CENTRO ESTADUAL DE EDUCAÇÃO TECNOLOGICA PAULA SOUZA
ETEC TRAJANO CAMARGO
3º ANO – ENSINO MÉDIO INTEGRADO AO CURSO DE TÉCNICO EM ELETROELETRÔNICA

LEONARDO CAMPOS CAMARGO

LUCAS GRANUSSO

VICTOR MANOEL OLIVEIRA PERES

PARKEASY: DESENVOLVIMENTO DE UM SISTEMA DE ESTACIONAMENTO INTELIGENTE BASEADO EM IoT

LIMEIRA, SP

2025

LEONARDO CAMPOS CAMARGO

LUCAS GRANUSSO

VICTOR MANOEL OLIVEIRA PERES

PARKEASY: DESENVOLVIMENTO DE UM SISTEMA DE ESTACIONAMENTO INTELIGENTE BASEADO EM IoT

Trabalho de conclusão de curso apresentado como requisito para a obtenção da formação de técnico em eletroeletrônica.

Orientador: Profº. Carlos Alberto Serpeloni

LIMEIRA, SP

2025

AGRADECIMENTOS

Finalizar este projeto representa não apenas a conclusão de uma etapa acadêmica, mas também o resultado de um percurso construído com o apoio de muitas pessoas às quais somos profundamente gratos.

A Deus, nosso agradecimento maior, por ter nos dado a força necessária, a saúde e a coragem para seguir em frente mesmo diante dos desafios. Sua presença foi essencial em todos os momentos desta caminhada.

Às nossas famílias, que foram nossa base e nosso alicerce. O apoio constante, o incentivo diário e a compreensão incondicional fizeram toda a diferença. Em cada palavra de apoio, cada gesto de cuidado e cada sacrifício, encontramos a motivação para continuar.

Aos professores e à instituição de ensino, nossa gratidão pelo conhecimento compartilhado, pela paciência nas dúvidas e pela dedicação em formar não apenas profissionais, mas pessoas críticas, preparadas e comprometidas. Suas orientações foram fundamentais para a construção deste trabalho e para o nosso crescimento ao longo do curso.

Aos colegas de classe, amigos e amigas, agradecemos pela parceria construída ao longo dessa jornada. Estivemos juntos nos momentos de dificuldade e nas pequenas vitórias, nos ajudamos mutuamente, trocamos conhecimentos, experiências e muitas risadas que tornaram o caminho mais leve.

Por fim, agradecemos a todos aqueles que, de alguma forma, contribuíram para este projeto acontecer. Cada ajuda, conselho ou gesto de incentivo teve um papel importante na realização deste trabalho.

RESUMO

Nosso projeto ParkEasy tem como objetivo desenvolver e analisar um sistema de estacionamento inteligente, que visa otimizar o processo de busca por vagas por meio da integração entre *hardware*, *software* e serviços em nuvem. O estudo se propõe a investigar a aplicação da internet das coisas (*IoT*) como solução tecnológica para monitorar a ocupação de vagas em tempo real e permitir que o usuário reserve uma vaga antes de chegar ao local, contribuindo para a redução do tempo de procura, do consumo de combustível e do congestionamento em áreas urbanas.

A pesquisa foi conduzida com base em uma abordagem experimental e aplicada, envolvendo o desenvolvimento de um protótipo funcional. O sistema foi implementado utilizando o microcontrolador **ESP32**, responsável pela leitura dos sensores ultrassônicos **HC-SR04** para detecção de veículos, controle de **LEDs** para sinalização e de um **buzzer** para alertas sonoros. A comunicação entre o *hardware* e o site ocorreu através da **plataforma Google *Firebase***, que gerenciou o banco de dados em tempo real e a autenticação de usuários. O software foi desenvolvido em **HTML** *(HyperText Markup Language)*, **CSS** *(Cascading Style Sheets ou Folha de Estilo em Cascata)* e **JS (***JavaScript***)**, integrando-se ao Firebase para exibir o status das vagas e permitir ações de reserva, cancelamento e check-in. Durante o desenvolvimento, foram realizadas etapas de teste, correção de falhas e ajustes de sincronização entre os módulos.

O protótipo do ParkEasy apresentou funcionamento satisfatório e estável após os testes realizado. A comunicação entre o hardware e a nuvem mostrou-se eficiente, permitindo a atualização instantânea das vagas no site e o controle visual por LEDs. A substituição do login com Google pelo sistema de autenticação via e-mail e senha do Firebase aumentou a confiabilidade e a consistência do sistema. Os resultados demonstraram que a proposta é viável e pode ser aplicada em estacionamento de pequeno e médio porte, com baixo custo e alta eficiência.

O desenvolvimento do ParkEasy comprovou a viabilidade da utilização de tecnologias de internet das coisas (IoT) na automação e controle de estacionamentos. O projeto contribui para a redução de tempo e recursos por vagas, além de aprimorar a gestão de espaços urbanos. Como sugestões para trabalhos futuros, propõe-se a integração de um sistema de barreiras automatizadas, pagamento digital e painel de análise de dados. O estudo evidencia a importância da IoT como ferramenta para o avanço de soluções inteligentes e sustentáveis voltadas à mobilidade urbana.

ABSTRACT

Our project, **ParkEasy**, aims to develop and analyze an intelligent parking system designed to optimize the process of finding parking spaces through the integration of hardware, software, and cloud services. The study seeks to investigate the application of the Internet of Things (IoT) as a technological solution to monitor parking space occupancy in real time and allow users to reserve a spot before arriving at the location. This contributes to reducing search time, fuel consumption, and traffic congestion in urban areas.

The research was conducted using an experimental and applied approach, involving the development of a functional prototype. The system was implemented using the **ESP32 microcontroller**, responsible for reading **HC-SR04 ultrasonic sensors** to detect vehicles, controlling **LEDs** for visual signaling, and a **buzzer** for sound alerts. Communication between the hardware and the website was handled through the **Google Firebase platform**, which managed the real-time database and user authentication. The software was developed using **HTML, CSS, and JavaScript**, integrating with Firebase to display parking space status and enable reservation, cancellation, and check-in actions. During development, testing phases, bug fixes, and synchronization adjustments between modules were carried out.

The ParkEasy prototype demonstrated satisfactory and stable performance after the tests. The communication between hardware and the cloud proved efficient, allowing instant updates of parking space status on the website and visual control via LEDs. Replacing Google login with Firebase's email and password authentication system increased the system's reliability and consistency. The results showed that the proposal is viable and can be applied to **small and medium-sized parking lots**, offering **low cost and high efficiency**.

The development of ParkEasy confirmed the feasibility of using **Internet of Things (IoT)** technologies in parking automation and control. The project contributes to reducing the time and resources spent searching for parking spaces, while also improving urban space management. For future work, it is suggested to integrate **automated barrier systems, digital payment**, and a **data analysis dashboard**. The study highlights the importance of IoT as a tool for advancing smart and sustainable solutions aimed at urban mobility.

SUMÁRIO

1. INTRODUÇÃO

O crescimento urbano e o aumento do número de veículos nas cidades têm gerado um problema recorrente: a dificuldade em encontrar vagas de estacionamento disponíveis. Esse desafio afeta diretamente o trânsito, o consumo de combustível e o tempo gasto pelos motoristas, além de contribuir para o aumento da poluição e do estresse diário. Diante desse cenário, o uso da tecnologia surge como uma ferramenta essencial para tornar a mobilidade mais eficiente e sustentável.

Nesse contexto, o projeto ParkEasy foi desenvolvido como uma proposta de solução prática e acessível, utilizando conceitos de Internet das Coisas para automatizar e otimizar a gestão de vagas de estacionamento. O sistema visa não apenas monitorar a ocupação das vagas em tempo real, mas também permitir que o usuário reserve uma vaga antecipadamente, garantindo mais conforto e agilidade ao chegar ao local.

