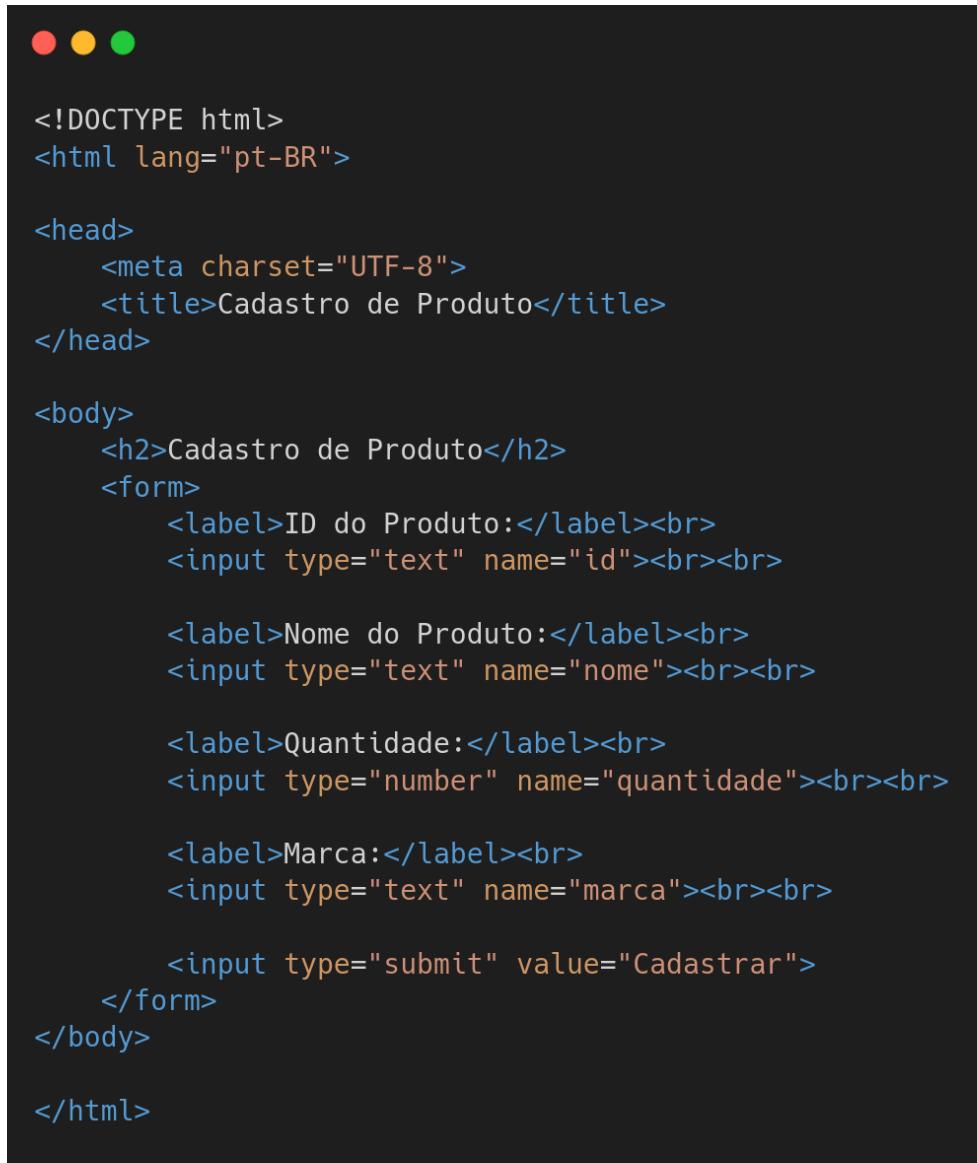
```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Exemplo de Estrutura HTML</title>
</head>
<body>
    <h1>Bem-vindo à página de exemplo</h1>
    <p>Este é um exemplo simples utilizando as principais tags do HTML.</p>
</body>
</html>
```

Fonte: Do próprio autor, 2025

O detalhamento dos comandos utilizados na figura 9 estão detalhados no Anexo B.

A variedade de tags dentro do HTML é vasta, abordaremos a seguir na figura 10 algumas tags exemplificadas na criação de um formulário de cadastro de produto.

Figura 10 — Exemplo de uma tela HTML para cadastro de produtos.



```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="UTF-8">
    <title>Cadastro de Produto</title>
</head>

<body>
    <h2>Cadastro de Produto</h2>
    <form>
        <label>ID do Produto:</label><br>
        <input type="text" name="id"><br><br>

        <label>Nome do Produto:</label><br>
        <input type="text" name="nome"><br><br>

        <label>Quantidade:</label><br>
        <input type="number" name="quantidade"><br><br>

        <label>Marca:</label><br>
        <input type="text" name="marca"><br><br>

        <input type="submit" value="Cadastrar">
    </form>
</body>

</html>
```

Fonte: Do próprio autor, 2025

O código mostrado na figura 10, resulta na renderização visual feita pelo navegador apresentada na figura 11.

Figura 11 — Resultado do código HTML de uma página de cadastro de produtos.

The screenshot shows a web page with a light gray background and a white central content area. At the top center, the title 'Cadastro de Produto' is displayed in a bold, dark font. Below the title, there are four input fields, each with a label to its left. The first field is labeled 'ID do Produto:' and contains a blank text input box. The second field is labeled 'Nome do Produto:' and also contains a blank text input box. The third field is labeled 'Quantidade:' and contains a text input box with a small up/down arrow icon to its right, indicating it's a numeric input field. The fourth field is labeled 'Marca:' and contains a blank text input box. At the bottom left of the form area, there is a single button labeled 'Cadastrar'.

Fonte: Do próprio autor, 2025

O detalhamento das tags utilizadas na criação da tela representada na figura 11 estão detalhados no Anexo C.

Com a estrutura da página construída, é possível visualizar os elementos organizados, porém somente o HTML não garante a aparência da tela do nosso sistema. Por isso, se torna necessário uma folha de estilo (CSS), conforme abordaremos no próximo tópico.

2.13 CSS

Para complementar o visual dos sites, é utilizado a folhas de estilo em cascata. Como destaca Knight (2018) a folhas de estilo em cascata, do inglês Cascading Style Sheets (CSS) é uma tecnologia usada para estilizar páginas web. Duckett (2016) explica que é possível criar regras que informam como determinado elemento se comporta em uma página web, controlando atributos como cores, tamanhos, fontes e cor de fundo da página.

O CSS é um dos principais fatores para uma experiência satisfatória ao usar uma página web, com o intuito de que seja consumida da melhor forma possível conforme apresentado por Eis (2012).

Como abordado por Duckett (2016), de início, para estilizar uma tela HTML, é necessário realizar a conexão entre os arquivos, HTML com o arquivo CSS. O intermediário desta ligação é a tag de ligação, quando inserida no código HTML referenciando a folha de estilo, o HTML e o CSS conseguem trabalhar em conjunto. O código a seguir na figura 12 mostra a conexão entre os dois arquivos.

Figura 12 — Exemplo de conexão entre HTML e CSS.

A screenshot of a terminal window with a black background and white text. At the top, there are three colored window control buttons: red, yellow, and green. Below them, the terminal displays the following HTML code:

```
<head>
  <title>Document</title>
  <link rel="stylesheet" href="style.css">
<head>
```

Fonte: Do próprio autor, 2025

O Detalhamento dos papéis de atributo na conexão estão no Anexo D. Ao especificar o tipo do arquivo e o diretório do mesmo, o navegador é capaz de atribuir os estilos declarados às marcações.

Além disso, Duckett (2016) ressalta que CSS é a junção de duas partes, uma se chama seletor e a outra declaração. Para um elemento HTML ser estilizado, são utilizados os seletores, que apontam qual tag estará recebendo as regras de estilo.

A figura 13 faz parte da estilização do site e apresenta como escrever regras na folha de estilo.

Figura 13 — Exemplo de escrita na CSS.



```
body {  
    font-family: sans-serif;  
    background: #f2f2f2;  
    padding: 30px;  
}  
  
form {  
    background: #fff;  
    padding: 20px;  
    max-width: 350px;  
    margin: auto;  
    border-radius: 6px;  
}
```

Fonte: Do próprio autor, 2025

A explicação dos seletores e das declarações se encontram no Anexo E.

A figura 14 exibe a solução completa para estilização do formulário de cadastro de produtos da nossa dashboard.

Figura 14 — CSS da tela de cadastro de produtos

```
● ○ ●

body {
    font-family: sans-serif;
    background: #121212;
    padding: 30px;
}

form {
    background: #ffffff;
    padding: 20px;
    margin: 0 auto;
    width: 100%;
    max-width: 350px;
    border-radius: 8px;
}

label {
    display: block;
    margin: 10px 0 5px 0;
}

input[type="text"],
input[type="number"] {
    width: 90%;
    padding: 8px;
    margin-bottom: 10px;
    border-radius: 4px;
    border: 1px solid #ccc;
}

input[type="submit"] {
    width: 100%;
    padding: 10px;
    background: #007bff;
    color: white;
    border-radius: 4px;
    border: none;
    cursor: pointer;
}
```

Fonte: Do próprio autor, 2025

Somado a isso, com a união de todas as tecnologias apresentadas, o resultado da página de cadastro de produtos representado na figura 15.

Figura 15 — Resultado da tela de cadastro HTML e CSS em conjunto.

