CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
ESCOLA TÉCNICA ESTADUAL DE CUBATÃO
CURSO DE ENSINO TÉCNICO EM INFORMÁTICA

Beatriz Araújo da Silva Lima
Jorge Manoel Domingues Filho

SGVE – DESENVOLVIMENTO DE UM SISTEMA DE GESTÃO DE VAGAS DE ESTACIONAMENTO COM MONITORAMENTO DE OCUPAÇÃO

Cubatão
2026

Beatriz Araújo da Silva Lima
Jorge Manoel Domingues Filho

SGVE – DESENVOLVIMENTO DE UM SISTEMA DE GESTÃO DE VAGAS DE ESTACIONAMENTO COM MONITORAMENTO DE OCUPAÇÃO

Relatório Técnico apresentado como Trabalho de Conclusão de Curso na Escola Técnica de Cubatão, no Curso de Técnico em Informática, como exigência parcial para obtenção do título de Técnico em Informática.

Orientador: Prof. Marcelo Batista Onuki

Cubatão
2026

RESUMO

O presente trabalho tem como objetivo desenvolver um protótipo funcional de um Sistema de Gestão de Vagas de Estacionamento (SGVE) baseado no microcontrolador ESP32, capaz de monitorar e exibir a disponibilidade de vagas de estacionamento em tempo real. A problemática parte da constatação de que a ausência de informações atualizadas sobre vagas disponíveis contribui para congestionamentos, desperdício de combustível e insatisfação dos usuários. A metodologia adotada combina pesquisa bibliográfica sobre Smart Parking, Internet das Coisas e tecnologias web com coleta de dados quantitativa aplicada a usuários de estacionamentos, totalizando 37 respostas válidas. O sistema integra sensores infravermelhos conectados a um microcontrolador com conectividade Wi-Fi, que transmite os dados de ocupação a um servidor local responsável pelo processamento e armazenamento das informações. Uma interface web exibe o mapa de vagas e sua disponibilidade e um painel de métricas para administradores, permitindo o monitoramento e a gestão do estacionamento. O protótipo foi validado em maquete em escala reduzida, comprovando baixa latência e funcionamento eficiente em ambiente controlado. Os resultados confirmaram as hipóteses levantadas: a disponibilização de dados instantâneos reduziu a incerteza do usuário na busca por vagas, o monitoramento automatizado melhorou o controle gerencial e a solução demonstrou viabilidade técnica e econômica para ambientes de pequeno e médio porte. A pesquisa de validação indicou que 92,11% dos participantes acreditam que o sistema reduziria o tempo de busca e 86,84% concordam que melhoraria a fluidez interna, confirmando o alcance dos objetivos propostos.

Palavras-chave: Smart Parking. Internet das Coisas. ESP32. Estacionamento Inteligente.

Lista de figuras

Sumário

1 INTRODUÇÃO

A mobilidade urbana contemporânea enfrenta desafios crescentes, especialmente no que diz respeito à saturação das infraestruturas de estacionamento em ambientes privados. O tempo despendido por condutores na busca por vagas não apenas gera inconveniência individual, mas também compromete a fluidez do tráfego interno e a eficiência operacional dos estabelecimentos.

Diante desse cenário, surge a seguinte problemática: como desenvolver um sistema de gestão de vagas de estacionamento que otimize o controle de ocupação, reduza o tempo de procura e melhore a experiência dos usuários e gestores?

A ineficiência no controle de ocupação afeta diretamente a gestão de recursos e a satisfação do público, justificando a busca por soluções inteligentes e de baixo custo. O uso da plataforma ESP32 apresenta-se como alternativa viável para a criação de protótipos funcionais com custo reduzido.

Assume-se como hipótese que o uso de sistemas automatizados baseados em ESP32 é capaz de reduzir o congestionamento interno nos estacionamentos, apresentando viabilidade econômica para sua implementação.

O objetivo geral deste trabalho é desenvolver um protótipo funcional baseado em ESP32 capaz de monitorar e exibir a disponibilidade de vagas de estacionamento em tempo real.

Para alcançar esse objetivo, definem-se os seguintes objetivos específicos: integrar os componentes de hardware com sensores de presença; desenvolver uma interface visual e um backend para interação do usuário; e validar o sistema por meio de testes em maquete em escala reduzida.

A pesquisa bibliográfica constitui a base metodológica deste trabalho e será utilizada para fundamentar teoricamente o desenvolvimento do protótipo. Essa etapa justifica-se pela necessidade de compreender os conceitos relacionados à mobilidade urbana, sistemas embarcados e soluções de automação já existentes, tendo como base fontes acadêmicas, técnicas e trabalhos anteriores com temática similar.

A abordagem qualitativa e quantitativa será aplicada por meio de entrevistas e questionários com usuários e gestores de estacionamentos, com o objetivo de justificar a problemática identificada e levantar informações sobre as dificuldades enfrentadas no cotidiano. Esses dados permitirão compreender as expectativas de

ambos os perfis e identificar oportunidades de melhoria tanto na experiência do usuário que busca vagas quanto na rotina de quem gerencia o espaço.

2 REFERENCIAL TEÓRICO

O referencial teórico apresenta os conceitos fundamentais que embasam o desenvolvimento deste trabalho, abrangendo temas como mobilidade urbana, sistemas embarcados, automação e IoT. Sua utilização justifica-se pela necessidade de contextualizar as tecnologias empregadas no protótipo e demonstrar que a solução proposta está alinhada com estudos e pesquisas já consolidados na área.

2.1 GESTÃO DE ESTACIONAMENTOS

A gestão de estacionamentos fechados envolve o monitoramento contínuo das vagas, a consolidação das informações em uma central de controle e a disponibilização desses dados ao usuário de forma organizada. O gerenciamento não se limita à detecção de ocupação, mas integra controle, tomada de decisão e organização do fluxo interno. Segundo Mitsuhashi e Cavamura (2014), a utilização de sensores integrados a uma central permite que o gestor tenha visão precisa do estado das vagas, reduzindo tempo de busca e aumentando a eficiência operacional.