O trabalho apresentado tem como objetivo demonstrar a aplicação dos conhecimentos adquiridos ao longo do curso técnico em Eletroeletrônica, por meio do desenvolvimento de um protótipo funcional que integra hardware, software e serviços em nuvem. Assim, o ParkEasy busca unir tecnologia e praticidade em um sistema inteligente, capaz de representar uma alternativa viável para o gerenciamento moderno de estacionamentos.

1.1 Visão Geral do Projeto

O ParkEasy é um sistema de estacionamento inteligente desenvolvido com base nos princípios da Internet das Coisas, cujo objetivo é otimizar a utilização de vagas de estacionamento e facilitar a experiência dos motoristas. O sistema é composto por três partes principais: o hardware, responsável pela detecção e sinalização das vagas; a nuvem, que centraliza e processa as informações; e o software, que permite a interação do usuário com o sistema através de um site.

O hardware, construído a partir de um microcontrolador ESP32 (*Espressif32*), utiliza sensores ultrassônicos (HC-SR04) para identificar a presença de veículos em cada vaga e LEDs (*Light Emitting Diode*) para indicar visualmente o status (livre, ocupada ou reservada). Esses dados são enviados em tempo real para a nuvem, onde o Google Firebase atua como intermediário entre o hardware e o site, armazenando e atualizando as informações continuamente.

Por meio do software, desenvolvido em HTML, CSS e JS, o usuário pode visualizar o mapa do estacionamento, reservar uma vaga e realizar o check-in ao chegar. Todo o sistema opera em sincronia, atualizando automaticamente o status das vagas conforme as mudanças detectadas pelo hardware.

1.2 Objetivo do Sistema

O objetivo principal do ParkEasy é desenvolver uma solução tecnológica capaz de monitorar e gerenciar vagas de estacionamento em tempo real, integrando dispositivos físicos e plataformas digitais de forma automatizada e eficiente.

Entre os objetivos específicos, destacam-se:

- Aplicar os conceitos de IoT para conectar sensores e sistemas online;
- Reduzir o tempo e o estresse na procura por vagas;
- Melhorar a organização e o fluxo de veículos em estacionamentos;
- Demonstrar a viabilidade de um sistema automatizado de baixo custo;
- Permitir a reserva e o acompanhamento de vagas via interface web.

Dessa forma, o ParkEasy busca oferecer uma alternativa moderna e acessível para otimizar o gerenciamento de estacionamentos, unindo tecnologia, praticidade e sustentabilidade.

1.3 Público-Alvo

O sistema ParkEasy foi concebido com um público-alvo primário abrangente: todos os condutores de veículos automotores que necessitam utilizar vagas de estacionamento em estabelecimentos privados de médio a grande porte. Isso inclui frequentadores de shoppings, centros, complexos comerciais, centros empresariais, universidades, hospitais, aeroportos e outros locais onde a busca por uma vaga livre pode ser uma experiência frustrante e demorada.

Embora, em essência, qualquer pessoa habilitada a dirigir seja um potencial usuário, o ParkEasy destina-se especialmente àqueles que valorizam **conveniência, otimização do tempo e redução de estresse**. O sistema atende diretamente à necessidade do condutor moderno que busca soluções tecnológicas para simplificar tarefas cotidianas.

Ao fornecer informações em tempo real sobre a disponibilidade de vagas e oferecer a funcionalidade de reserva, o ParkEasy beneficia diretamente o motorista que:

- Frequentemente enfrenta dificuldades para encontrar vagas em horários de pico.
- Deseja evitar a circulação excessiva dentro do estacionamento, economizando combustível e tempo.
- Precisa garantir uma vaga próxima a um acesso específico (em futuras implementações com vagas categorizadas).
- Está confortável em utilizar interfaces web (e, potencialmente, aplicativos móveis em futuras versões) para planejar sua chegada.

Secundariamente, o sistema também beneficia os **administradores dos estabelecimentos**, que passam a contar com uma ferramenta para gerenciar melhor o fluxo de veículos, entender os padrões de ocupação e, potencialmente, aumentar a satisfação dos seus clientes ao oferecer um serviço diferenciado e mais eficiente.

Portanto, o público-alvo do ParkEasy é o condutor comum inserido no contexto urbano moderno, que busca tecnologia para facilitar sua rotina e otimizar sua experiência ao utilizar grandes estacionamentos privados.

1.4 Problema Resolvido

O ParkEasy busca solucionar um dos principais problemas enfrentados em áreas urbanas: a dificuldade de encontrar vagas de estacionamento disponíveis. Esse desafio acarreta aumento do congestionamento, desperdício de tempo e combustível e elevação dos níveis de poluição.

Ao permitir o monitoramento em tempo real e a reserva antecipada de vagas, o sistema reduz o tempo de procura, melhora o fluxo de veículos e otimiza o uso dos espaços. Assim, o projeto propõe uma solução tecnológica eficiente e sustentável para uma necessidade cotidiana.

2. ARQUITETURA DO SISTEMA

O sistema ParkEasy foi desenvolvido com base no conceito de Internet das Coisas, integrando *hardware*, *nuvem* e *software* em uma solução automatizada para estacionamento inteligente. Essa arquitetura modular permite a comunicação em tempo real entre os sensores, a nuvem e a interface web, garantindo a atualização instantânea do estado das vagas e a interação eficiente do usuário com o sistema.

2.1 Estrutura Geral

O sistema ParkEasy é composto por três camadas principais que trabalham de forma integrada: *Hardware*, *Nuvem* e *Software*.

Essas camadas comunicam-se entre si através da internet, formando uma estrutura de Internet das Coisas completa.

- *Hardware*: Responsável pela detecção e sinalização física das vagas de estacionamento.
- *Nuvem*: Atua como intermediário de comunicação, armazenando dados e garantindo a sincronização em tempo real.
- *Software*: Interface web acessada pelos usuários para visualizar o estado das vagas, realizar reservas e efetuar check-ins.

A estrutura geral foi projetada para garantir comunicação bidirecional em tempo real entre todos os componentes, permitindo que qualquer mudança no ambiente físico seja refletida instantaneamente no site, e vice-versa.

2.2 Componentes Principais

O sistema ParkEasy é composto por elementos de hardware, nuvem e software, cada um desempenhando um papel essencial para o funcionamento completo do estacionamento inteligente.

2.2.1 *Hardware*

Os componentes físicos são responsáveis pela detecção, sinalização e comunicação

com a nuvem:

ESP32 (Microcontrolador):

Atua como unidade central de processamento da camada física. Controla sensores, LEDs e buzzer, processa os dados coletados e envia informações ao Firebase via Wi-Fi.

Sensores Ultrassônicos (HC-SR04):

Detectam a presença de veículos medindo a distância até o objeto. Permitem identificar automaticamente quando uma vaga está ocupada ou livre.

LEDs (Verde, Amarelo e Vermelho):

Indicam visualmente o estado da vaga:

- Verde: livre
- Amarelo: reservada
- Vermelho:

ocupada Buzzer:

Dispositivo sonoro que alerta quando ocorre ocupação indevida de uma vaga reservada ou quando alguma ação requer atenção do usuário.

Protoboard e Jumpers:

Utilizados para a montagem experimental e conexão dos circuitos de forma prática e segura.

2.2.2 Nuvem

A camada de nuvem garante armazenamento seguro, comunicação em tempo real e gerenciamento de usuários:

Google Firebase Realtime Database:

Armazena os estados das vagas, reservas e informações dos usuários. Sua principal característica é a atualização em tempo real, permitindo que todas as alterações sejam refletidas instantaneamente no site e no hardware.

Firebase Authentication:

Gerencia o cadastro e login dos usuários de forma segura. Garante que apenas usuários autenticados possam realizar reservas, cancelamentos ou *check-ins*, mantendo a integridade do sistema.

2.2.3 *Software*

A camada de software permite interação do usuário com o sistema e oferece uma interface visual clara e intuitiva:

HTML: Estrutura o conteúdo do site, incluindo páginas de login, mapa de vagas e informações do sistema.