The image shows a screenshot of a web page with a light gray background. In the center, there is a white rectangular form with rounded corners. At the top left of the form, the text "Informações do Produto" is displayed. Below this, there are four input fields, each preceded by a label: "ID do Produto:", "Nome do Produto:", "Quantidade:", and "Marca:". Each label is followed by a white input field with a thin gray border. At the bottom of the form is a blue rectangular button with the white text "Cadastrar Produto".

Fonte: Do próprio autor, 2025

2.14 JavaScript

Conforme explica Silva (2010), com a criação de HTML, responsável pela estrutura dos websites, e o CSS, com o papel de estilizar a página para melhor experiência do usuário final, houve a necessidade de uma tecnologia com a função de adicionar interações para tais páginas.

Na visão de Lepsen (2018), o JavaScript é a ferramenta encarregada pela criação das interações da aplicação web com o consumidor final. É a tecnologia capaz de fornecer funcionalidades aos elementos web, por meio dos campos de formulários, configurações gerais de uma página e interações para salvar nossos dados nos navegadores, o JavaScript se comunica com o visitante deixando a página com ar mais dinâmico.

A dinamicidade do JavaScript proporciona ao usuário uma experiência mais agradável durante sua navegação pelo website. Atrelado ao painel (dashboard), essa característica pode gerar uma sensação ainda mais marcante. Segundo Groner (2019), com essa ferramenta, é possível agir tanto na parte visual (Front-end) aplicando animações e manipulando elementos, quanto na parte lógica (Back-end) criando funcionalidades, que podem ser chamadas de scripts.

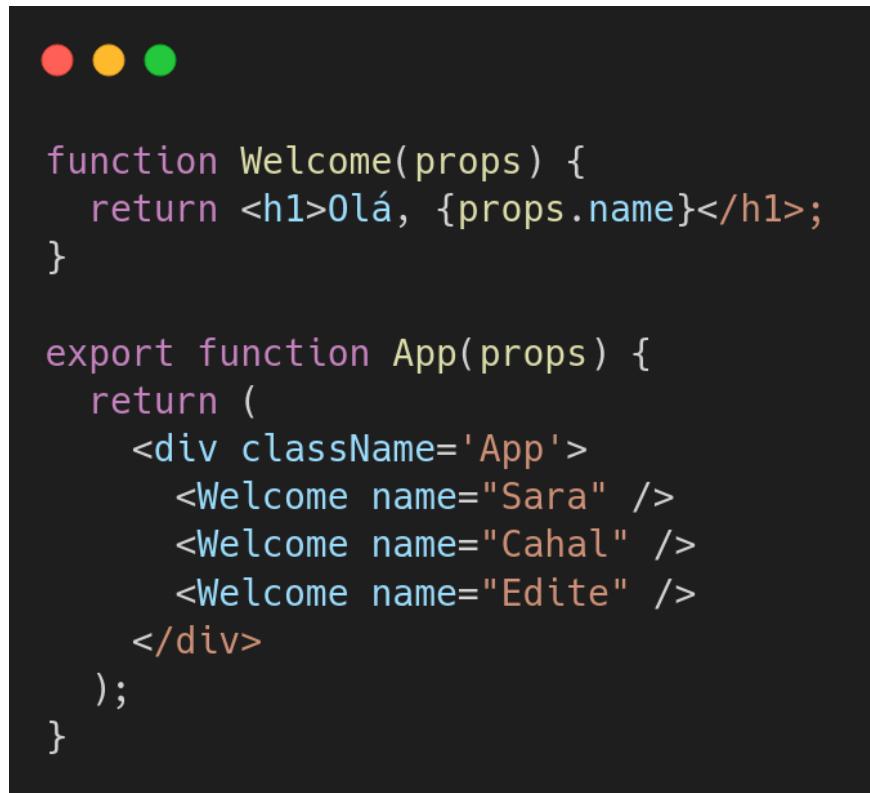
JavaScript é utilizado principalmente para rodar scripts no lado do cliente, os responsáveis pela interpretação são os navegadores, que entendem e executam as funcionalidades.

2.15 React

Com essas tecnologias apresentadas já poderíamos iniciar o desenvolvimento do nosso site, porém escrever um site utilizando puramente HTML, CSS e JavaScript faz com que a manutenção posterior e a escalabilidade (capacidade de um site/sistema de crescer de acordo com novas demandas) fiquem comprometidas. Para contornar problemas como esses, optamos pela utilização do React, um framework de criação web.

O React foi criado por Jordan Walke, quando atuava como engenheiro de software do Facebook. O React possibilita a criação de componentes, que são blocos de código (META, [s.d.]). A figura 16 mostra a estrutura de um componente React.

Figura 16 — Componente React



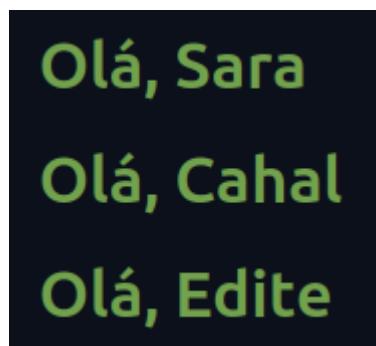
```
function Welcome(props) {
    return <h1>Olá, {props.name}</h1>;
}

export function App(props) {
    return (
        <div className='App'>
            <Welcome name="Sara" />
            <Welcome name="Cahal" />
            <Welcome name="Edite" />
        </div>
    );
}
```

Fonte: Meta, [s.d.]

A explicação do código presente na figura 16 se encontra no Anexo F. A figura 17 mostra o resultado da renderização do componente.

Figura 17 — Renderização componente React



Fonte: Do próprio autor, 2025

O React usufrui de um tipo de arquivo capaz de misturar JavaScript e HTML, como é possível observar na figura 16; esse tipo de arquivo é chamado JSX. Essa característica deste framework é o que torna tão prático a criação de sites (WIERUCH; ROMERO, 2018).

Outra característica do React é de que as páginas com ele criadas são oferecidas dinamicamente, ou seja, a medida que o usuário interage com a página, novos componentes vão sendo carregados. Sites com essa característica recebem o nome de Aplicativo de Página Única, ou Single Page Application (SPA) (OLIVEIRA, 2017).

2.16 Node.js

Para que possamos integrar a parte física que se comunica com a internet e disponibilizar essas informações na nossa interface web, precisaremos de um ambiente que nos proporcione essa capacidade de forma escalável e eficiente. Escolhemos por fim o Node.js, por se tratar de um ambiente que utiliza uma linguagem familiar, o JavaScript.

De acordo com Pereira (2014), Node.js, criado em 2009 por Ryan Dahl e com ajuda inicial de 14 colaboradores, se destaca especialmente em aplicações que possuem muitas entradas e saídas de dados (Input/Output - I/O). Nesse cenário, ele consegue usufruir o máximo do poder de processamento dos servidores de forma produtiva, sem interromper o funcionamento da aplicação enquanto processa esses dados, como ocorria em outros ambientes.

Em relação ao projeto atual, é necessário devido ao alto fluxo de dados recebidos pelos dispositivos IoT. A sua característica de não bloquear o funcionamento da aplicação enquanto processa os dados é fundamental no nosso projeto.

Segundo Moraes (2023), Node.js utiliza o motor V8 da Google e permite criar rotas web usando diversos protocolos de redes (regras de como dispositivos devem se comunicar entre si) como, como HTTPS, DNS, FTP etc. Permitindo assim a construção de sites e aplicativos para sistemas operacionais iOS e Android.

Para Powers (2017), Node.js é uma ferramenta versátil e completa, podendo ser usada em diversos cenários e situações, graças ao seu ambiente com diversas funcionalidades e um bom alcance.

O exemplo a seguir é de um código que junta o nome e o sobrenome do usuário, exibindo ao final o nome completo. Muito usado em sistemas em que é necessário exibir o nome e sobrenome do usuário juntos, representado na figura 18.

Figura 18 — Função juntar nome e sobrenome Node.js



```

● ● ●
1 function JuntarNomes(nome, sobrenome) {
2     return console.log(nome + " " + sobrenome);
3 }
4
5 JuntarNomes("Ananda", "Holanda");
6

```

Fonte: Do próprio autor, 2025

A explicação do código contido na figura 18 se encontra no Anexo G. A figura 19 ilustra o que é exibido no terminal após a execução do código.

Figura 19 — Resultado função Node.js



Ananda Holanda

Fonte: Do próprio autor, 2025

2.17 Banco de dados

Para guardar as informações provenientes do IoT recebidas por intermédio do Node.js, precisaremos utilizar um banco de dados.

O banco de dados consiste no armazenamento eletrônico de dados com foco no seu acesso e alteração de forma rápida. A linguagem que utilizaremos na criação do nosso banco de dados é a Linguagem de Consulta Estruturada, do inglês Structured Query Language (SQL) que atualmente figura como a principal linguagem para banco de dados relacionais (EEEP, [s.d.]).

Essa linguagem, como mencionado, tem como sua principal característica a forma relacional como organiza o banco de dados; sendo aqui relacional relativo à

criação de tabelas que interconectam os dados dentro dela (DATE, 2003). Na Figura 20 podemos observar a criação e a inserção de dados em uma tabela de banco de dados.