A ineficiência na busca por vagas pode ser compreendida a partir do conceito de tempo de procura (search time), entendido como o intervalo entre a entrada do veículo na área de estacionamento e o momento em que ele efetivamente ocupa uma vaga. O estudo de Briones (2021) destaca que esse tempo representa o indicador mais direto da experiência do usuário, pois reflete o esforço necessário para localizar uma vaga disponível. Diferentemente de abordagens tradicionais que avaliam apenas dimensões físicas ou capacidade estrutural do estacionamento, o tempo de busca evidencia o desempenho operacional real do sistema.

O autor também demonstra que o aumento da taxa de ocupação impacta diretamente esse tempo. Quando o nível de ocupação ultrapassa aproximadamente 85%, observa-se crescimento significativo no tempo médio de procura, o que caracteriza saturação da oferta e perda de eficiência. Esse cenário implica maior circulação interna de veículos, aumento do consumo de combustível, geração de congestionamento interno e elevação do estresse do usuário.

Sob essa perspectiva, a ineficiência na busca por vagas não se limita à ausência física de espaços disponíveis, mas está associada à dificuldade operacional de localizá-los em tempo adequado. Assim, o tempo de procura consolida-se como

variável central para avaliar o nível de serviço em estacionamentos, permitindo mensurar objetivamente o impacto da ocupação sobre a fluidez e a qualidade do sistema.

O desempenho operacional de estacionamentos fechados pode ser compreendido a partir da relação entre ocupação, tempo de procura e controle informacional. Sistemas que monitoram continuamente as vagas e disponibilizam dados em tempo real reduzem incerteza, minimizam circulação interna improdutiva e preservam a eficiência do fluxo. Assim, o impacto operacional não se restringe à disponibilidade física de vagas, mas à capacidade do sistema de organizar, informar e controlar o uso do espaço.

2.2 ESTACIONAMENTO INTELIGENTE (SMART PARKING)

O conceito de Smart Parking está inserido no contexto das cidades inteligentes e dos sistemas de transporte inteligentes, sendo fundamentado no uso de tecnologias como Internet das Coisas (IoT), redes de sensores e plataformas computacionais para monitoramento e gerenciamento de vagas em tempo real. De acordo com Fahim et al. (2021), os sistemas de estacionamento inteligente surgem como resposta ao problema da escassez de vagas e ao elevado tempo gasto na busca por estacionamento em áreas urbanas densamente povoadas.

Os autores destacam que uma parcela significativa do tráfego urbano é composta por veículos procurando vagas disponíveis, o que contribui diretamente para congestionamentos, desperdício de combustível e aumento da poluição atmosférica. Nesse contexto, os Smart Parking Systems (SPS) são apresentados como parte essencial da infraestrutura de Smart Cities, integrando sensores, redes de comunicação e sistemas computacionais para monitorar, processar e disponibilizar informações sobre ocupação de vagas aos usuários em tempo real.

Assim, o Smart Parking pode ser compreendido como um sistema inteligente de gerenciamento de vagas que utiliza tecnologias de detecção, comunicação e processamento de dados para otimizar o uso do espaço urbano e reduzir impactos operacionais e ambientais.

2.3 INTERNET DAS COISAS

A Internet das Coisas, conhecida pela sigla IoT (Internet of Things), representa um dos avanços mais significativos da tecnologia da informação na atualidade. Esse conceito refere-se à integração entre objetos físicos e sistemas digitais conectados à internet, permitindo que dispositivos coletem, processem e transmitam dados de forma automática. Segundo Fachini et al. (2017), a IoT caracteriza-se pela incorporação de conectividade em objetos do cotidiano, possibilitando sua participação em processos informacionais. Complementando essa visão, Nascimento e Silva, Frota e Santos (2023) destacam que essa tecnologia permite que os objetos adquiram uma espécie de “vida digital”, interagindo com o ambiente e com outros dispositivos conectados.

O desenvolvimento da IoT está diretamente relacionado à evolução da internet e das tecnologias de comunicação. Inicialmente, a internet era utilizada principalmente para conectar computadores, caracterizando um período conhecido como “internet das máquinas”. Com o avanço tecnológico e a popularização de dispositivos móveis, ocorreu a expansão da chamada “internet das pessoas”, na qual os usuários passaram a acessar serviços digitais de forma ampla. Nesse contexto, surge a Internet das Coisas como uma evolução natural desse processo, ampliando a conectividade para além das pessoas, incluindo também objetos e dispositivos físicos (FACHINI et al., 2017).

A IoT pode ser compreendida como uma infraestrutura tecnológica que possibilita a interconexão entre dispositivos, sensores e sistemas computacionais. De acordo com Nascimento e Silva, Frota e Santos (2023), essa integração ocorre por meio de arquiteturas tecnológicas que organizam a coleta, processamento e transmissão de dados, permitindo a criação de sistemas inteligentes e automatizados. Dessa forma, objetos comuns passam a ter capacidade de comunicação e tomada de decisão, tornando-se elementos ativos dentro de ambientes digitais.

Um dos principais componentes da Internet das Coisas são os sensores, responsáveis pela coleta de dados do ambiente, como temperatura, movimento e localização. Esses dados podem ser processados localmente ou enviados para sistemas maiores, que realizam análises e geram respostas automáticas. Além disso, a comunicação entre dispositivos é um aspecto essencial da IoT, permitindo que

equipamentos troquem informações entre si sem a necessidade de intervenção humana. Esse tipo de interação contribui para o desenvolvimento de sistemas autônomos e mais eficientes (FACHINI et al., 2017).