CSS: Define o estilo do site, utilizando cores para sinalização das vagas e modo escuro para melhor visualização.

JavaScript: Controla a lógica do site:

- Conecta o site ao Firebase em tempo real;
- Atualiza visualmente o estado das vagas;
- Gerencia reservas, cancelamentos e check-ins;
- Controla temporizadores de reservas e eventos de clique;
- Protege páginas, garantindo acesso apenas a usuários autenticados.

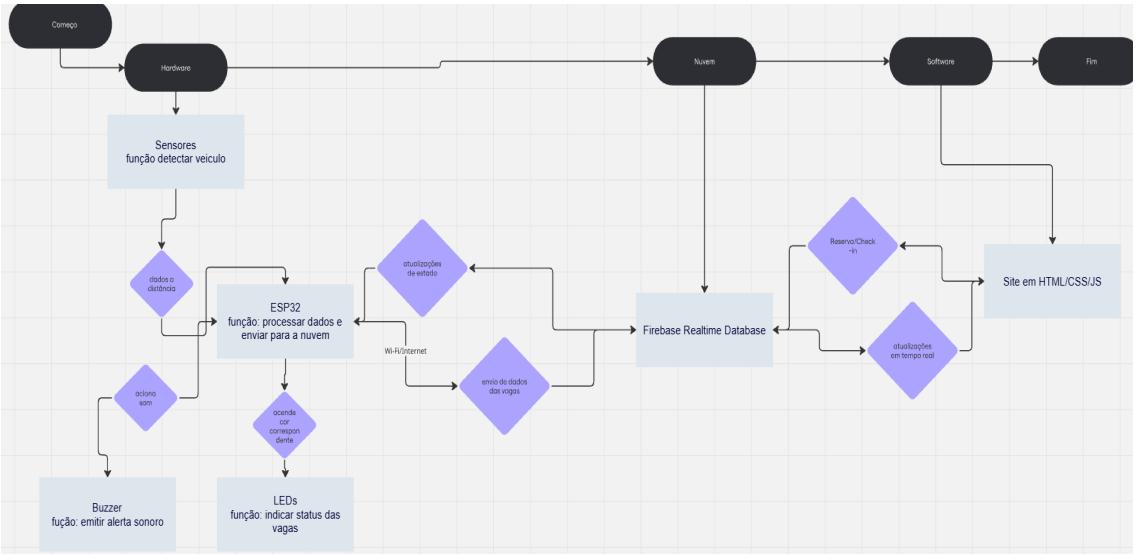
Essa integração entre *hardware*, *nuvem* e *software* garante que o ParkEasy funcione de forma coordenada e confiável, fornecendo informações precisas aos usuários e permitindo futuras expansões, como dashboards analíticos, sistemas de pagamento e barreiras físicas automáticas.

2.3 Diagrama da Arquitetura do Sistema

O diagrama apresentado ilustra a arquitetura integrada do sistema ParkEasy, detalhando a comunicação bidirecional entre os componentes físicos e digitais. Ele demonstra como os sensores e o microcontrolador ESP32 processam dados do ambiente para atualizar a nuvem via Firebase. Ao mesmo tempo, o fluxo indica como as interações no site (HTML/CSS/JS) geram atualizações em tempo real que retornam ao *hardware* para acionamento de LEDs e sinalização sonora.

A estrutura visual dessa integração pode ser observada no diagrama abaixo:

FIGURA 1 – DIAGRAMA DE PROJETO



FONTE: Acervo pessoal dos autores(2025).

3. DETALHAMENTO DOS COMPONENTES

3.1 Hardware

Hardware envolve qualquer componente físico que compõe um aparelho eletrônico. Os *hardwares* costumam ser relacionados somente aos computadores, mas também abrangem outros eletrônicos da tecnologia de consumo.

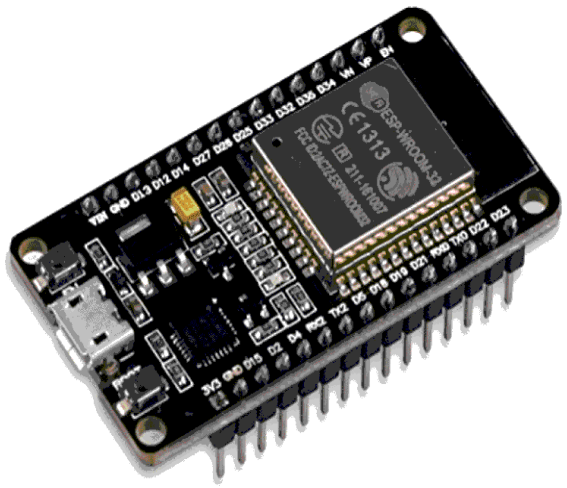
Há dois tipos de classificação comum para os *hardwares*: os internos (que ficam dentro de um dispositivo) ou os externos (que são conectados ao aparelho). Também existem componentes elétricos e mecânicos, não necessariamente ligados à computação.

Placa-mãe, processador central ou gráfico e memória RAM são exemplos de *hardwares* internos. Já monitor, teclado, mouse e webcam fazem parte dos *hardwares* externos.

3.1.1 Microcontrolador ESP32

O ESP32 é um microcontrolador desenvolvido pela *Espressif Systems*, com uma série de recursos que o tornam uma das opções mais poderosas do mercado. Ele é uma versão mais avançada do famoso ESP8266, e foi projetado para aplicações de Internet das Coisas devido à sua capacidade de conectividade **Wi-Fi e *Bluetooth* integradas.**

FIGURA 2 – EXEMPLO DE UM ESP32



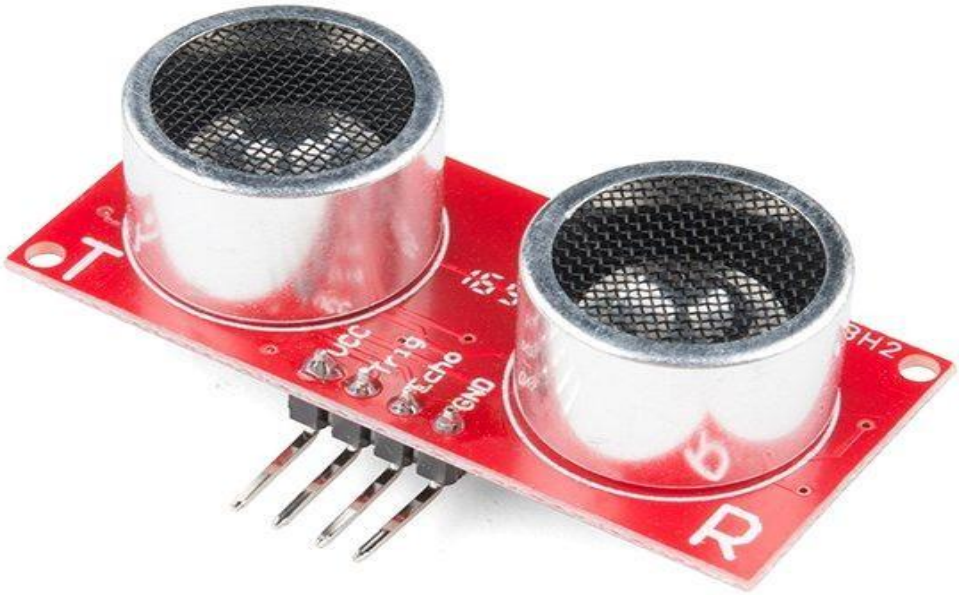
FONTE: Embarcados (2025).

Com um processador dual-core de 32 bits, o ESP32 oferece um desempenho muito superior ao de muitos microcontroladores tradicionais. Ele é ideal para projetos que exigem mais poder de processamento, conectividade e versatilidade. Além disso, ele pode ser programado em C/C++, Python, Lua, Micropython ou JavaScript, possibilitando que o desenvolvedor escolha a linguagem que mais achar conveniente para diferentes projetos.