Figura 20 — Código SQL Criação e inserção de dados tabela de Produtos com consulta

```

CREATE TABLE Produtos (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    categoria VARCHAR(50),
    preco DECIMAL(10,2) NOT NULL,
    estoque INT NOT NULL
);

INSERT INTO Produtos (nome, categoria, preco, estoque)
VALUES
    ('Mouse Óptico', 'Periféricos', 59.90, 120),
    ('Teclado Mecânico', 'Periféricos', 249.90, 80),
    ('HD 1TB', 'Armazenamento', 329.90, 40);

SELECT nome, estoque
FROM Produtos
WHERE estoque < 100;

```

Fonte: Do próprio autor, 2025

O código apresentado cria uma tabela para armazenar produtos. São inseridos alguns valores nessa tabela de produtos e por último é consultado itens com estoque abaixo de um limite de cem. O resultado dessa consulta e as informações obtidas são mostradas na figura 21.

Figura 21 — Resultado consulta banco de dados tabela Produtos

	nome	estoque
	Teclado Mecânico	80
	HD 1TB	40

Fonte: Do próprio autor, 2025

Escolhemos esta tecnologia devido a sua funcionalidade de promover persistência dos dados e ser de fácil utilização em conjunto com as demais tecnologias utilizadas.

3 DESENVOLVIMENTO

Com a teoria já consolidada, agora, antes de iniciar o desenvolvimento do nosso projeto, precisamos primeiro entender os requisitos por completo, para que posteriormente possamos começar a implementação do hardware seguido da implementação do sistema web que exibirá as informações para o gerente do mercado autônomo.

3.1 Requisitos do Sistema

Para podermos ter uma melhor visualização do sistema, primeiro faremos a modelagem e a arquitetura do sistema da Estok. Começaremos detalhando o projeto conceitual para entender sua estrutura. O quadro 1 mostra os requisitos funcionais do nosso sistema, o estabelecimento destes são essenciais para entendermos as funcionalidades principais do nosso sistema.

Quadro 1 — Requisitos funcionais

Código	Descrição
RF01	Permitir o login dos usuários com autenticação segura.
RF02	Permitir o cadastro, edição e exclusão de usuários (restrito ao administrador).
RF03	Registrar automaticamente a chegada ou retirada de produtos via leitura RFID.
RF04	Atualizar o estoque com base nas leituras RFID.
RF05	Exibir no dashboard a quantidade atual de produtos por categoria.
RF06	Emitir alertas visuais no dashboard quando um produto estiver em falta.
RF07	Emitir alertas quando um produto estiver próximo da data de vencimento.
RF10	Enviar dados recebidos do ESP32 para o frontend por WebSocket.
RF12	Possibilitar a pesquisa e o filtro de produtos dentro do dashboard.

Fonte: Do próprio autor, 2025

As próximas informações a serem apresentadas no quadro 2 representam os requisitos não funcionais que, ao invés de definir as principais funcionalidades que o nosso sistema deverá atender, definem características do nosso sistema não necessariamente operacionais.

Quadro 2 — Requisitos não funcionais

Código	Descrição
RNF01	O sistema deverá ser responsivo e acessível via navegadores de desktop e dispositivos móveis.
RNF02	O backend deve garantir comunicação segura por meio de HTTPS.
RNF05	A comunicação entre ESP32 e backend deve utilizar protocolos HTTP, priorizando a simplicidade.
RNF06	A autenticação de usuários deve ser segura, com senhas criptografadas e tokens de sessão.
RNF07	O sistema deverá estar hospedado na nuvem.
RNF08	A interface deverá ser intuitiva, visando facilidade de uso para gerentes e repositores.

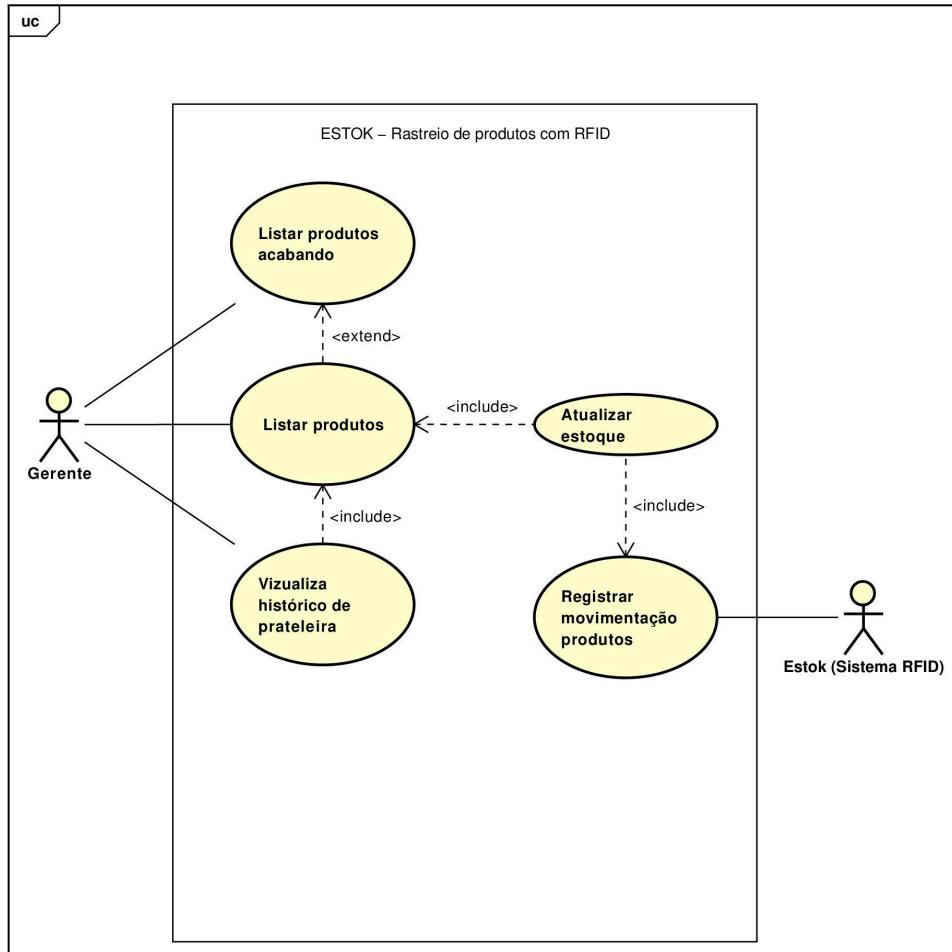
Fonte: Do próprio autor, 2025

3.2 Diagramação

Para ter uma visualização inicial do nosso projeto utilizaremos o diagrama de caso de uso que contempla os atores principais do nosso projeto e a maneira como eles interagirão com o nosso sistema.

A figura 22 mostra o diagrama de casos de uso modelado projetando as funcionalidades principais do nosso sistema.

Figura 22 — Diagrama casos de uso Estok



Fonte: Do próprio autor, 2025

O diagrama mostra que o gerente pode consultar o sistema para “listar produtos”, obtendo uma visão geral de todos os itens lidos. Essa funcionalidade serve como base para outras operações, já que oferece o panorama completo do estoque monitorado por RFID.

A partir dessa listagem, o caso de uso “listar produtos acabando” é ativado como extensão. Ele filtra automaticamente os itens cuja quantidade está abaixo do nível mínimo configurado, permitindo que o gerente identifique rapidamente riscos de ruptura e tome decisões de reposição. Trata-se de uma ampliação da listagem geral, acionada apenas quando o gerente deseja enxergar especificamente produtos críticos.

O gerente também pode “visualizar o histórico de prateleira”, função incluída dentro da listagem de produtos. Ela apresenta registros de movimentações, como entradas e saídas do item da prateleira. O histórico é construído a partir dos dados capturados pelo sistema RFID.

O sistema RFID, representado como ator Estok, é responsável por “registrar movimentação de produtos”, função interna que coleta automaticamente eventos de leitura das etiquetas. Esse registro alimenta o banco de dados em tempo real e garante rastreabilidade completa.

Com base nos registros provenientes do RFID, o sistema executa “atualizar estoque”, que ajusta as quantidades de cada item conforme as movimentações detectadas. Esse caso de uso é incluído como parte do processo de registro, pois depende diretamente dos dados capturados pelo leitor RFID.

O quadro 3 apresenta a documentação completa do caso de uso, descrevendo sua finalidade, os atores envolvidos, as condições necessárias para sua execução e os resultados esperados após sua realização. Esse quadro permite compreender claramente o objetivo da funcionalidade dentro do sistema Estok.

Quadro 3 — Documentação do Caso de Uso: UC01 Listar Produtos

Nome do Caso de Uso	UC01 – Listar Produtos
Caso de Uso Geral	—
Autor Principal	Gerente
Atores Secundários	Sistema Estok (RFID)
Resumo	Este caso de uso permite que o gerente visualize a lista de produtos cadastrados no sistema Estok, consultando informações de estoque e status de prateleira. Também serve como base para ações

	complementares, como listar produtos acabando e visualizar histórico de prateleira.
Pré-condições	O gerente deve estar autenticado no sistema. O sistema Estok deve estar sincronizado com os leitores RFID.
Pós-condições	A lista de produtos é exibida, podendo ser utilizada para atualização de estoque ou visualização de histórico.

O quadro 4 introduz o cenário principal do caso de uso, detalhando a interação passo a passo entre o gerente e o sistema. Ele mostra o fluxo normal de execução.

Quadro 4 — UC01 Cenário Principal

Ações do Ator	Ações do Sistema
1. O gerente acessa o módulo de controle de estoque.	2. O sistema exibe o menu de opções de estoque.
3. O gerente seleciona a opção “Listar produtos”.	4. O sistema consulta o banco de dados RFID.
5. —	5.1 O sistema inclui o caso de uso “Atualizar estoque” para garantir dados atualizados.
6. —	6.1 O sistema exibe a lista completa de produtos, quantidades e localização em prateleiras.