As aplicações da IoT são amplas e abrangem diferentes setores. No ambiente industrial, por exemplo, sensores são utilizados para monitorar máquinas e prever falhas, contribuindo para a manutenção preventiva e a otimização de processos. Em ambientes urbanos, a tecnologia pode ser aplicada no controle de tráfego, monitoramento ambiental e gestão de serviços públicos. Já no contexto residencial, a IoT está presente nas chamadas casas inteligentes, nas quais dispositivos conectados permitem controlar iluminação, temperatura e segurança de forma automatizada.

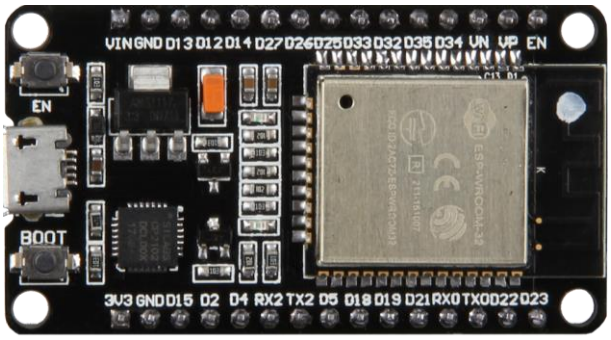
Diante desse cenário, a Internet das Coisas pode ser entendida como um paradigma tecnológico que promove a integração entre o mundo físico e o digital. Seu avanço está relacionado ao desenvolvimento de tecnologias de comunicação, sensores e capacidade de processamento de dados. Nesse sentido, a IoT tende a desempenhar um papel cada vez mais relevante na transformação digital da sociedade, contribuindo para a criação de soluções mais eficientes, automatizadas e inteligentes.

2.3.1 ESP32

O microcontrolador ESP32 tem se destacado como uma plataforma amplamente utilizada no desenvolvimento de sistemas embarcados e aplicações relacionadas à Internet das Coisas (IoT). Esse dispositivo reúne em uma única placa recursos de processamento, comunicação e controle de periféricos, permitindo o desenvolvimento de soluções tecnológicas voltadas à automação e ao monitoramento de ambientes. No contexto de sistemas inteligentes, o uso do ESP32 possibilita a integração entre sensores, dispositivos eletrônicos e redes de comunicação, permitindo a coleta e o processamento de dados em tempo real. Conforme destacam Rall, Leite e Miranda (2023), microcontroladores como o ESP32 desempenham papel fundamental em sistemas automatizados, pois permitem o gerenciamento de sensores e atuadores responsáveis pela execução de diferentes tarefas em um ambiente monitorado.

A Figura abaixo apresenta o módulo do microcontrolador ESP32 utilizado no desenvolvimento do sistema, evidenciando sua estrutura física e os principais componentes responsáveis pela conexão com sensores e demais dispositivos.

Figura 1 - Módulo Microcontrolador ESP32



Fonte: JOY-IT. SBC-NodeMCU-ESP32.

No desenvolvimento de sistemas de gestão de estacionamento, o ESP32 pode ser utilizado como o componente responsável pelo monitoramento das vagas e pela coleta de dados provenientes de sensores instalados no ambiente. Esses sensores podem detectar a presença ou ausência de veículos nas vagas, permitindo que o sistema identifique quais espaços estão ocupados e quais estão disponíveis. A partir dessas informações, o microcontrolador processa os dados coletados e os envia para o sistema responsável pelo armazenamento e gerenciamento das informações do estacionamento.

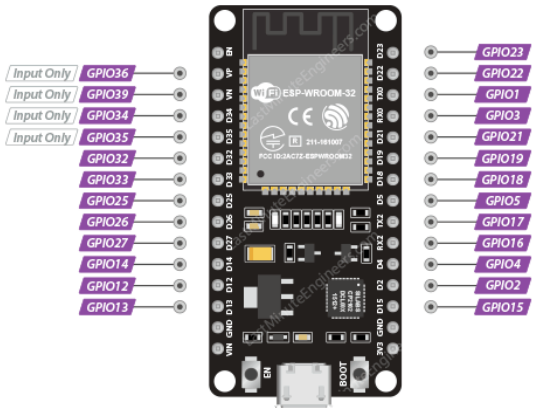
Uma das principais características do ESP32 é a presença de recursos de comunicação sem fio integrados, como Wi-Fi e Bluetooth. Esses recursos permitem que o microcontrolador envie dados para outros sistemas conectados à rede, possibilitando a integração entre o hardware responsável pela coleta de informações e o sistema responsável pelo processamento e visualização dos dados. De acordo com Rall, Leite e Miranda (2023), a conectividade integrada do ESP32 facilita o desenvolvimento de aplicações baseadas em IoT, pois permite que dispositivos físicos se comuniquem diretamente com sistemas digitais por meio da rede.

Outro aspecto relevante do ESP32 é sua capacidade de interação com diferentes dispositivos eletrônicos por meio de suas portas de entrada e saída, conhecidas como GPIO (General Purpose Input Output). Essas portas permitem a conexão com sensores e outros componentes eletrônicos utilizados em sistemas de automação. Em um sistema de gestão de estacionamento, por exemplo, sensores

podem ser instalados em cada vaga para detectar a presença de veículos, enviando essas informações ao microcontrolador, que será responsável por interpretar os dados e atualizar o estado do sistema.

A seguir um exemplar do mapeamento dos pinos (pinout) do ESP32, evidenciando as diferentes portas disponíveis e suas respectivas funções, aspecto essencial para a correta configuração e utilização do microcontrolador no projeto.