3.1.2 Sensor Ultrassônico (HC-SR04)

Os sensores ultrassônicos são dispositivos que utilizam ondas sonoras de alta frequência para medir distâncias ou detectar objetos. Eles funcionam emitindo um pulso sonoro que, ao bater em um objeto, retorna ao sensor como um eco. A partir do tempo que o som leva para voltar, o sensor calcula a distância até o objeto.

FIGURA 3 – EXEMPLO DE UM HC SR04



FONTE: ISAAC (ProjectHub Arduino, 2025).

Esses sensores não requerem contato físico com o item a ser detectado e são amplamente utilizados em diversas áreas, como robótica, automação industrial e até em sistemas de estacionamento de veículos. Além disso, funcionam bem em ambientes terrestres, já que utilizam o ar como meio para propagação do som (MECÂNICA INDUSTRIAL, 2025).

O HC-SR04 é um sensor ultrassônico usado para medir distâncias sem contato

físico. Ele funciona emitindo um som inaudível (40 kHz) pelo pino *Trigger*, que reflete em um objeto e retorna ao sensor, sendo captado pelo pino *Echo*. Com base no tempo que o som leva para ir e voltar, ele calcula a distância até o objeto. É muito usado em robôs, sistemas de estacionamento e projetos com Arduino, com alcance de 2 a 400 cm e boa precisão (MECÂNICA INDUSTRIAL, 2025).

3.1.3 LEDs e Sinalização Visual

Os LEDs são componentes fundamentais para a sinalização visual do sistema ParkEasy, permitindo identificar de forma imediata o estado de cada vaga de estacionamento. Eles são conectados ao ESP32, que controla sua ativação conforme os dados recebidos do Firebase.

Cada vaga possui três LEDs de cores diferentes (verde, vermelho e amarelo), e cada cor representa uma condição específica:

Verde: Vaga livre e disponível para reserva.

Amarelo: Vaga reservada por um utilizador autenticado. Vermelho: Vaga ocupada por um veículo estacionado.

O controle das cores é feito através das saídas digitais do ESP32, que enviam sinais elétricos para acender o LED correspondente. Dessa forma, sempre que o estado de uma vaga é alterado no banco de dados, o microcontrolador atualiza automaticamente a cor exibida na maquete.

Além da função visual, o uso dos LEDs contribui para o entendimento rápido e intuitivo do funcionamento do sistema, permitindo que qualquer observador identifique o status das vagas sem a necessidade de consultar o site. Essa sinalização direta torna a maquete mais interativa e próxima de um ambiente real de estacionamento inteligente.

3.1.4 *Buzzer*

O *buzzer* é um pequeno dispositivo sonoro utilizado para emitir alertas no sistema ParkEasy. Ele converte sinais elétricos enviados pelo ESP32 em vibrações sonoras, funcionando como uma forma simples de notificação auditiva.

No projeto, o *buzzer* é acionado quando ocorre alguma condição específica, como

a ocupação de uma vaga que está reservada sem que o utilizador autorizado tenha realizado o *check-in*. Nesses casos, o microcontrolador envia um sinal digital para o pino de saída conectado ao *buzzer*, produzindo um som curto e contínuo que indica a irregularidade.

O componente é do tipo ativo, ou seja, já possui um oscilador interno, necessitando apenas de um sinal de nível lógico para funcionar. Essa característica simplifica a ligação elétrica e reduz a necessidade de programação adicional.

O uso do *buzzer* complementa os LEDs de sinalização, adicionando uma camada sonora de aviso, que facilita a identificação de eventos importantes e aumenta a interatividade do sistema.

3.2 Nuvem (*Google Firebase*)

A nuvem é o elo entre o *hardware* e o *software* do sistema ParkEasy, sendo responsável por armazenar e sincronizar as informações em tempo real. Para isso, foi utilizada a plataforma *Google Firebase*, que oferece serviços integrados de banco de dados e autenticação.

O *Firebase Realtime Database* funciona como o banco de dados central, guardando o estado de cada vaga (livre, reservada ou ocupada) e os dados das reservas, como ID do utilizador e tempo de expiração. Sua principal vantagem é a atualização em tempo real, permitindo que qualquer mudança feita pelo ESP32 ou pelo site seja refletida instantaneamente.

O *Firebase Authentication* gerencia o cadastro e login dos utilizadores por e-mail e senha, garantindo que apenas usuários autenticados possam interagir com o sistema.

A comunicação entre o ESP32, o Firebase e o site ocorre via internet segura (HTTPS). Assim, sempre que o microcontrolador detecta uma alteração, o banco é atualizado e o site exibe a mudança automaticamente.

O uso do *Firebase* simplificou o desenvolvimento, dispensando um servidor próprio e tornando o sistema mais prático, estável e eficiente.

3.2.1 *Firebase Realtime Database*

O *Firebase Realtime Database* é o serviço do Google *Firebase* responsável por armazenar e sincronizar os dados do sistema ParkEasy em tempo real. Ele atua como o

banco de dados central, mantendo todas as informações atualizadas e acessíveis tanto para o ESP32 quanto para o site.

Cada vaga de estacionamento possui um nó individual no banco, que contém dados como o estado atual (livre, ocupada ou reservada), o ID do utilizador que fez a reserva e o tempo de expiração.

A principal característica desse serviço é a atualização instantânea: sempre que o ESP32 envia uma nova informação, por exemplo, uma vaga que acabou de ser ocupada , o *Realtime Database* notifica automaticamente todos os dispositivos conectados. Isso permite que o site e o hardware estejam sempre sincronizados sem a necessidade de recarregar a página ou executar comandos adicionais.

Além disso, o *Firebase* utiliza o formato JSON (*JavaScript Object Notation*) para organizar os dados, facilitando a comunicação com o código *JavaScript* do site e com o *firmware* do microcontrolador.

Esse recurso garante rapidez, estabilidade e simplicidade na troca de informações, sendo essencial para o funcionamento dinâmico do ParkEasy.

3.2.2 *Firebase Authentication*

O *Firebase Authentication* é o serviço responsável pelo controle de acesso e autenticação de utilizadores no sistema ParkEasy. Ele permite que apenas pessoas registradas e devidamente identificadas possam interagir com as funções principais, como reservar ou cancelar vagas.

A autenticação é feita por meio de e-mail e senha, garantindo praticidade e segurança ao utilizador. Durante o cadastro, o serviço cria um identificador único (UID) para cada conta, que é utilizado para associar reservas e ações específicas ao respetivo utilizador no banco de dados.

Esse sistema também mantém a sessão ativa, permitindo que o utilizador permaneça logado mesmo após atualizar a página, até que opte por sair manualmente.

O *Firebase Authentication* foi escolhido por sua facilidade de integração com o *Realtime Database* e pelo suporte direto a aplicações *JavaScript*, o que simplifica o código do site e reduz o tempo de desenvolvimento.

Com essa ferramenta, o ParkEasy garante segurança, confiabilidade e controle de

acesso, essenciais para o bom funcionamento do sistema e para a integridade dos dados armazenados na nuvem.

3.3 *Software* (Site)

O *software* do sistema ParkEasy consiste na interface web que permite ao utilizador interagir com o estacionamento inteligente de forma intuitiva. Desenvolvido em HTML, CSS e *JavaScript*, o site é responsável por exibir o estado das vagas em tempo real, permitir reservas, *check-ins* e controlar a experiência do usuário.

Estrutura do Site:

HTML: Estrutura todas as páginas do site, incluindo *Login*, Cadastro, Mapa de Vagas e Informações do Sistema.

CSS: Define a identidade visual, com cores específicas para os LEDs (verde, amarelo e vermelho) e modo escuro para facilitar a leitura.