O quadro 5 apresenta o cenário alternativo “Listar Produtos Acabando”, que representa uma variação do fluxo principal voltada para situações em que o gerente deseja identificar apenas produtos com estoque baixo.

Quadro 5 — UC01 Cenário Alternativo Listar Produtos Acabando

Ações do Ator	Ações do Sistema
1. O gerente solicita a listagem de produtos com estoque baixo.	2. O sistema estende o caso de uso “Listar produtos” e exibe apenas os itens cujo estoque está abaixo do nível mínimo configurado.

O quadro 6 introduz o cenário alternativo “Visualizar Histórico de Prateleira”, que descreve o processo de consulta ao histórico de movimentações de um produto. Esse quadro destaca como o sistema utiliza dados RFID para fornecer rastreabilidade, expandindo o caso de uso principal com informações detalhadas.

Quadro 6 — UC01 Cenário Alternativo Visualizar Histórico de Prateleira

Ações do Ator	Ações do Sistema
1. O gerente solicita o histórico de movimentação de um produto.	2. O sistema inclui o caso de uso “Visualiza histórico de prateleira” e apresenta registros de movimentações anteriores capturados via RFID.

3.3 Dispositivo físico

Visando a possibilidade de usar as tags RFID nesse controle de estoque, primeiro pensamos em uma maneira de adequar e viabilizar esse sistema. Foi então que nos deparamos com os casos da implementação das lojas Renner, que nos salientou uma realidade do setor logístico que consiste na implementação das etiquetas de radiofrequência diretamente no produto final. Isso nos fez pensar em como o setor logístico, mais especificamente os mercados e mercados autônomos, se beneficiaram com a implementação dessa tecnologia nas prateleiras.

A escolha das etiquetas e leitores RFID se dá pela sua versatilidade. Se tratando de um sistema que não consome muita energia e que pode facilmente ser

escalado (aumentar sua capacidade), a tecnologia se torna ideal no cenário apresentado de mercados autônomos.

Para que a leitura dos produtos seja feita de forma eficaz, a disponibilidade do leitor deve ser feita em uma localização que siga as restrições de faixa de leitura do leitor. No protótipo que fabricamos, o leitor escolhido tem um alcance de até 3 metros, e será posicionado na parte superior da prateleira, conforme mostra a representação tridimensional na figura 23.

Figura 23 — Representação tridimensional Estok em prateleira



Fonte: Do próprio autor, 2025

Começamos, portanto, validando os componentes físicos do módulo escolhido. O leitor RFID de modelo YPD-4035 da marca Yanpodo foi testado por meio do software da própria fabricante para confirmar a leitura correta das tags, assegurando que o hardware funcionava. Posteriormente o ESP32 foi conectado ao leitor, estabelecendo uma comunicação.

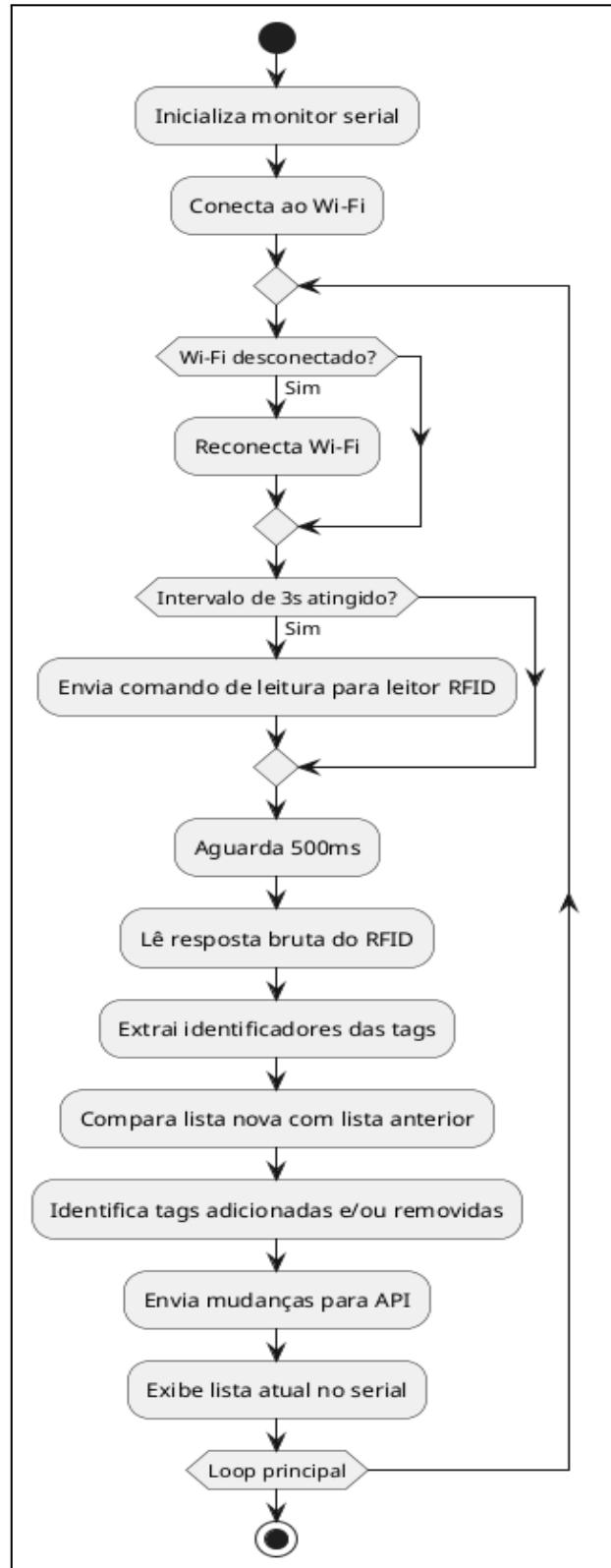
Após testado o funcionamento dos componentes, iniciou-se a programação. O código em C++ foi feito com base em um loop contínuo, responsável por enviar comandos seriais para ativar o modo de leitura, receber as informações das tags detectadas e armazená-las em uma lista temporária. A leitura é então comparada com a próxima leitura para identificar alterações no estoque. Sempre que mudanças são detectadas, enviamos os dados organizados em JSON via HTTP POST para as

rotas da API responsáveis pelo registro de entradas e saídas. Essa abordagem faz com que o envio seja otimizado, evitando erros e mantendo o backend atualizado.

Durante essa etapa, surgiram desafios para compreender o protocolo do leitor. Precisamos dos comandos hexadecimais fornecidos pelo fabricante e, depois de realizar testes, conseguimos incorporar corretamente o modo de leitura contínua. Para evitar envios desnecessários para a API, a lógica de comparação foi implementada. Também ocorreram leituras erradas e falsos positivos devido a interferências, esses problemas foram solucionados por meio de regulagem da potência e do tempo de leitura. Com esses ajustes, o sistema atingiu uma maior estabilidade.

O fluxograma representado na figura 24 demonstra o funcionamento do código C++ embarcado.

Figura 24 — Fluxograma código embarcado C++



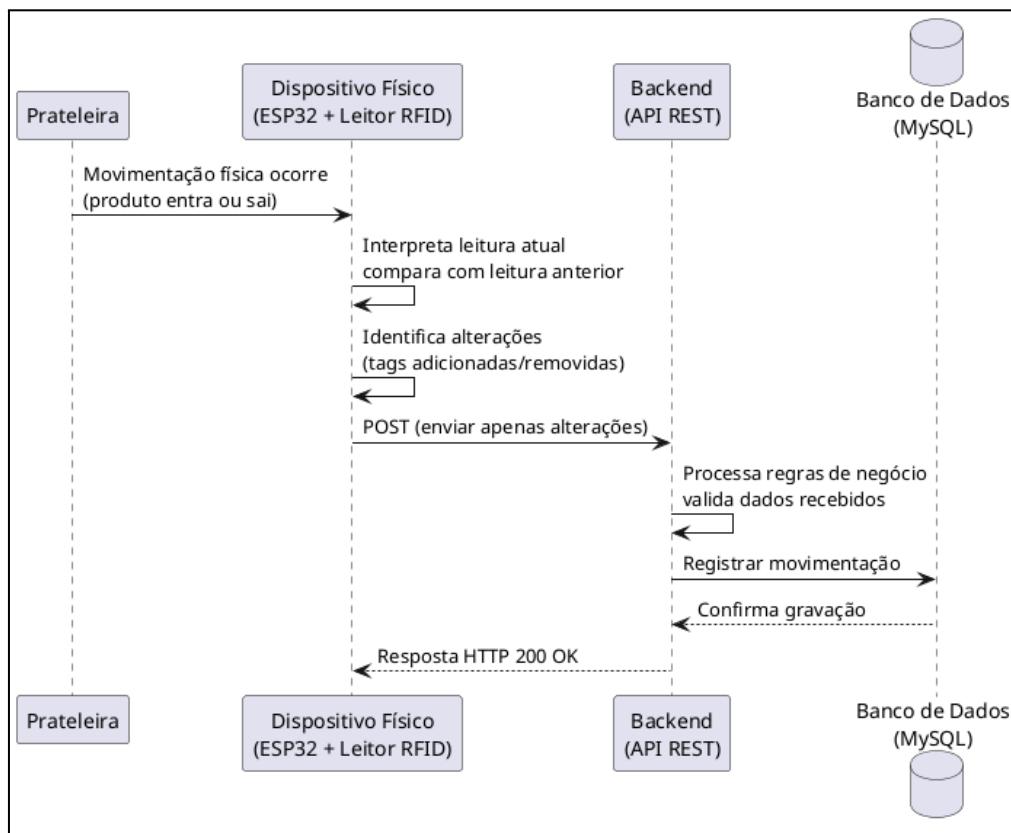
Fonte: Do próprio autor, 2025

3.4 Lógica de aplicação

Agora precisaremos passar os dados para o backend, que será responsável pelo processamento e persistência das informações. Essa parte, que denominaremos de lógica de aplicação, garante que cada detecção de movimentação no ambiente físico resulte em uma atualização na dashboard.