Figura 2 - Mapeamento de pinos do ESP32 (Pinout)

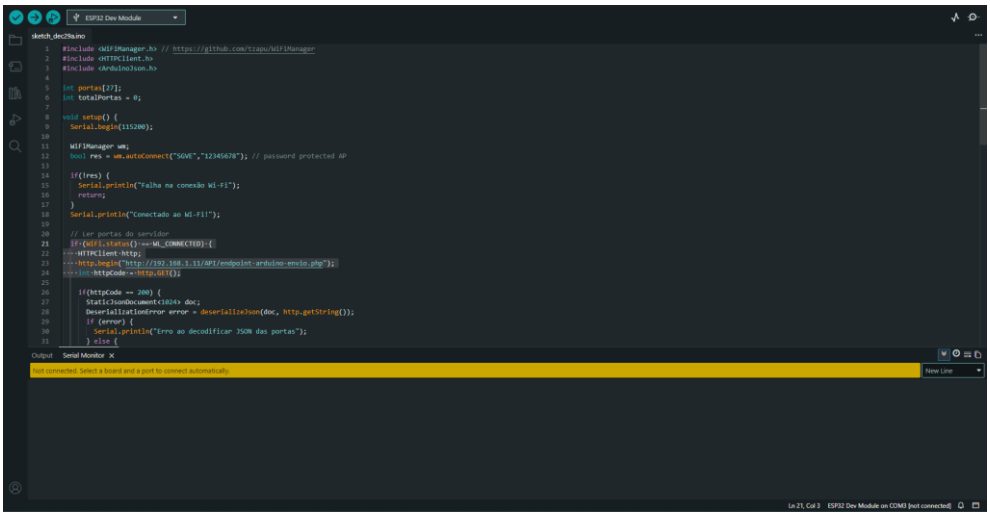


Fonte: LAST MINUTE ENGINEERS. ESP32 Pinout Reference.

Para o desenvolvimento das aplicações que executam no microcontrolador, pode-se utilizar ambientes de desenvolvimento integrados como o Arduino IDE. Esse ambiente oferece ferramentas que permitem escrever, compilar e enviar programas para o ESP32 de forma simplificada. Embora tenha sido inicialmente desenvolvido para placas da plataforma Arduino, o Arduino IDE também oferece suporte ao ESP32 por meio de bibliotecas e ferramentas que permitem a criação de aplicações utilizando as linguagens de programação C e C++. Conforme apresentado por Rall, Leite e Miranda (2023), esse ambiente facilita o desenvolvimento de aplicações embarcadas ao disponibilizar recursos que auxiliam na comunicação com sensores e outros dispositivos conectados ao microcontrolador.

Nesse sentido, observa-se, na Figura 3, a interface do Arduino IDE utilizada no desenvolvimento do sistema, evidenciando o ambiente de programação e os recursos disponíveis para a escrita e envio do código ao microcontrolador.

Figura 3 - Interface do Arduino IDE



Fonte: Os autores

Nos sistemas de monitoramento baseados em microcontroladores, os sensores desempenham papel fundamental, pois são responsáveis por coletar informações do ambiente em tempo real. Esses dispositivos podem detectar diferentes condições, como movimento, presença ou variações ambientais. No caso de um sistema de gestão de estacionamento, sensores instalados nas vagas podem identificar quando um veículo ocupa ou libera um espaço, enviando essas informações ao microcontrolador responsável pelo sistema.

Dessa forma, o ESP32 atua como um elemento central na arquitetura do sistema, sendo responsável por coletar dados dos sensores, processar essas informações e enviá-las ao sistema responsável pelo gerenciamento do estacionamento. A utilização desse tipo de tecnologia permite a criação de soluções automatizadas capazes de melhorar o controle das vagas disponíveis, otimizar a utilização dos espaços e fornecer informações atualizadas sobre a ocupação do estacionamento.

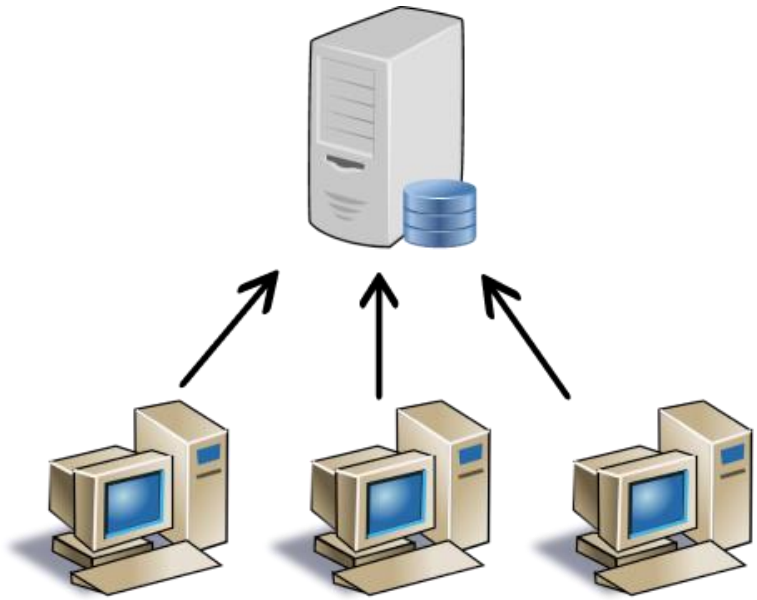
Assim, o ESP32 apresenta-se como uma plataforma adequada para o desenvolvimento de sistemas inteligentes de monitoramento, especialmente em aplicações relacionadas à gestão de estacionamento. A integração entre sensores, microcontroladores e sistemas de comunicação permite a criação de soluções tecnológicas capazes de automatizar a coleta de dados e melhorar a eficiência na administração de espaços destinados ao estacionamento de veículos.

2.4 ARQUITETURA CLIENTE-SERVIDOR

A arquitetura cliente-servidor é um modelo amplamente utilizado no desenvolvimento de sistemas distribuídos, no qual as responsabilidades da aplicação são separadas entre dois componentes principais: o cliente e o servidor. Nesse modelo, o cliente é responsável por realizar requisições e apresentar informações ao usuário, enquanto o servidor processa essas requisições, executa regras de negócio e fornece respostas adequadas. Essa abordagem possibilita a divisão de responsabilidades entre diferentes camadas do sistema, favorecendo maior organização estrutural, manutenção e escalabilidade das aplicações (SOUZA, 2021).