JavaScript: Implementa a lógica do site, incluindo:

- Conexão em tempo real com o *Firebase Realtime Database*;
- Atualização dinâmica da interface conforme mudanças no estado das vagas;
- Controle de reservas, cancelamentos e *check-ins*;
- Proteção de páginas, garantindo acesso apenas a usuários autenticados;
- Temporizadores de contagem regressiva para reservas expiradas.

Integração com a Nuvem:

O site permanece constantemente conectado ao *Firebase*, “ouvindo” alterações no banco de dados e refletindo as mudanças na interface sem a necessidade de atualização manual. Quando um usuário realiza uma ação, como reservar uma vaga, o site envia os comandos diretamente para o Firebase, que por sua vez atualiza o ESP32 e os LEDs físicos.

Funcionalidade e Usabilidade:

O ParkEasy apresenta uma plataforma web projetada para ser altamente intuitiva, eficiente e totalmente compatível com dispositivos móveis. Através dela, os utilizadores têm acesso a uma visão dinâmica do mapa de vagas, atualizada em tempo real,

permitindo-lhes reservar um lugar de forma ágil e receber *feedback* imediato sobre o estado da sua reserva e do estacionamento. A combinação de uma interface de utilizador límpida com a integração robusta e direta aos serviços de nuvem resulta num sistema que opera com fiabilidade e conveniência, consolidando *hardware*, *software* e *cloud* (Nuvem) numa solução tecnológica unificada.

3.3.1 HTML

O HTML (*HyperText Markup Language*) é a linguagem utilizada para estruturar o conteúdo do site do sistema ParkEasy. Ele define a organização de todas as páginas, determinando a disposição de textos, botões, campos de formulário e o mapa de vagas do estacionamento.

Cada página do site — incluindo *Login*, Cadastro, Mapa de Vagas e Informações do Sistema — é construída com elementos HTML, como divs, *sections*, *buttons*, *inputs* e *tables*, permitindo que o conteúdo seja corretamente exibido em navegadores web.

O HTML funciona como a base estrutural do site, garantindo que o layout seja consistente, que os elementos interativos possam ser manipulados pelo *JavaScript* e que o CSS possa aplicar estilos e cores de forma adequada.

Essa camada é essencial para a interatividade e acessibilidade, proporcionando que todos os componentes visuais e funcionais do ParkEasy sejam exibidos corretamente aos utilizadores. (ANOTAÇÕES DE AULA, 2024).

3.3.2 CSS

O CSS (*Cascading Style Sheets*) é a linguagem utilizada para definir a identidade visual e o estilo do site do sistema ParkEasy. Ele é responsável por controlar cores, fontes, tamanhos, espaçamentos e layout geral das páginas, garantindo uma interface clara e agradável para o usuário.

No ParkEasy, o CSS é usado para:

- Aplicar o modo escuro ao site, melhorando a visualização;
- Destacar os LEDs virtuais das vagas com cores correspondentes aos estados (verde, amarelo e vermelho);
- Organizar os elementos da página de forma responsiva, adaptando o layout

para diferentes tamanhos de tela;

- Estilizar botões, formulários e temporizadores de forma consistente com a identidade visual do sistema.

O CSS trabalha em conjunto com o HTML, que fornece a estrutura, e o *JavaScript*, que controla a lógica do site. Essa separação entre estrutura, estilo e comportamento permite que o site seja manutenível, escalável e visualmente intuitivo para os utilizadores. (ANOTAÇÕES DE AULA, 2024).

3.3.3 *JavaScript*

O *JavaScript* é a linguagem de programação responsável por adicionar interatividade e lógica ao site do sistema ParkEasy. Ele permite que a interface web responda a ações do usuário e se comunique em tempo real com o *Firebase* e o ESP32.

No ParkEasy, o *JavaScript* é utilizado para:

- Conectar o site ao *Firebase Realtime Database*, permitindo atualização instantânea do estado das vagas;
- Controlar *login* e cadastro de usuários, garantindo que apenas pessoas autenticadas possam realizar reservas e *check-ins*;
- Gerenciar reservas, cancelamentos e *check-ins*, enviando as informações ao *Firebase* e recebendo o status atualizado do *hardware*;
- Atualizar dinamicamente a interface do usuário, alterando cores dos LEDs virtuais e o estado dos botões conforme mudanças nas vagas;
- Implementar temporizadores de contagem regressiva para reservas, alertando os usuários sobre expiração;
- Proteger páginas e funcionalidades, garantindo que apenas usuários logados tenham acesso às funções restritas do sistema.

Dessa forma, o *JavaScript* integra a estrutura (HTML) e o estilo (CSS) do site com a camada de nuvem e *hardware*, tornando o ParkEasy interativo, responsivo e confiável para os usuários. (ANOTAÇÕES DE AULA, 2025).

4. FLUXO DE FUNCIONAMENTO

Esta seção detalha a jornada do utilizador ao interagir com o sistema ParkEasy, desde o acesso inicial até a utilização completa das funcionalidades de monitorização e reserva de vagas. O fluxo foi projetado para ser intuitivo e eficiente, guiando o condutor através das etapas necessárias.

4.1 Cadastro e *Login* do Utilizador

O acesso às funcionalidades interativas do ParkEasy, como a reserva de vagas, requer autenticação prévia do utilizador para garantir a segurança e a personalização da experiência.

- Acesso Inicial: O utilizador acede ao sistema através da página *login*.html.
- Opção de Cadastro/*Login*: A página apresenta dois formulários distintos: um para utilizadores existentes ("Entrar na sua Conta") e outro para novos utilizadores ("Criar Nova Conta"). Links permitem alternar facilmente entre os dois formulários.
- Cadastro: Um novo utilizador preenche os campos de nome, *e-mail* e senha. Ao clicar em "Cadastrar", o *script login*.js utiliza a função *createUserWithEmailAndPassword* do *Firebase Authentication* para criar a conta. O nome do utilizador é guardado no perfil através da função *updateProfile*.
- *Login*: Um utilizador existente preenche o seu *e-mail* e senha. Ao clicar em "Entrar", o *script login*.js utiliza a função *signInWithEmailAndPassword* do *Firebase Authentication* para validar as credenciais.
- Redirecionamento: Em caso de sucesso no cadastro ou no login, o utilizador é automaticamente redirecionado para a página principal de monitorização (vagas.html). Mensagens de erro são exibidas na própria página caso as credenciais sejam inválidas ou ocorra outro problema.

FIGURA 4 – TELA DE LOGIN DO SITE

FONTE: Acervo pessoal dos autores(2025).

4.2 Visualização e Navegação no Mapa de Vagas

Após a autenticação bem-sucedida, o utilizador é direcionado para a página vagas.html, a interface central do ParkEasy. Primeiramente, ao carregar a página, o *script*.js principal verifica se há um utilizador ativo através do *onAuthStateChanged*. Caso contrário, redireciona imediatamente para *login*.html, protegendo assim o acesso não autorizado. Uma vez validado, a página exibe uma representação visual das 4 vagas do estacionamento. A característica principal desta tela é a atualização em tempo real, possibilitada pela conexão persistente (*onValue*) com o *Firebase Realtime Database*. Qualquer alteração no estado das vagas (seja iniciada pelo *hardware* ou por outro utilizador) é refletida instantaneamente na interface, alterando a cor e o texto de cada vaga: Verde para Livre, Vermelho para Ocupada e Amarelo para Reservada ou Em Processo de Estacionamento. Adicionalmente, o cabeçalho apresenta informações do utilizador logado ("Olá, [Nome]!") e um botão "Sair", enquanto um contador informa o número de vagas disponíveis no momento.