A implementação foi estruturada em três etapas. Primeiro a interpretação dos eventos gerados pelo dispositivo físico, passando depois pelo processamento interno das regras de negócio e por último registrando as informações no banco de dados. A partir do momento em que o dispositivo físico identifica alterações na prateleira, o dispositivo físico determina quais produtos entraram ou saíram. Apenas essas alterações são enviadas via HTTP para o backend, reduzindo tráfego desnecessário e garantindo que o servidor processe apenas os eventos relevantes. O diagrama de sequência contido na figura 25 exemplifica os processos nomeados.

Figura 25 — Diagrama sequência etapas da lógica de aplicação



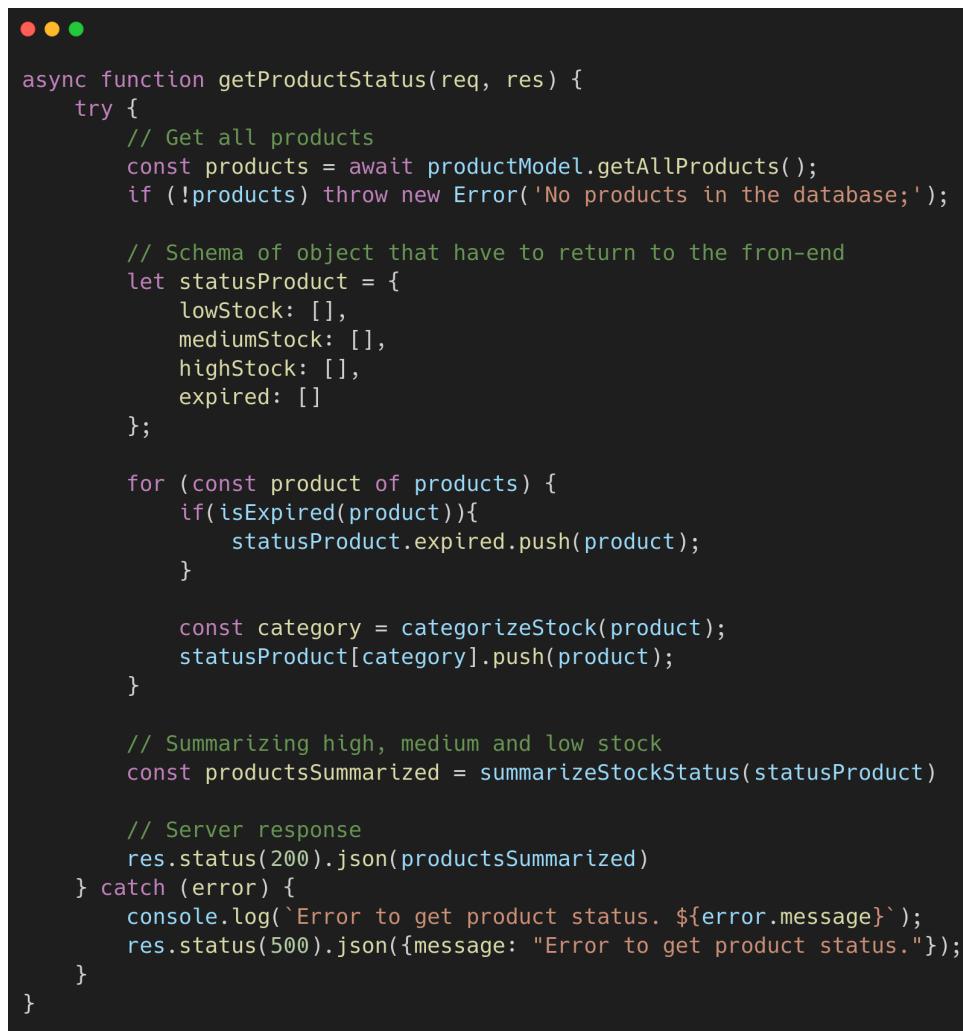
Fonte: Do próprio autor, 2025

Ao receber esses eventos, o backend aplica a lógica de aplicação. Cada requisição é validada e enviada para persistência no banco de dados. Para cada

movimentação o servidor registra a tag envolvida, o tipo de ação (entrada ou saída), a data e hora do evento e atualiza o histórico do produto. Esse processo garante que todas as movimentações fiquem armazenadas de forma cronológica, o que permite um acompanhamento detalhado do fluxo dos itens.

Além da atualização do histórico, a lógica de aplicação também é responsável por realizar consultas categorizadas por estado de produtos, ou seja, o backend executa consultas que retornam itens agrupados por status do estoque (validade ou quantidade atual). Essa organização permite que o frontend apresente uma visão clara do estoque, podendo organizar a visualização entre produtos vencidos, em baixa quantidade, em nível intermediário e adequados. Presente na figura 26 está o trecho da lógica de aplicação responsável por retornar os produtos por categoria de necessidade de reposição.

Figura 26 — Lógica de retorno por necessidade de reposição Node.js



```

● ● ●
async function getProductStatus(req, res) {
    try {
        // Get all products
        const products = await productModel.getAllProducts();
        if (!products) throw new Error('No products in the database');

        // Schema of object that have to return to the front-end
        let statusProduct = {
            lowStock: [],
            mediumStock: [],
            highStock: [],
            expired: []
        };

        for (const product of products) {
            if(isExpired(product)){
                statusProduct.expired.push(product);
            }

            const category = categorizeStock(product);
            statusProduct[category].push(product);
        }

        // Summarizing high, medium and low stock
        const productsSummarized = summarizeStockStatus(statusProduct)

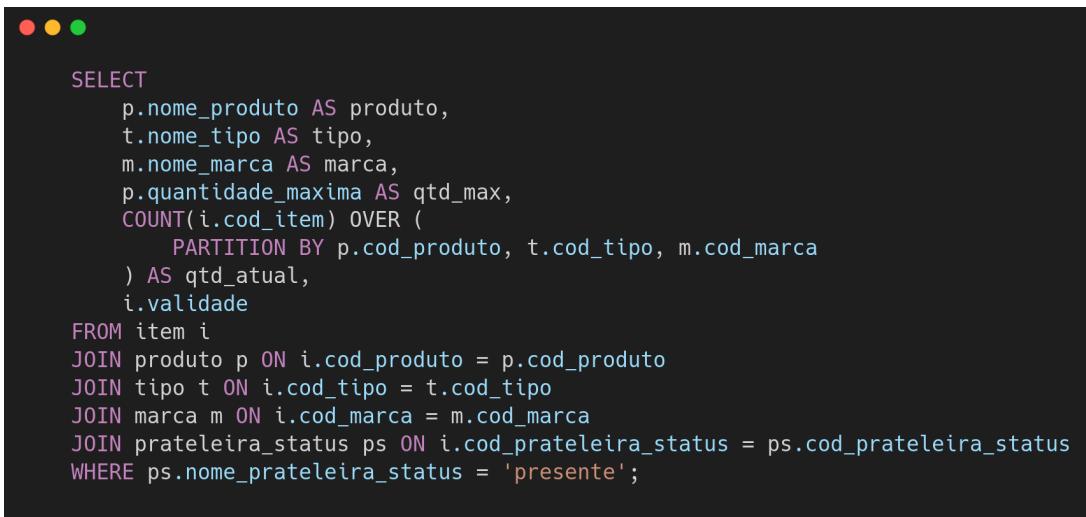
        // Server response
        res.status(200).json(productsSummarized)
    } catch (error) {
        console.log(`Error to get product status. ${error.message}`);
        res.status(500).json({message: "Error to get product status."});
    }
}

```

Fonte: Do próprio autor, 2025

Foi necessário também garantir a estabilidade devido a latência causada pela hospedagem remota do banco de dados. A solução adotada foi estabelecer consultas únicas de SQL, tornando-as mais eficientes ao reduzir o número de chamadas ao banco e diminuindo o tempo de resposta. Todos os fluxos sensíveis foram implementados de forma assíncrona para evitar bloqueios e manter a aplicação responsiva mesmo durante picos de requisições vindas do dispositivo físico. Presente na figura 27 está uma consulta otimizada utilizada na lógica de aplicação.

Figura 27 — Consulta de produtos otimizada



```

SELECT
    p.nome_produto AS produto,
    t.nome_tipo AS tipo,
    m.nome_marca AS marca,
    p.quantidade_maxima AS qtd_max,
    COUNT(i.cod_item) OVER (
        PARTITION BY p.cod_produto, t.cod_tipo, m.cod_marca
    ) AS qtd_atual,
    i.validade
FROM item i
JOIN produto p ON i.cod_produto = p.cod_produto
JOIN tipo t ON i.cod_tipo = t.cod_tipo
JOIN marca m ON i.cod_marca = m.cod_marca
JOIN prateleira_status ps ON i.cod_prateleira_status = ps.cod_prateleira_status
WHERE ps.nome_prateleira_status = 'presente';

```

Fonte: Do próprio autor, 2025

Foram realizados testes com simulações completas do dispositivo físico em conjunto com o leitor RFID, enviando informações de mudanças nas tags identificadas ao backend e armazenando-as no banco, validando todo o ciclo de comunicação. Esses testes confirmaram a confiabilidade da lógica, demonstrando que o sistema responde corretamente às mudanças físicas no ambiente e mantém o inventário digital atualizado em tempo real.