A Figura 4 a seguir representa o modelo cliente-servidor, evidenciando a relação entre os clientes que realizam requisições e o servidor centralizado que as processa e responde.

Figura 4 - Modelo de Arquitetura Cliente-Servidor

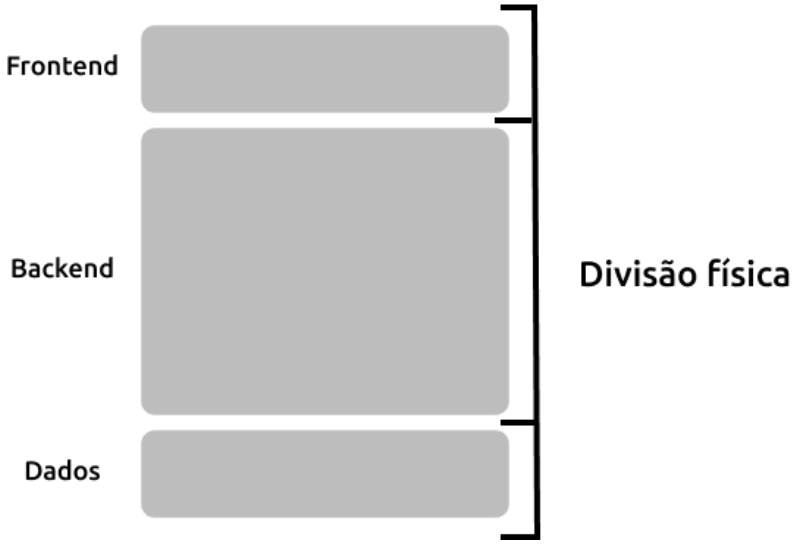


Fonte: SOUZA, Isaac Felisberto de. Camadas. Guia.dev, 2021.

A arquitetura cliente-servidor fundamenta a divisão do SGVE em camadas funcionais bem definidas. Segundo Souza (2021), essa organização é composta por três camadas principais: a camada de apresentação, responsável pela interface com o usuário; a camada de aplicação ou lógica de negócio, que contém as regras e operações do sistema; e a camada de dados, responsável pela persistência e gerenciamento das informações. Essa estrutura facilita a manutenção do sistema, pois cada camada pode ser modificada sem afetar diretamente as demais.

A Figura 5 a seguir ilustra essa divisão em três camadas — Frontend, Backend e Dados —, correspondendo respectivamente às camadas de apresentação, lógica de negócio e persistência adotadas no SGVE.

Figura 5 - Arquitetura em Camadas (Three-tier Architecture)



Fonte: SOUZA, Isaac Felisberto de. Camadas. Guia.dev, 2021.

No modelo cliente-servidor, a camada de apresentação está localizada no cliente, sendo responsável pela interação com o usuário e pelo envio de requisições ao servidor. O servidor concentra a lógica de negócio e o acesso ao banco de dados, processando as solicitações recebidas e retornando os resultados ao cliente. Esse fluxo de comunicação ocorre por meio de protocolos de rede, permitindo que o sistema seja acessado remotamente e utilizado por múltiplos usuários simultaneamente.

A centralização de recursos no servidor possibilita maior controle sobre dados e regras do sistema, reduzindo inconsistências e facilitando a implementação de mecanismos de segurança e controle de acesso. No contexto do SGVE, essa característica é essencial, pois o servidor é o único ponto de processamento das leituras enviadas pelos sensores IoT e das requisições geradas pela interface web, garantindo que os dados de ocupação das vagas sejam sempre consistentes independentemente do número de clientes conectados.

A separação entre cliente e servidor também contribui para a manutenibilidade do sistema ao longo do tempo. Como as responsabilidades estão bem definidas entre as camadas, é possível atualizar ou substituir partes específicas da aplicação sem comprometer toda a estrutura. No SGVE, isso significa que ajustes na interface web

podem ser realizados sem alterar a lógica de processamento do servidor, e melhorias no banco de dados podem ocorrer sem impacto direto na experiência do usuário (SOUZA, 2021).

2.5 API REST

A **API REST (Representational State Transfer)** é um estilo arquitetural utilizado para comunicação entre sistemas distribuídos por meio do protocolo HTTP. Nesse modelo, a interação ocorre por meio de **recursos**, que representam entidades do sistema — no contexto do SGVE, os principais recursos estão relacionados aos dados de ocupação das vagas e aos sensores monitorados. Cada recurso é identificado por uma URL (endpoint), permitindo acesso e manipulação de forma padronizada (IBM, 2023).

Os **métodos HTTP** definem as operações realizadas sobre esses recursos. No SGVE, destacam-se o uso do método **POST**, utilizado pelo ESP32 para enviar ao servidor os dados de ocupação das vagas, e o método **GET**, utilizado tanto pelo microcontrolador para obter configurações quanto pela interface web para consultar o estado atualizado das vagas. Esses métodos organizam a comunicação entre hardware, backend e frontend, garantindo padronização e clareza no fluxo de dados.

A troca de informações é realizada no formato **JSON (JavaScript Object Notation)**, que permite representar os dados em pares de chave e valor de forma leve e de fácil interpretação. No sistema desenvolvido, o ESP32 envia os dados dos sensores em formato JSON ao backend, que processa e armazena essas informações no banco de dados. Da mesma forma, a interface web consome esses dados em JSON para exibir, em tempo real, a ocupação das vagas aos usuários (IBM, 2023).

No contexto do SGVE, a API REST é fundamental para viabilizar a comunicação entre o **ESP32 e o backend**, permitindo que os dados coletados pelos sensores sejam transmitidos continuamente por meio de requisições HTTP. Essa integração possibilita a atualização em tempo real das informações exibidas na interface web, caracterizando o sistema como uma aplicação de **Internet das Coisas (IoT)**. Dessa forma, a API REST atua como elemento central na arquitetura do sistema, conectando a camada de hardware à aplicação web de forma eficiente e padronizada.