4.3 Processo de Reserva de Vaga

Utilizadores autenticados podem reservar uma vaga livre antecipadamente, garantindo sua disponibilidade por um tempo limitado. O processo inicia-se quando o utilizador identifica uma vaga livre (verde) no mapa e clica no botão "Reservar" correspondente. Esta ação dispara uma função no script.js que envia um comando para o *Firebase Realtime Database*, atualizando o registo daquela vaga específica com o status "reservada", o ID único do utilizador (reservadoPor) e um timestamp de expiração (tempoExpiracao) definido para 3 minutos no futuro. Imediatamente, a interface do site reage, mudando a cor da vaga para amarelo, atualizando o texto para "RESERVADA" e substituindo o botão "Reservar" pelos botões "Estacionar Agora" e "Cancelar". Um temporizador de contagem regressiva começa a ser exibido no botão "Estacionar Agora". Simultaneamente, o ESP32, que monitora constantemente o *Firebase*, detecta esta alteração e inicia a sinalização física na maquete, fazendo o LED amarelo da vaga piscar. O sistema também inclui uma lógica de cancelamento automático: se os 3 minutos expirarem sem que o utilizador faça o *check-in* ou cancele a reserva, tanto o *script* do site quanto o *firmware* do ESP32 estão programados para reverter o estado da vaga para "LIVRE" no *Firebase*.

4.4 *Check-in* e Estacionamento

Ao chegar fisicamente à vaga que reservou, o utilizador deve realizar o processo de *check-in* através do site para sinalizar sua intenção de estacionar e evitar que o sistema interprete sua chegada como uma invasão. Este passo é iniciado clicando no botão "Estacionar Agora", que exibe a contagem regressiva da reserva. A ação no botão faz com que o *script*.js atualize novamente o registo da vaga no *Firebase*, mudando o status para "estacionando" e definindo um novo tempoExpiracao de 60 segundos (ou utilizando o campo tempoCheckinExpira), mantendo a informação de quem reservou (reservadoPor). Visualmente, o site atualiza o texto da vaga para "ESTACIONANDO", inicia um novo temporizador de 60 segundos (geralmente exibido dentro do bloco da vaga) e mantém o botão "Cancelar" visível. O ESP32, por sua vez, continua a piscar o LED amarelo, mas agora sua lógica interna está ciente de que o utilizador está autorizado a entrar nos próximos 60 segundos. Caso este tempo expire antes de o sensor detetar o veículo, a reserva é automaticamente cancelada por ambos os sistemas (site e ESP32), e a vaga volta ao estado "LIVRE".

4.5 Ocupação e Liberação da Vaga

A detecção da presença ou ausência física do veículo é realizada pelo *hardware* e sincronizada com o sistema. Quando um veículo entra na vaga, o Sensor Ultrassônico regista uma distância inferior ao limite pré-definido. O ESP32 deteta esta mudança em relação ao estado anterior e, antes de agir, verifica o status atual da vaga no *Firebase*. Se o status for "reservada", o sistema entende que ocorreu uma ocupação indevida (sem *check-in*) e dispara o alarme sonoro (*Buzzer*). Se o status for "estacionando", o sistema reconhece a chegada do utilizador legítimo e não dispara o alarme. Independentemente do estado prévio, a deteção física do carro leva o ESP32 a enviar a atualização status: *true* para o *Firebase*. Esta mudança é então propagada para o site, que exibe a vaga como "OCUPADA" (vermelho), e para o próprio ESP32, que acende o LED vermelho. O processo inverso ocorre quando o veículo sai: o sensor deteta a ausência (distância maior que o limite), o ESP32 percebe a mudança e envia um objeto JSON para o *Firebase* que redefine a vaga para o estado livre ({ *status*: *false*, reservadoPor: *null*, ... }). Esta atualização é novamente refletida no site (vaga verde, "LIVRE") e no *hardware* (LED verde aceso).

4.6 Sinalização e Atualizações em Tempo Real

A eficácia do ParkEasy reside na sua capacidade de manter todas as partes do sistema sincronizadas instantaneamente. O *Firebase Realtime Database* funciona como a "única fonte da verdade", centralizando o estado de todas as vagas. A comunicação é bidirecional: o site atualiza o *Firebase* com base nas ações do utilizador (reservar, cancelar, *check-in*), enquanto o ESP32 atualiza o *Firebase* com base nas leituras do sensor (ocupar, liberar). Graças à natureza em tempo real do *Firebase*, qualquer alteração num dos lados é imediatamente notificada aos outros clientes conectados. O utilizador recebe, assim, um *feedback* multimodal constante: visualmente no site (cores, textos, botões, temporizadores), visualmente na maquete (LEDs verde, vermelho, amarelo) e auditivamente na maquete (alarme do *buzzer*). Esta arquitetura garante uma experiência de utilizador coesa, onde a interface digital reflete fielmente o estado físico do estacionamento e vice-versa.

5. DESENVOLVIMENTO E DESAFIOS TÉCNICOS

O desenvolvimento do projeto ParkEasy, apesar de conceitualmente direto, apresentou diversos desafios técnicos inerentes à integração de múltiplas tecnologias (*hardware* embarcado, serviços de nuvem e *frontend web*) e à natureza assíncrona da comunicação em tempo real. Esta seção detalha os principais obstáculos encontrados e as soluções implementadas para superá-los.

5.1 Desafios Encontrados Durante o Desenvolvimento

Desde o início, a complexidade da arquitetura distribuída do ParkEasy trouxe desafios. A necessidade de manter a coerência dos dados entre o estado físico da vaga (lido pelo ESP32), o estado armazenado na nuvem (*Firebase Realtime Database*) e o estado exibido na interface do utilizador (site) exigiu um planeamento cuidadoso da lógica de comunicação e sincronização. A escolha inicial da plataforma *Firebase*, embora poderosa, também introduziu desafios específicos, como a gestão de diferentes bibliotecas e métodos de autenticação.

Outro desafio significativo foi a depuração de problemas que ocorriam apenas na interação entre as camadas. Um bug no *firmware* do ESP32, por exemplo, poderia manifestar- se como um comportamento inesperado no site, exigindo uma análise cuidadosa do fluxo de dados através do *Firebase* para identificar a causa raiz. A natureza assíncrona das operações (leituras de sensor, comunicação Wi-Fi, atualizações do *Firebase*, redesenho da interface) também criou potenciais "condições de corrida" (*race conditions*), onde a ordem inesperada de execução de certas tarefas levava a estados inconsistentes no sistema. Além disso, a gestão das bibliotecas no ambiente de desenvolvimento da Arduino IDE provou ser uma fonte recorrente de problemas inesperados.

5.2 Solução para o *Bug* do *Login* com Google

A implementação da autenticação de utilizadores foi um dos pontos mais desafiantes. A escolha inicial pelo *Login* com Google, utilizando o *Firebase Authentication*, parecia a opção mais conveniente para o utilizador final. No entanto, esta abordagem apresentou um bug persistente e de difícil diagnóstico no ambiente de desenvolvimento local (utilizando o *Live Server* no *VS Code*). Ao clicar no botão de

login, a janela *pop-up* do Google abria e fechava instantaneamente ("piscada"), ou, ao utilizar o método de redirecionamento, o utilizador era autenticado com sucesso no Google, mas ao retornar para a aplicação, o estado de login não era reconhecido pelo *script*.

Foram realizadas inúmeras tentativas de solução, incluindo a verificação e adição dos domínios localhost e 127.0.0.1 à lista de domínios autorizados no *Firebase Authentication*, a configuração detalhada da Tela de Consentimento O *Auth* no Google *Cloud Platform* (nome da aplicação, *e-mail* de suporte, publicação do estado de "teste" para "produção") e a experimentação com diferentes métodos de persistência da sessão de login no *Firebase* (*browserSessionPersistence*). Apesar de todos os esforços e da aparente correção das configurações, o problema persistiu no ambiente local. A única forma de validar o funcionamento foi publicando o site no *GitHub Pages*, o que tornava o ciclo de desenvolvimento impraticável. Diante deste impasse, foi tomada a decisão estratégica de pivotar a solução de autenticação. Abandonou-se o *Login* com Google e optou-se pela implementação do sistema nativo de **Login com *E-mail* e Senha** do *Firebase Authentication*. Esta abordagem, por não depender de *pop-ups* ou redirecionamentos complexos, funcionou de forma robusta e imediata no ambiente de desenvolvimento local, resolvendo o bloqueio e permitindo a continuidade do projeto.