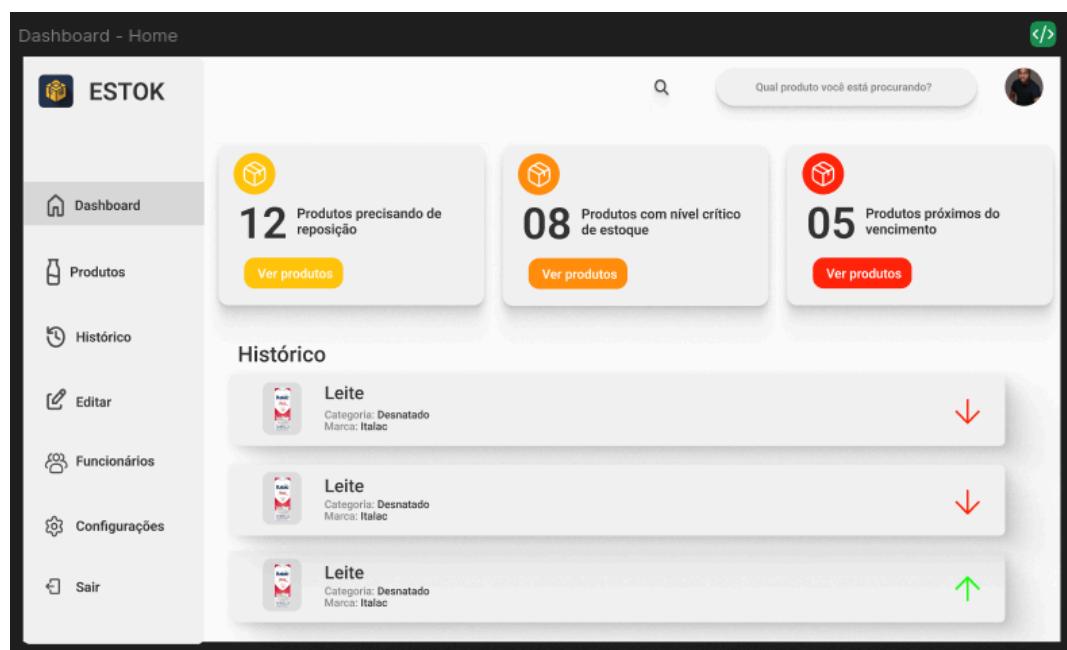
3.5 Interface do usuário

Nesta etapa da criação do nosso projeto construímos uma interface visual com foco na clareza e usabilidade do gerente. Essa parte representa o ponto de interação do gerente com todo o sistema, nela que os dados provenientes do backend serão enfim mostrados. A construção da interface considerou não apenas a

estética, mas principalmente a experiência do usuário durante o acompanhamento do estoque.

O desenvolvimento começou com a prototipação no Figma, onde foram definidos as telas fundamentais como Dashboard, Histórico e Cadastro. Essa fase permitiu organizar visualmente as informações e validar os fluxos de navegação antes de começar a desenvolver em código. A prototipagem da tela principal de dashboard está presente na figura 28.

Figura 28 — Protótipo de dashboard Figma



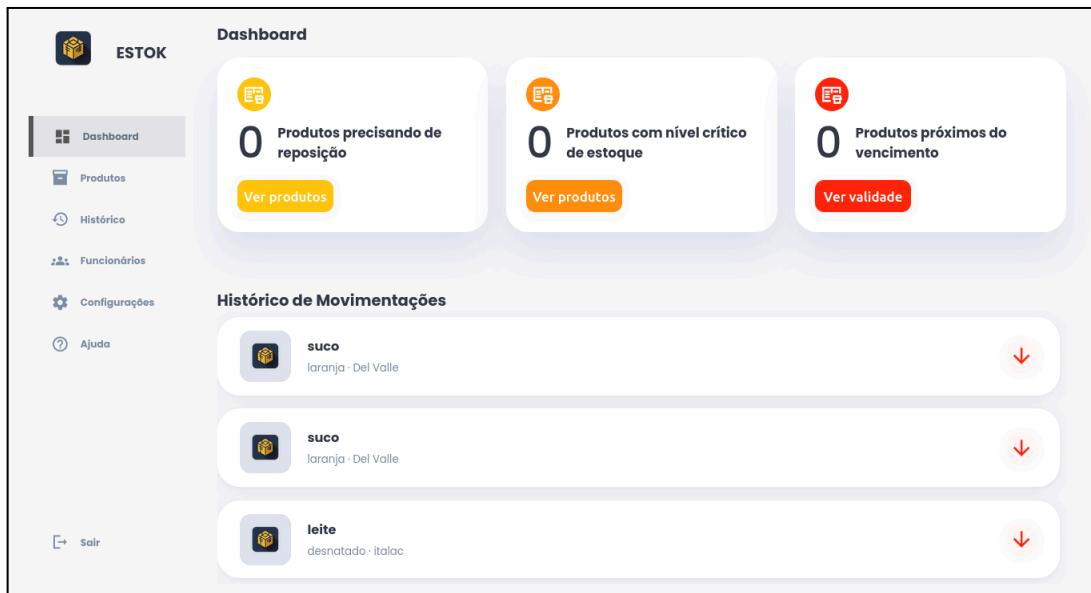
Fonte: Do próprio autor, 2025

Depois do planejamento, a interface foi estruturada em React, utilizando componentes reutilizáveis para garantir modularidade e manutenção simples ao longo do projeto.

Cada página da interface executa uma função específica dentro da lógica do sistema. O Dashboard, tela principal, concentra os status dos produtos, que são categorizados por nível de necessidade de reposição.

Representado na figura 29, está a tela principal do site Estok. A tela exibe as opções de escolha de página ao lado esquerdo, além dos cards que mostram os produtos que precisam de reposição, que estão em nível crítico ou próximo da data de vencimento. Também exibe as últimas movimentações da prateleira no componente histórico de movimentações.

Figura 29 — Tela principal dashboard React



Fonte: Do próprio autor, 2025

A navegação lateral é essencial para estruturar o acesso às funcionalidades, reduzindo o tempo de deslocamento entre telas e mantendo o usuário sempre orientado dentro do sistema. Os cards de reposição atuam como indicadores preventivos, permitindo que o responsável antecipe compras ou reorganize o estoque antes que a falta de produtos impacte o atendimento ou a operação, enquanto os cards de nível crítico reforçam situações de urgência, evitando perda de vendas, interrupções no fluxo de trabalho e falhas na disponibilidade de itens essenciais. Por sua vez, os cards de vencimento próximo acrescentam uma camada de controle de qualidade ao alertar sobre produtos que precisam ser consumidos, vendidos ou retirados rapidamente, prevenindo prejuízos e garantindo conformidade com normas sanitárias. Já o histórico de movimentações fornece um breve resumo de entradas, saídas e alterações, permitindo auditorias rápidas.

A página de histórico, apresentada na figura 30, apresenta uma lista detalhada de entradas e saídas com data e hora, permitindo acompanhamento cronológico das movimentações captadas.

Figura 30 — Página de histórico

Histórico de Movimentações						
Produto	Tipo	Marca	Validade	Qtde	Status	Horário
 suco	laranja	Del Valle	31-08-2025	0	 Saída	21:19:47
 suco	laranja	Del Valle	31-12-2025	0	 Saída	21:19:47
 leite	desnatado italac		31-12-2025	0	 Saída	21:19:46
 suco	laranja	Del Valle	31-12-2025	0	 Entrada	21:16:14

Fonte: Do próprio autor, 2025

A comunicação com o backend para alimentar essas páginas ocorre de forma assíncrona. Essa integração foi feita de forma a garantir que as informações exibidas na interface estejam sempre atualizadas. A cada requisição, os dados retornam do servidor e são armazenados em estados da aplicação, permitindo a reatividade e atualização dinâmica dos componentes, como mostra a figura.

Usamos o TailwindCSS na estilização, que trouxe eficiência na construção do visual, permitindo estilização mais rápida. A interface responsiva adapta-se a diferentes resoluções. Componentes como cards, tabelas e indicadores visuais foram projetados para transmitir ao usuário apenas o essencial, sem sobrecarregar a experiência com elementos desnecessários.

Durante o desenvolvimento, foram enfrentados desafios relacionados à sincronização dos dados com o backend e ao comportamento responsivo da interface. A solução adotada foi o uso de useEffect (Gancho de efeitos colaterais do React) para atualizações periódicas das informações e a aplicação do Tailwind para garantir fluidez visual entre diferentes dispositivos.

4 Considerações Finais

A implementação do Estok demonstrou que a tecnologia RFID é uma alternativa viável para aumentar a rastreabilidade de produtos nas prateleiras de mercados autônomos. O Estok conseguiu detectar corretamente a entrada e saída de itens com as tags, respeitando as limitações de alcance do leitor e evidenciando que o conceito central do projeto funciona na prática. Embora a leitura tenha apresentado interferências em produtos com estruturas metálicas, essa limitação é conhecida na literatura e pode ser mitigada com o uso de etiquetas anti metal, o que não invalida os resultados obtidos.