2.6 APLICAÇÕES WEB

A construção do SGVE exigiu a seleção criteriosa de tecnologias capazes de atender a três requisitos centrais: comunicação em tempo real com dispositivos IoT, exibição dinâmica de dados de ocupação de vagas e controle de acesso por perfil de usuário. As tecnologias adotadas — HTML, CSS, JavaScript e PHP — não foram escolhidas arbitrariamente, mas em função de sua complementaridade técnica, maturidade como padrão e adequação às limitações de infraestrutura típicas de ambientes de estacionamento, onde soluções leves e sem dependências externas complexas são preferíveis.

2.6.1 HTML COMO LINGUAGEM DE MARCAÇÃO

A linguagem HTML (HyperText Markup Language) é a base estrutural da maioria das páginas disponíveis na World Wide Web. Diferentemente de linguagens de programação tradicionais, o HTML é classificado como uma linguagem de marcação, cuja função principal é organizar e estruturar o conteúdo de uma página web por meio de elementos denominados tags, que permitem definir títulos, parágrafos, imagens, links e outros componentes que compõem uma interface web (K19, 2013).

A estrutura de um documento HTML é composta por elementos representados por tags escritas entre sinais de menor e maior (< >), que geralmente aparecem em pares. A organização básica inclui os elementos <html>, <head> e <body>, responsáveis respectivamente por delimitar o documento, armazenar configurações e conter o conteúdo visível da página. Quando um navegador acessa uma página web, realiza uma requisição ao servidor e recebe como resposta o código HTML que será renderizado na tela do usuário (K19, 2013).

No contexto do SGVE, o HTML foi adotado como camada de estruturação da interface por ser universalmente interpretado por navegadores sem necessidade de instalação de plugins ou aplicativos adicionais, o que garante acessibilidade ao sistema tanto em computadores quanto em dispositivos móveis presentes no ambiente do estacionamento. Essa característica é relevante num cenário onde operadores e administradores podem acessar o painel de controle a partir de diferentes equipamentos, sem depender de um sistema operacional específico.

2.6.2 CSS COMO SEPARAÇÃO ENTRE ESTRUTURA E APRESENTAÇÃO

O CSS (Cascading Style Sheets) é uma linguagem utilizada para definir a aparência visual de documentos estruturados em HTML, controlando aspectos como cores, fontes, espaçamentos, posicionamento de elementos e layout geral da interface. Essa separação entre estrutura e apresentação tornou-se um princípio fundamental do desenvolvimento web moderno, pois facilita a manutenção e a organização do código.

O CSS foi proposto em 1996 por Håkon Wium Lie, em colaboração com Bert Bos, com o objetivo de resolver limitações na formatação de páginas HTML. Antes disso, elementos de estilo eram incorporados diretamente no HTML, tornando os documentos extensos e difíceis de manter. Com as folhas de estilo, tornou-se possível definir regras de formatação separadamente do conteúdo, permitindo que um único arquivo CSS seja aplicado a diversas páginas de um mesmo site (K19, 2013). O conceito de cascata define qual regra prevalece quando há conflitos, considerando fatores como prioridade e especificidade dos seletores. Além disso, técnicas modernas como Flexbox e Grid Layout permitem a construção de interfaces responsivas, compatíveis com dispositivos móveis, tablets e computadores (K19, 2013).

A decisão de utilizar CSS no SGVE está diretamente ligada à necessidade de organizar visualmente as informações de ocupação de vagas de forma clara e hierárquica. Em um painel de controle acessado a partir de um monitor fixo, a apresentação visual bem estruturada é determinante para que operadores identifiquem rapidamente o estado de cada vaga — se ocupada, livre ou com falha de sensor. O uso de recursos como Flexbox permitiu posicionar os indicadores de vaga de forma ordenada e consistente na tela, enquanto a separação entre estrutura HTML e estilos CSS facilitou ajustes visuais sem a necessidade de alterar o código da lógica da aplicação, contribuindo para a manutenibilidade do sistema ao longo do desenvolvimento.

2.6.3 JAVASCRIPT COMO CAMADA DE INTERATIVIDADE

A linguagem JavaScript foi criada em 1995 por Brendan Eich na empresa Netscape Communications, com o objetivo de permitir que páginas web apresentassem comportamentos dinâmicos diretamente no navegador. Junto com

HTML e CSS, forma o conjunto de tecnologias fundamentais do desenvolvimento web: o HTML define a estrutura, o CSS controla a apresentação e o JavaScript implementa a lógica e a interatividade (BIFFI, 2021).

De acordo com Biffi (2021), o JavaScript permite que o sistema responda a eventos gerados pelo usuário, como cliques em botões e preenchimento de formulários. Uma de suas funcionalidades mais importantes é a manipulação do DOM (Document Object Model), que representa os elementos da página como uma estrutura de objetos. Por meio do DOM, é possível alterar conteúdos, estilos e comportamentos da interface sem recarregar a página, tornando as aplicações web mais dinâmicas e responsivas.

No SGVE, essa capacidade é particularmente crítica: a situação das vagas muda em tempo real conforme veículos entram e saem, e seria inviável exigir que o usuário recarregue manualmente a página para obter informações atualizadas. A utilização de JavaScript para realizar requisições assíncronas ao servidor — por meio de técnicas como AJAX — permitiu que o estado das vagas fosse atualizado automaticamente na interface sem interrupção do fluxo de uso. Isso garante que operadores do estacionamento visualizem dados precisos sem ação adicional, o que é essencial para a tomada de decisão em tempo real.