5.3 Problemas de Sincronia (*Race Conditions*) e Resolução

Um desafio recorrente foi garantir que a interface do utilizador refletisse sempre o estado mais atualizado, tanto do login quanto das vagas. Observou-se que, por vezes, especialmente após o redirecionamento do *login* (na fase em que ainda se tentava o *Login* com Google) ou durante carregamentos rápidos da página, a interface das vagas era renderizada antes que o estado de autenticação do utilizador fosse completamente estabelecido pelo *Firebase* (*onAuthStateChanged*). Isto resultava numa experiência inconsistente, onde um utilizador logado via a página como se estivesse deslogado (sem os botões de interação).

Para resolver esta "condição de corrida", a arquitetura do *script*.js principal (da página vagas.html) foi refatorada significativamente. Implementou-se uma lógica de inicialização mais rigorosa e sincronizada. O observador do estado de autenticação (*onAuthStateChanged*) tornou-se o ponto central e a única porta de entrada para a renderização da interface principal. Somente após a confirmação do estado do utilizador

(logado ou não), a função responsável por iniciar a escuta dos dados das vagas no *Firebase* (*listenToVagas*) era chamada. Além disso, criou-se uma única função de renderização (*renderVagas*), que passou a ser responsável por desenhar toda a interface das vagas com base nas informações mais recentes, tanto do utilizador (*currentUser*) quanto dos dados do *Firebase* (*vagasData*), garantindo que a informação de *login* estivesse sempre disponível no momento do desenho. Esta abordagem eliminou a competição entre diferentes partes do código para atualizar a UI, garantindo que a renderização ocorresse sempre com o estado correto e mais atualizado disponível.

5.4 Erro nos Botões e Temporizadores

Após diversas refatorações no *script*.js para corrigir os problemas de *login* e sincronia, surgiu um novo *bug* intermitente: os botões de interação com as vagas ("Reservar", "Estacionar Agora", "Cancelar") por vezes paravam de responder aos cliques. Verificou-se que a forma como os "ouvintes" de eventos (*addEventListener*) estavam a ser adicionados aos botões individualmente era frágil e quebrava quando a interface era redesenhada dinamicamente pela função *renderVagas* (que precisava limpar e recriar elementos, como os temporizadores). A lógica complexa dos temporizadores visuais, que dependia da leitura correta do tempoExpiracao do *Firebase* e da gestão dos intervalos (*setInterval*), também se tornou instável e deixou de funcionar corretamente em alguns cenários.

O problema dos botões foi resolvido de forma definitiva através da implementação da técnica de **delegação de eventos**. Em vez de adicionar um "ouvinte" a cada um dos 12 botões, um único *addEventListener* foi adicionado ao elemento pai que contém todas as vagas (*containerVagas*). Este "ouvinte" captura todos os cliques dentro do contentor, verifica eficientemente se o alvo do clique foi um botão com um ID esperado (ex: btn-reservar-1) e, só então, extrai a ação ("reservar") e o número da vaga ("1") do ID para chamar a função correspondente. Esta abordagem é mais robusta, eficiente e imune a problemas causados pela recriação dinâmica dos elementos. A lógica dos temporizadores foi cuidadosamente reintegrada à função *renderVagas*, garantindo que eles fossem corretamente iniciados (*startTimer*), atualizados e, crucialmente, limpos (*clearInterval*) a cada redesenho da interface, utilizando o tempoExpiracao lido do *Firebase* como base para a contagem regressiva e evitando "vazamentos" de intervalos.

5.5 Soluções Implementadas

Ao longo do desenvolvimento do ParkEasy, diversos desafios técnicos surgiram, exigindo diagnósticos precisos e a implementação de soluções robustas e, por vezes, a reavaliação de decisões de arquitetura. A instabilidade inicial do *Login* com Google no ambiente de desenvolvimento local foi contornada pela adoção estratégica do sistema de *E-mail*/Senha do *Firebase Authentication*, que se provou mais fiável para testes locais. Problemas de sincronia e "condições de corrida" na atualização da interface web foram resolvidos refatorando o *script*.js para garantir uma ordem de execução controlada (*login*-primeiro) e centralizando a lógica de renderização numa única função. A falha intermitente nos botões de interação foi solucionada com a implementação da delegação de eventos, uma técnica mais resiliente a atualizações dinâmicas do DOM. Conflitos de bibliotecas do *Firebase* no ambiente Arduino IDE foram superados através de uma limpeza completa do ambiente e da reinstalação controlada da biblioteca correta (*Firebase_ESP_Client*), seguida pela adaptação do código *firmware* para garantir a compatibilidade. Por fim, a instabilidade ocasional da conexão do ESP32 com o *Firebase* foi mitigada pela inclusão de um certificado SSL no *firmware* e pela implementação de lógicas de tratamento de erro e reconexão automática no *loop()* principal. Cada desafio superado, embora frustrante no momento, contribuiu significativamente para o aprendizado da equipa e para a criação de um sistema final

6. SUGESTÕES PARA TRABALHOS FUTUROS

O protótipo do ParkEasy, embora funcional e abrangente, representa a base sobre a qual diversas funcionalidades e melhorias podem ser construídas, transformando-o de um sistema de monitorização e reserva num ecossistema completo de gestão de estacionamento inteligente. As sugestões apresentadas a seguir visam elevar o nível técnico e comercial do projeto, abordando áreas como análise de dados, automação avançada e integração com sistemas de monetização.

6.1 *Dashboard* de Análise

Uma evolução natural e de grande valor seria a implementação de um *dashboard* de análise de dados. Atualmente, o sistema monitoriza o estado das vagas em tempo real, mas não armazena um histórico detalhado da sua utilização. A criação de um módulo de *logging* permitiria registar eventos importantes, como a ocupação e a libertação de cada vaga, associados a um timestamp e, potencialmente, ao ID do utilizador (para vagas reservadas).

Com este histórico armazenado (preferencialmente numa estrutura otimizada no *Firebase* ou outro serviço de banco de dados), uma nova página no site (*dashboard.html*) poderia consumir esses dados e, através de bibliotecas de visualização como a *Chart.js*, apresentar gráficos e métricas relevantes para o gestor do estacionamento. Exemplos incluem gráficos de taxa de ocupação ao longo do dia (identificando horários de pico), mapas de calor mostrando as vagas mais e menos utilizadas, tempo médio de permanência dos veículos e relatórios sobre a eficácia do sistema de reservas. Esta funcionalidade transformaria o ParkEasy numa ferramenta de *Business Intelligence*, fornecendo *insights* valiosos para a otimização da operação do estacionamento.

6.2 Integração com Câmeras e Reconhecimento de Placas (LPR/ANPR)

Considerando a inviabilidade de barreiras físicas individuais em larga escala (como num shopping), a evolução mais impactante e profissional para garantir a segurança das reservas e eliminar a necessidade do "*check-in*" manual pelo utilizador seria a integração com um sistema de **Leitura Automática de Placas de Matrícula** (LPR/ANPR - *Automatic Number Plate Recognition*). Esta tecnologia substituiria ou complementaria o sensor ultrassónico como método primário de deteção e identificação.

Neste cenário, cada vaga ou grupo de vagas seria monitorizado por uma câmera. Ao detetar a entrada de um veículo (seja pelo sensor ultrassónico ou por análise de vídeo), a câmera capturaria a imagem da placa. Um *software* de OCR (Reconhecimento Óptico de Caracteres), que poderia rodar num servidor local, no próprio ESP32-CAM (para protótipos) ou num serviço de nuvem especializado, extrairia o texto da placa. O sistema então compararia a placa lida com a placa associada à reserva (que o utilizador teria de cadastrar previamente no seu perfil). Se as placas coincidirem, a ocupação é confirmada automaticamente, eliminando a necessidade de o utilizador interagir com o site ("Estacionar Agora"). Se uma placa diferente ocupar uma vaga reservada, o sistema identificaria a infração, dispararia o alarme e poderia acionar os procedimentos de segurança ou multa definidos pelo estabelecimento. Esta integração representa o estado da arte em gestão de estacionamentos, oferecendo segurança, automação e uma experiência de utilizador significativamente mais fluida.