O desenvolvimento utilizando leitura de protocolos proprietários permitiu que o grupo adquirisse domínio sobre a integração entre hardware e software, assim como aumentou nosso entendimento sobre o fluxo de dados entre ESP32 e backend. O conhecimento em interpretação técnica da documentação de terceiros foi essencial para a construção do firmware. Esses aprendizados foram fundamentais para construir um sistema funcional capaz de comunicar, registrar e exibir dados de forma consistente.

Os resultados preliminares indicam que o Estok cumpre parcialmente seu objetivo, pois ainda não foi submetido a testes em ambiente real de operação. Contudo, ele já se mostrou capaz de registrar movimentações e manter o histórico das mudanças detectadas, validando a proposta de aumentar a rastreabilidade. Além disso, o volume de dados gerado pelo sistema abre caminho para aplicações futuras, como modelos preditivos capazes de antecipar reposições com base em padrões de consumo.

Diante do avanço alcançado, o Estok se torna uma contribuição prática e acadêmica que reforça o uso da tecnologia RFID no contexto de mercados autônomos e adiciona conteúdo em língua portuguesa voltado à automação aplicada ao varejo e as possibilidades do uso de RFID no mesmo. A próxima etapa natural é testar o sistema em ambiente real para avaliar seu desempenho operacional completo.

REFERÊNCIAS

BOMJORNO, Maria; HERINGER, Eudiman; MADUREIRA, Eduardo. **O IMPACTO DA INTERNET DAS COISAS (IoT) E DO BIG DATA NOS PROCESSOS LOGÍSTICOS: INOVAÇÕES NO PICKING E PACKING NA ERA DA INDÚSTRIA 4.0.** Paraná, Centro Universitário FAG. 2025.

CAMPOS, Matheus. **Análise da viabilidade de implementação de etiquetas RFID na construção civil.** 2021. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Civil) – Universidade Tecnológica Federal do Paraná, Departamento Acadêmico de Construção Civil, Campo Mourão, 2021.

CARDOSO, Carlos. **HTML4.** Rio de Janeiro: Axcel Books, 1999.

CARVALHO, Rafael et al. **IOT PARA CONTROLE E GERENCIAMENTO RESIDENCIAL.** Rio de Janeiro: RevistaFT, 2023.

COSTA, Alexander. **RFControl: sistema de gerência de estoque utilizando RFID Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação).** Minas Gerais: Universidade Federal de Ouro Preto, Instituto de Ciências Exatas e Aplicadas, 2018.

DATE, C. J.. **Instrução a Sistemas de Bancos de Dados.** 8a Edição. Rio de Janeiro: Elsevier, 2003.

DUCKETT, Jon. **HTML & CSS: projete e construa websites.** Rio de Janeiro: Alta Books, 2016.

EEEP. **Curso Técnico de Redes de Computadores.** Ceará: Escola Estadual De Educação Profissional, [s.d.].

EIS, Diego et al. **HTML5 e CSS3: com farinha e pimenta.** São Paulo, Brasil: Tableless, 2012.

FERREIRA, Juliana Borges. **Desenvolvimento de etiquetas planares passivas UHF RFID em superfícies metálicas e não metálicas com a proposta de uma figura de mérito para avaliação.** 2021. [Tese de Doutorado]. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2021. Orientador: Dr. Álvaro Augusto Almeida de Salles.

FOWLER, Martin; SCOTT, Kendall. **UML Essencial Um breve guia para a linguagem-padrão de modelagem de objetos.** 2. ed. Porto Alegre: Bookman, 2000.

FRAGOSO, Wallace. **Guia Prático HTML & CSS: Aprenda os conceitos básicos de como construir páginas web.** [S.I.]: DevAcademy, [s.d.]. E-Book.

GONÇALVES, Luiz. **Sistema de Controle de Acesso Físico por Dispositivos com Identificação por RFID e Autenticação por Biometria de Impressão Digital.** 2025. Monografia (Graduação em Engenharia de Controle e Automação) – Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2025.

GOOGLE. **Entender os projetos do Firebase.** Disponível em:
<https://firebase.google.com/docs/projects/learn-more?hl=pt-br>. Acesso em: 20 mai. 2025.

GRONER, Loiane. **Estruturas de dados e algoritmos com JavaScript.** 2. ed. São Paulo: Novatec, 2019.

GUEDES, Gilleanes. **UML2 UMA ABORDAGEM PRÁTICA.** 3. ed. São Paulo: Novatec, 2018.

IBM. **Rational Software Architect Standard Edition.** [S. I.]: IBM, 2021. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams>. Acesso em: 20 out. 2025.

IBM. **Rational Software Architect.** [S. I.]: IBM, 2025. Disponível em:
<https://www.ibm.com/docs/pt-br/rational-soft-arch/10.0?topic=diagrams-activity>. Acesso em: 11 ago. 2024.

IBM. **Rational Software Modeler.** [S. I.]: IBM, 2021. Disponível em:
<https://www.ibm.com/docs/pt-br/rational-soft-arch/10.0?topic=diagrams-use-case>. Acesso em: 10 ago. 2024.

INFORCHANNEL. **Lojas Renner conclui projeto de implementação de etiquetas RFID.** [S.I.]: InforChannel, 2022. Disponível em:
<https://inforchannel.com.br/2022/06/30/lojas-renner-conclui-projeto-de-implementacao-de-etiquetas-rfid/>. Acesso em: 11 ago. 2025.

KNIGHT, Indira. **Conectando o Arduino à web: Desenvolvimento de frontend usando JavaScript.** 1. ed. São Paulo: Novatec Editora, 2018.

KPMG. **Pesquisa Abrappe de Perdas no Varejo Brasileiro 2024.** [s.l.], 2024. Disponível em:
<https://kpmg.com/br/pt/home/insights/2024/11/pesquisa-abrappe-2024.html>. Acesso em: 27 mai. 2025.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos de metodologia científica.** São Paulo: Atlas, 2003. E-Book.

LEÃO, Leonardo. **Minimercados autônomos ganham força no Brasil.** Belo Horizonte: Diário do Comércio, 2024. Disponível em:
<https://diariodocomercio.com.br/negocios/minimercados-autonomos-ganham-forca-no-brasil/>. Acesso em: 24 mai. 2025.

LEPSEN, Edécio. **Lógica de Programação e Algoritmos com JavaScript: UMA INTRODUÇÃO À PROGRAMAÇÃO DE COMPUTADORES COM EXEMPLOS E EXERCÍCIOS PARA INICIANTES.** 1. ed. São Paulo: Novatec, 2018.

LUCIANO, Josué; ALVES, Wallison. **PADRÃO DE ARQUITETURA MVC: MODEL-VIEW-CONTROLLER.** Bebedouro, UNIFAFIBE. 2011.

MACHADO , Kheronn. **Angular 11 e Firebase:** Construindo uma aplicação integrada com a plataforma do Google. Brasil: Casa do Código, 2021.

MAGRANI, Eduardo. **A Internet Das Coisas.** Rio de Janeiro: FGV Editora, 2018.

META. **React A biblioteca para web e interfaces de usuário nativas.** [S.I.]: Meta Open Source, [s.d.]. Disponível em: <https://pt-br.react.dev/>. Acesso em: 4 nov. 2025.

META. **React Reconhecimentos.** [S.I.]: Meta Open Source, [s.d.]. Disponível em: <https://pt-br.react.dev/community/acknowledgements>. Acesso em: 4 nov. 2025.

MICROSOFT. **Bem-vindo de volta ao C++ – C++ moderno.** [S.I.]: Microsoft, 2023. Disponível em:
<https://learn.microsoft.com/pt-br/cpp/cpp/welcome-back-to-cpp-modern-cpp?view=msvc-170>. Acesso em 11 ago. 2025.

Microsoft. **Criar um diagrama de atividade UML.** [S.I.]: Microsoft, 2025. Disponível em:
<https://support.microsoft.com/pt-br/topic/criar-um-diagrama-de-atividade-uml-19745d-ae-2872-4455-a906-13b736f01685#officeversion=windows>. Acesso em: 11 ago. 2025.

Microsoft. **Criar um diagrama de sequência UML.** [S.I.]: Microsoft, 2025. Disponível em:
<https://support.microsoft.com/pt-br/topic/criar-um-diagrama-de-sequ%C3%A3Ancia-u> ml-c61c371b-b150-4958-b128-902000133b26. Acesso em: 12 ago. 2025.

MORAES, William. **Construindo aplicações com NodeJS.** 4. ed. São Paulo: Novatec Editora, 2023.

OLIVEIRA, Ana. **Estudo paramétrico e análise de impedância de uma etiqueta RFID UHF passiva.** Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica). João Pessoa: Instituto Federal da Paraíba, 2023.

OLIVEIRA, Daniel. **Uma proposta de arquitetura para Single-Page Applications.** Recife: Universidade Federal de Pernambuco Centro de Informática, 2017.

PEREIRA, Caio. Pereira. [S.I.]: Casa do Código, 2014.

PET. **Introdução à Programação Embarcada.** Minas Gerais: Itajubá, 2021.

PORDEUS, Igor Costa. **Desenvolvimento de um aplicativo para adoção de pets utilizando Flutter e Firebase.** 2021. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Centro Universitário Christus, Fortaleza, 2021.

POWERS, Shelley. **Aprendendo Node.** 1. ed. Brasil: Novatec Editora, 2017.