2.6.4 PAPEL DO BACKEND (PHP)

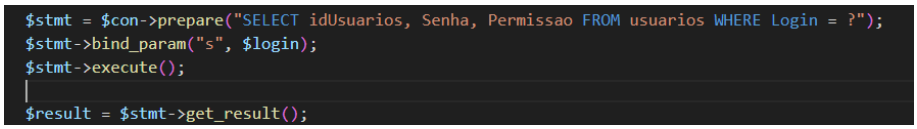
O PHP (PHP: Hypertext Preprocessor) é uma linguagem de programação amplamente utilizada no desenvolvimento de aplicações web dinâmicas, executada no lado do servidor. Foi criada em 1994 por Rasmus Lerdorf e evoluiu para se tornar uma linguagem completa para desenvolvimento web. De acordo com Bento (2017), o PHP permite o processamento de páginas antes de serem enviadas ao navegador, gerando o conteúdo final em formato HTML a partir de dados processados no servidor. Quando um usuário acessa uma página PHP, o servidor recebe a requisição, executa o script e retorna o resultado como HTML ao navegador, possibilitando a criação de aplicações dinâmicas capazes de interagir com bancos de dados e responder às ações dos usuários (BENTO, 2017).

Uma das principais funcionalidades do PHP é sua capacidade de integração com bancos de dados relacionais. Por meio de extensões como o mysqli, é possível

estabelecer conexões com sistemas gerenciadores de banco de dados, executar consultas SQL e manipular os dados armazenados. O uso de prepared statements — instruções SQL parametrizadas — é uma prática recomendada nesse contexto, pois impede que dados fornecidos pelo usuário sejam interpretados como comandos SQL, prevenindo ataques do tipo SQL Injection.

A Figura 6 a seguir demonstra a aplicação dessa técnica na consulta de autenticação do SGVE, evidenciando como os parâmetros do usuário são separados do comando SQL para garantir a integridade da operação.

Figura 6 - Consulta preparada em PHP para autenticação de usuário.



Fonte: Os autores.

O PHP também oferece suporte ao gerenciamento de sessões, essencial para sistemas com autenticação de usuários. Por meio da função `session_start()` e da variável superglobal `$_SESSION`, é possível armazenar informações do usuário autenticado no servidor e recuperá-las ao longo da navegação entre diferentes páginas. Esse mecanismo permite que o sistema identifique qual perfil está acessando cada recurso e aplique as regras de controle de acesso correspondentes, garantindo que cada perfil visualize apenas as funcionalidades a ele permitidas.

A Figura 7 a seguir apresenta as funções de controle de sessão e verificação de permissão implementadas no SGVE, responsáveis por restringir o acesso às páginas conforme o nível do usuário autenticado.

Figura 7 - Funções PHP para controle de sessão, login e verificação de permissão.

```php
function iniciarSessao() {
    if (session_status() === PHP_SESSION_NONE) {
        session_start();
    }
}

function exigirLogin() {
    iniciarSessao();

    if (!isset($_SESSION['usuario_id'])) {
        header("Location: ../Views/login.php");
        exit;
    }
}

function exigirPermissaoMinima($nivelMinimo) {
    iniciarSessao();

    if (!isset($_SESSION['usuario_id'])) {
        header("Location: /login.php");
        exit;
    }

    if (!isset($_SESSION['permissao'])) {
        http_response_code(403);
        echo "Acesso negado";
        exit;
    }
```

Fonte: Os autores.

No que diz respeito à segurança de credenciais, o PHP disponibiliza funções nativas para o armazenamento seguro de senhas. A função `password_hash()` aplica o algoritmo bcrypt sobre a senha fornecida, gerando um hash criptográfico armazenado no banco de dados no lugar da senha original. O bcrypt incorpora um fator de custo que torna o processo computacionalmente custoso, dificultando ataques de força bruta e de dicionário. No momento da autenticação, a função `password_verify()` compara a senha informada com o hash armazenado, garantindo que a senha nunca trafegue ou seja armazenada em texto simples.

A Figura 8 a seguir demonstra a aplicação da função `password_hash()` no módulo de cadastro de usuários do SGVE, ilustrando como a senha é convertida em hash antes de ser persistida no banco de dados.

Figura 8 - Criptografia de senha com bcrypt.

```php
$hash = password_hash($senha, PASSWORD_BCRYPT);
```

Fonte: Os autores.

A escolha do PHP como linguagem de backend no SGVE foi motivada por fatores tanto técnicos quanto práticos. Do ponto de vista técnico, o PHP oferece integração nativa com o banco de dados MariaDB utilizado no projeto, suporte robusto a sessões e mecanismos consolidados de segurança — como prepared statements e

hashing de senhas —, sem exigir configurações adicionais complexas. Do ponto de vista prático, sua ampla documentação, baixo custo de hospedagem e compatibilidade com servidores Apache — infraestrutura padrão em ambientes educacionais e de pequeno porte — tornaram-no uma escolha adequada ao contexto de desenvolvimento do sistema. A combinação de prepared statements, gerenciamento de sessões e hashing de senhas representa um conjunto de práticas fundamentais para o desenvolvimento de sistemas web seguros e confiáveis, especialmente em um sistema que armazena dados de acesso de usuários e histórico de movimentação de veículos.

2.7 BANCO DE DADOS RELACIONAL

Os sistemas de banco de dados surgiram como uma solução para organizar, armazenar e recuperar grandes volumes de informação de forma estruturada. Conforme destacam França e Júnior (2015), um banco de dados deve ter significado implícito e se referir a fatos, sendo estruturado para facilitar a busca e atualização dos dados. De maneira geral, pode ser definido como uma coleção de dados relacionados que representam algum aspecto do mundo real, organizados para atender a uma finalidade específica dentro de um sistema de informação (ELMASRI; NAVATHE, 2011).

Nesse contexto surgiu o modelo de banco de dados relacional, que se tornou o padrão predominante utilizado pelos sistemas de gerenciamento de banco de dados. De acordo com França e Júnior (2015), esse modelo foi concebido para separar o modelo físico do conceitual, sendo baseado na lógica e na teoria de conjuntos. Ele organiza os dados em tabelas compostas por linhas e colunas, onde cada linha representa um registro e cada coluna representa um atributo. Cada registro é identificado de forma única por uma chave primária, enquanto as chaves estrangeiras estabelecem vínculos entre tabelas distintas, permitindo representar estruturas mais complexas sem duplicação de informações (ELMASRI; NAVATHE, 2011; FRANÇA; JÚNIOR, 2015).