6.3 Integração com Sistema de Pagamento

Para transformar o ParkEasy num modelo de negócio viável, a integração com um sistema de pagamento é fundamental. Uma abordagem seria implementar uma taxa de conveniência cobrada no momento da reserva da vaga. Isto exigiria a integração do *frontend web* com um *gateway* de pagamento (como *Stripe* ou Mercado Pago), utilizando as suas APIs e ambientes de teste (*sandbox*) para simular transações.

O fluxo seria modificado: ao clicar em "Reservar", o utilizador seria direcionado para o *checkout* seguro do *gateway* para efetuar o pagamento da taxa. Somente após a confirmação do pagamento bem-sucedido (recebida pelo site através de um *webhook* ou *callback*), o sistema procederia com a atualização do estado da vaga no *Firebase* para "reservada". Para maior segurança, a validação final da transação e a comunicação com a API do *gateway* poderiam ser intermediadas por *Firebase Cloud Functions*, atuando como um *backend* seguro. Esta funcionalidade não só abriria caminhos para a monetização do serviço, mas também demonstraria competências avançadas em integração de sistemas e desenvolvimento *web full- stack*.

7. CONSIDERAÇÕES FINAIS

Esta seção consolida as expectativas do projeto e reflete sobre o processo de desenvolvimento, concluindo com as principais aprendizagens e recomendações.

7.1 Resultados Esperados

Espera-se que o protótipo do ParkEasy demonstre com sucesso a viabilidade técnica da arquitetura proposta. O sistema deverá ser capaz de monitorizar o estado físico das vagas através dos sensores ultrassónicos, comunicar essa informação em tempo real para o *Firebase Realtime Database* e refletir o estado atualizado na interface web. A funcionalidade de autenticação de utilizadores com *E-mail*/Senha deve garantir o acesso seguro ao sistema. As operações de reserva, *check-in* e cancelamento deverão funcionar conforme o fluxo descrito, corretamente o *Firebase* e a interface do utilizador, incluindo a gestão dos temporizadores de expiração. O *hardware* (ESP32) deverá controlar os LEDs de sinalização (verde, vermelho, amarelo) de acordo com o estado lido do *Firebase* e acionar o alarme sonoro em caso de ocupação indevida de uma vaga reservada. Espera-se que a integração entre as três camadas (*hardware*, nuvem, *software*) seja fluida e responsiva, validando o conceito de um sistema de estacionamento inteligente e conectado.

7.2 Conclusão e Recomendações

O desenvolvimento do ParkEasy, apesar dos desafios técnicos encontrados, culminou num protótipo funcional que atinge os objetivos propostos. A integração bem-sucedida do ESP32 com o *Firebase* e uma interface web interativa demonstra a aplicação prática de conceitos de IoT, programação embarcada e desenvolvimento web. A superação de obstáculos, como os bugs de autenticação e sincronia, reforçou a importância de metodologias de depuração sistemática e da escolha de arquiteturas resilientes. O projeto valida a eficácia de utilizar serviços de *Backend-as-a-Service* (BaaS) como o *Firebase* para acelerar o desenvolvimento de aplicações conectadas. Recomenda-se, para futuras iterações, a implementação das melhorias detalhadas na seção anterior (*Dashboard*, LPR, Pagamentos) para aproximar o protótipo de uma solução comercial completa. Conclui-se que o ParkEasy serve como um excelente estudo de caso e uma base sólida para explorações futuras no domínio de cidades inteligentes e sistemas de gestão de estacionamento.

8. REFERÊNCIAS BIBLIOGRÁFICAS

EMBARCADOS. *Ambiente ESP32 no Windows.* Disponível em: https://embarcados.com.br/ambiente-esp32-no-windows/. Acesso em: 10 out. 2025.

ISAAC. *Getting started with the HC-SR04 Ultrasonic Sensor.* ProjectHub Arduino, 2024. Disponível em: https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1. Acesso em: 10 out. 2025.

MECÂNICA INDUSTRIAL. *O que é um sensor ultrassônico?* Disponível em: https://www.mecanicaindustrial.com.br/598-o-que-e-um-sensor-ultrassonico/. Acesso em: 10 out. 2025.

BOTSMAN, R.; ROGERS, R. O que é meu é seu: o surgimento do consumo colaborativo. Disponível em: https://pt.scribd.com/document/299477107/o-Que-e-Meu-e-Seu. Acesso em: 12 maio 2025.

CONFEDERAÇÃO NACIONAL DO TRANSPORTE (CNT). Pesquisa CNT de Mobilidade Urbana é apresentada ao Consetrans. Disponível em: https://cnt.org.br/agencia-cnt/pesquisa-cnt-de-mobilidade-urbana--apresentada-ao-consetrans. Acesso em: 4 jul. 2025.

ANGELIDOU, M. Smart cities: A conjuncture of four forces. Scientific Research Publishing, 2015. Disponível em: https://www.scirp.org/reference/referencespapers?referenceid=3161771. Acesso em: 23 ago. 2025.

MOZILLA DEVELOPER NETWORK. Adding interactivity. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Getting_started/Your_first_website/Adding_interactivity. Acesso em: 10 out. 2025.

OPENAI. ChatGPT. Disponível em: https://chatgpt.com. Acesso em: 10 out. 2025.

MOZILLA DEVELOPER NETWORK. What is JavaScript. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Core/Scripting/What_is_JavaScript. Acesso em: 10 out. 2025.

ALURA. JavaScript: o que é e como funciona. Disponível em: https://www.alura.com.br/artigos/javascript?srsltid=AfmBOopyPnuSBv3gsyvvXoZNa7nBUTiFwif6YsuinyizBYSEuEeuaMqL. Acesso em: 10 out. 2025.

MOZILLA DEVELOPER NETWORK. Styling the content. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Getting_started/Your_first_website/Styling_the_content. Acesso em: 10 out. 2025.

EBAC. O que é CSS e para que serve. Disponível em: https://ebaconline.com.br/blog/o-que-e-css-e-para-que-serve-seo. Acesso em: 10 out. 2025.

DEVMEDIA. Guia CSS. Disponível em: https://www.devmedia.com.br/guia/css/38149. Acesso em: 10 out. 2025.

MOZILLA DEVELOPER NETWORK. Creating the content. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Getting_started/Your_first_website/Creating_the_content. Acesso em: 10 out. 2025.

BORGES, L. Testar PHP. Disponível em: https://www.lncc.br/~borges/php/testar.html. Acesso em: 10 out. 2025.

BYND. As vagas de estacionamento estão difíceis? Saiba como se livrar desse problema. Disponível em: https://bynd.com.br/2019/07/as-vagas-de-estacionamento-estao-dificeis-saiba-como-se-livrar-desse-problema/. Acesso em: 10 out. 2025.

PORTAL DA INDÚSTRIA. 36% dos brasileiros de grandes cidades passam mais de 1 hora por dia no trânsito. Disponível em: https://noticias.portaldaindustria.com.br/noticias/infraestrutura/36-dos-brasileiros-de-grandes-cidades-passam-mais-de-1-hora-por-dia-no-transito/. Acesso em: 10 out. 2025.

DIÁRIO DE SUZANO. Motoristas reclamam da falta de vagas de estacionamentos. Disponível em: https://www.diariodesuzano.com.br/cidades/motoristas-reclamam-da-falta-de-vagas-de-estacionamentos/48031/. Acesso em: 10 out. 2025.

GOOGLE FIREBASE. Authentication documentation. Disponível em: https://firebase.google.com/docs/auth?hl=pt-br. Acesso em: 10 out. 2025.