PREDIGER, Daniel; FREITAS, Edison; SILVEIRA, Sidnei. **Modelo de aplicabilidade de sistema RFID para rastreabilidade na indústria alimentícia.** Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Rio Grande do Sul: Universidade Federal de Santa Maria, 2014. Disponível em:
<http://repositorio.ufsm.br/handle/1/12810>. Acesso em: 25 maio 2025.

PUC-Rio. **UML: Diagrama de Classes.** Rio de Janeiro: PUC-Rio, [s. d.]. Disponível em:
https://moodle.unesp.br/pluginfile.php/25933/mod_resource/content/1/diagrama_classes.pdf. Acesso em: 20 ago. 2025

ROVAROTO, Isabela. **Como esta empresa de calçados deixou de perder vendas por falta de estoque em suas 340 lojas.** [S.I.]: Exame, 2024. Disponível em:
<https://exame.com/negocios/como-esta-empresa-de-calcados-deixou-de-perder-venda-por-falta-de-estoque-em-suas-340-lojas/>. Acesso em: 24 mai. 2025.

SENSORMATIC. Renner reduz 87% da ruptura de seus estoques com tecnologia RFID da Sensormatic Solutions. Sensormatic. Disponível em:. Acesso em: 20 ago. 2025.

SILVA, Antonio; SANTOS JUNIOR, João. Vacmonitor: uma aplicação para o monitoramento de vacinas utilizando Flutter e Firebase. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação). Paraíba: Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, 2023.

SILVA, Maurício Samy. Javascript: Guia do Programador. 1. ed. São Paulo: Novatec, 2010.

SILVEIRA, Marcelo; PRATES, Rubens. HTML 4: guia de consulta rápida. São Paulo: Novatec, 2001.

STROUSTRUP, Bjarne. Princípios e práticas de programação com C++. 3. ed. Tradução de Eveline Vieira Machado. Revisão técnica de Daniel Antônio Callegari. Porto Alegre: Bookman, 2025.

TORTORELLI, Henrique. Estratégias de Negócios Internacionais: O caso da empresa Amazon.com. Covilhã: Universidade Da Beira Interior, 2018.

WENCESLAU, Fernando. Como a automação e a tecnologia preditiva estão mudando a gestão de estoque. Santa Cataria: Economia SC, 2024. Disponível em: <https://economiasc.com/2024/07/25/como-a-automacao-e-a-tecnologia-preditiva-esta-o-mudando-a-gestao-de-estoque/>. Acesso em: 25 mai. 2025.

WIENER, Richard; PINSON, Lewis J. Programação Orientada para Objeto e C++. São Paulo: Makron Books, 1991.

WIERUCH, Robin; ROMERO, Claudio. The Road to learn React (Português). [S.I.]: Leanpub, 2018. E-Book.

GLOSSÁRIO

Antena RFID – Componente responsável por emitir e receber ondas de rádio, permitindo a comunicação entre leitor e etiqueta.

API (Application Programming Interface) – Conjunto de regras que permite a comunicação entre sistemas diferentes.

Acuracidade de Estoque – Medida que indica o quanto preciso está o inventário registrado em relação ao inventário real.

Backend – Parte lógica do sistema responsável pelo processamento, regras de negócio e comunicação com o banco de dados.

Big Data – Grande volume de dados coletados e analisados para identificar padrões e auxiliar na tomada de decisões.

Dashboard – Interface visual que exibe informações relevantes de forma organizada.

Entrada/Saída (Movimentação) – Registro da ação de colocar ou retirar um produto da prateleira.

Etiqueta RFID (Tag) – Dispositivo identificado por radiofrequência, contendo um código único para cada produto.

ESP32 – Microcontrolador utilizado para processar leituras RFID e enviar dados ao backend.

Firmware – Código embarcado no microcontrolador que controla seu funcionamento.

Lógica de Aplicação – Conjunto de regras que transforma os dados recebidos pelo hardware em informações úteis ao sistema.

Prateleira Inteligente – Estrutura equipada com sensores ou RFID para rastrear movimentações de produtos.

Protótipo – Primeira versão funcional do sistema ou dispositivo construída para testes.

Repositório (de produtos) – Local na prateleira que possui leitores para identificação das tags.

Ruptura de Estoque – Situação em que o produto não está disponível na prateleira, mesmo existindo no estoque físico.

SPA (Single Page Application) – Aplicação web que carrega conteúdo dinamicamente em uma única página.

Uso de Diagramas UML – Representação visual da arquitetura e comportamento do sistema durante o desenvolvimento.

ANEXOS

ANEXO A - Detalhamento Hello World C++

Stroustrup (2025) explica o funcionamento do código, da seguinte forma:

- “import std” está sendo solicitado para que o computador forneça as funcionalidades disponíveis na biblioteca padrão std.
- “main” todo projeto C++ deve ter a função main. Sendo ela quem mostra para o computador onde ele deve começar a executar o código.
- “int” significando “inteiro”, sendo o tipo de retorno que o programa espera receber após a execução do código.
- “std” usado para identificar o cout dentro da biblioteca padrão do C++ que foi definida no início do código.
- “cout” abreviado de "character output stream" (fluxo de saída de caracteres), cout junto com o operador de saída << exibirá caracteres na tela.
- “<<” operador de saída usado junto com cout para exibir caracteres na tela.
- “ "Hello, World!\n" ” a mensagem que será exibida junto com um marcador de nova linha (\n). Marcador de nova linha coloca o cursor na próxima linha.
- “return 0” o retorno do número 0 pela função main indica que a execução do código foi bem-sucedida.

ANEXO B - Principais etiquetas HTML

Principais tags que o HTML possui segundo Silveira e Prates (2001).

- “<!doctype html>” indica o tipo de documento que está sendo escrito e a versão do HTML. No HTML5, essa declaração é obrigatória e deve ser a primeira linha do código.
- “<html>” Representa o elemento raiz de uma página HTML.
- “<head>” Armazena informações que não são diretamente exibidas nas páginas.
- “<body>” Representa o corpo da página, onde ficam todos os elementos visíveis ao usuário, como textos, imagens, vídeos, botões e demais conteúdos interativos.

ANEXO C - Etiquetas criação do formulário cadastro de produto

Conforme explica Duckett (2016), segue o funcionamento e a função de cada elemento HTML utilizado.

- “<form>” elemento que abrange todo formulário, utilizado para coleta das informações.
- “<h2>” elemento de títulos. As tags h1 a h6. Definem diversos níveis de títulos. Onde “<h1>” indica o título com maior importância e “<h6>” o com menor destaque.
- “<fieldset>” agrupa os campos do formulário, organizando o conteúdo com uma borda ao redor deles e simplificando a visualização das informações.
- “<legend>” concede um título para o grupo de campos do <fieldset>.
- “<label>” define um rótulo descritivo para um campo do formulário, ajudando na organização e acessibilidade.
- “<input>” criam campos interativos no formulário, como textos, números e botões de envio. Seu comportamento é definido no atributo type, que determina o tipo de dado aceito.
- “
” define uma quebra de linha.

ANEXO D - Detalhamento dos papéis de atributo na conexão entre HTML e CSS

Conforme Duckett (2016), segue o funcionamento das tags envolvidas no processo de conexão entre HTML e CSS.

- “<link>” serve para comunicar ao navegador onde procurar o arquivo CSS.
- “href” mostra em específico onde está localizada a folha de estilo.
- “type” declara o tipo do documento que o link está referenciando.
- “rel” informa a relação entre a página HTML e o arquivo que referencia, quando o link aponta para um documento CSS, o dado declarado deve ser stylesheet.

ANEXO E - Explicação dos seletores e das declarações CSS

Conforme Fragoso ([s.d.]), segue a explicação de algumas seletores CSS utilizados na estilização do código.

- “body” do HTML, indica tudo no corpo da página.
- “font-family” usada para declarar a fonte que será utilizada no texto de qualquer elemento HTML em que a regra CSS está agindo.
- “padding” define uma distância entre o conteúdo e a borda. Foi utilizado para definir o espaçamento entre os campos na tela de cadastro.
- “background” determina a cor que será utilizada em determinado elemento.
- “width” relata a largura de um elemento.
- “margin” controla o espaço entre blocos e elementos.
- “color” possibilita definir a cor do texto dentro de um elemento, foi usado para definir a cor do nosso botão.

ANEXO F - Explicação componente React

Segundo a documentação oficial do React (META, [s.d.]), segue a explicação das declarações do código previamente apresentado.

- “function” O código começa com a declaração de uma função (function) que aceitam propriedades (props).
- “return” Essa declaração delimita o que será retornado como componente. Os elementos declarados depois dela entre os parênteses farão parte do componente retornado em questão.
- “export” Seguindo os preceitos de modularidade do React. Os componentes podem ser criados em arquivos diferentes, tendo que ser exportados ao final de sua criação. A declaração “export”, comumente seguida da função principal do componente em questão, define o que será visível para importação no seu projeto.

ANEXO G - Explicação linhas código Node.js.

Segundo Powers (2017), segue a explicação das linhas do código de junção de nome utilizando Node.js.

- Linha 1: Declara uma função chamada “JuntarNomes” e, dentro dos parênteses, são solicitados dois parâmetros separados por vírgula (nome e sobrenome do usuário).
- Linha 2: Retorna o nome e o sobrenome juntos através do sinal de “+”, com um espaço vazio entre eles, declarado entre aspas, que representa o espaçamento entre nome e sobrenome.
- Linha 5: Chama a função passando como parâmetros o nome e o sobrenome separados por vírgula.