Além da estrutura baseada em tabelas e chaves, os bancos de dados relacionais utilizam restrições de integridade para garantir consistência e validade dos dados, destacando-se a integridade de entidade, que impede que a chave primária seja nula, e a integridade referencial, que assegura que chaves estrangeiras

correspondam a registros válidos (ELMASRI; NAVATHE, 2011). A manipulação dessas informações é realizada por meio da SQL (Structured Query Language), que permite operações de inserção, atualização, exclusão e consulta, tornando o modelo relacional uma das principais bases tecnológicas no desenvolvimento de sistemas de informação modernos (ELMASRI; NAVATHE, 2011; FRANÇA; JÚNIOR, 2015).

2.7.1 SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD) E MYSQL

Os bancos de dados são utilizados para armazenar e organizar informações de maneira estruturada. Entretanto, para que esses dados possam ser manipulados de forma eficiente e segura, é necessário o uso de um software responsável por gerenciar esse armazenamento e permitir o acesso às informações. Esse software é conhecido como Sistema Gerenciador de Banco de Dados (SGBD). De forma geral, um SGBD pode ser definido como um conjunto de programas que possibilita a criação, manipulação, controle e administração de bancos de dados, facilitando os processos de definição, construção, manipulação e compartilhamento dos dados (ELMASRI; NAVATHE, 2011; FRANÇA; JÚNIOR, 2015).

Segundo Elmasri e Navathe (2011), o SGBD atua como uma interface entre os usuários e o banco de dados, permitindo que aplicações e usuários possam acessar os dados sem a necessidade de conhecer detalhes internos de armazenamento. Dessa forma, o sistema gerenciador é responsável por diversas funções importantes, como controle de acesso aos dados, manutenção da integridade das informações, gerenciamento de transações e recuperação de dados em caso de falhas. Essas funcionalidades são fundamentais para garantir que os dados armazenados permaneçam consistentes e disponíveis para os usuários do sistema.

Outra característica importante dos SGBDs é a possibilidade de controlar o acesso simultâneo de múltiplos usuários ao banco de dados. Em ambientes corporativos, é comum que vários usuários ou sistemas acessem o banco de dados ao mesmo tempo. Nesse contexto, o SGBD precisa garantir que as operações realizadas não causem inconsistências nas informações armazenadas. Para isso, são utilizadas técnicas de controle de concorrência e gerenciamento de transações, que asseguram que os dados sejam manipulados de forma segura e confiável. Conforme destacam França e Júnior (2015), o SGBD precisa oferecer suporte indispensável à medida que crescem o volume de dados e o número de usuários simultâneos,

garantindo a robustez e a eficiência no gerenciamento das informações (ELMASRI; NAVATHE, 2011).

Além disso, os SGBDs fornecem mecanismos para definição da estrutura do banco de dados, normalmente por meio de linguagens específicas como a SQL (Structured Query Language). De acordo com França e Júnior (2015), de maneira simplista, um banco de dados especifica os dados, as estruturas e as restrições, enquanto o SGBD gerencia a manipulação desses dados, extraindo essas responsabilidades dos usuários e aplicativos. Essa linguagem permite que desenvolvedores e administradores criem tabelas, definam relacionamentos entre dados e realizem operações de consulta e manipulação das informações armazenadas. Dessa forma, os sistemas de gerenciamento de banco de dados tornam possível a construção de aplicações que dependem de grandes volumes de dados organizados.

Entre os diversos sistemas gerenciadores de banco de dados existentes atualmente, um dos mais utilizados é o MySQL. O MySQL é um sistema de gerenciamento de banco de dados relacional amplamente utilizado em aplicações web e sistemas corporativos. Ele utiliza o modelo relacional para organizar os dados em tabelas interligadas e oferece suporte à linguagem SQL para manipulação e consulta das informações armazenadas.

O MySQL se destaca por ser um software de código aberto, o que significa que seu código pode ser utilizado e modificado livremente, além de possuir grande comunidade de usuários e desenvolvedores. Esse sistema é amplamente empregado no desenvolvimento de aplicações web, principalmente em conjunto com tecnologias como PHP, JavaScript e servidores web. Sua popularidade está relacionada à facilidade de uso, bom desempenho e capacidade de lidar com grandes volumes de dados.

Outra característica importante do MySQL é a confiabilidade no armazenamento e processamento das informações. O sistema oferece mecanismos de segurança, controle de acesso de usuários e gerenciamento de transações, garantindo que os dados armazenados permaneçam íntegros mesmo em situações de falha ou acesso simultâneo por múltiplos usuários. Dessa forma, o MySQL tornou-se uma das principais soluções de banco de dados utilizadas no desenvolvimento de sistemas modernos.

Portanto, os sistemas gerenciadores de banco de dados desempenham um papel fundamental na organização e manipulação de informações em sistemas computacionais. Ferramentas como o MySQL possibilitam que aplicações armazenem grandes volumes de dados de maneira estruturada, segura e eficiente, sendo amplamente utilizadas em diversos tipos de sistemas e aplicações tecnológicas.

3 DESENVOLVIMENTO

Nesta seção, são apresentados os aspectos relacionados ao desenvolvimento do SGVE, incluindo a estrutura do sistema, as tecnologias utilizadas e o funcionamento geral da solução proposta para o monitoramento de vagas em tempo real.

3.1 VISÃO GERAL DA ARQUITETURA DO SISTEMA

O SGVE é estruturado sobre uma arquitetura cliente-servidor em três camadas, operando exclusivamente em rede local (LAN), sem exposição a redes externas ou acesso via internet. Essa escolha arquitetural reduz a complexidade de infraestrutura, minimiza superfícies de ataque e garante baixa latência na comunicação entre os componentes do sistema.

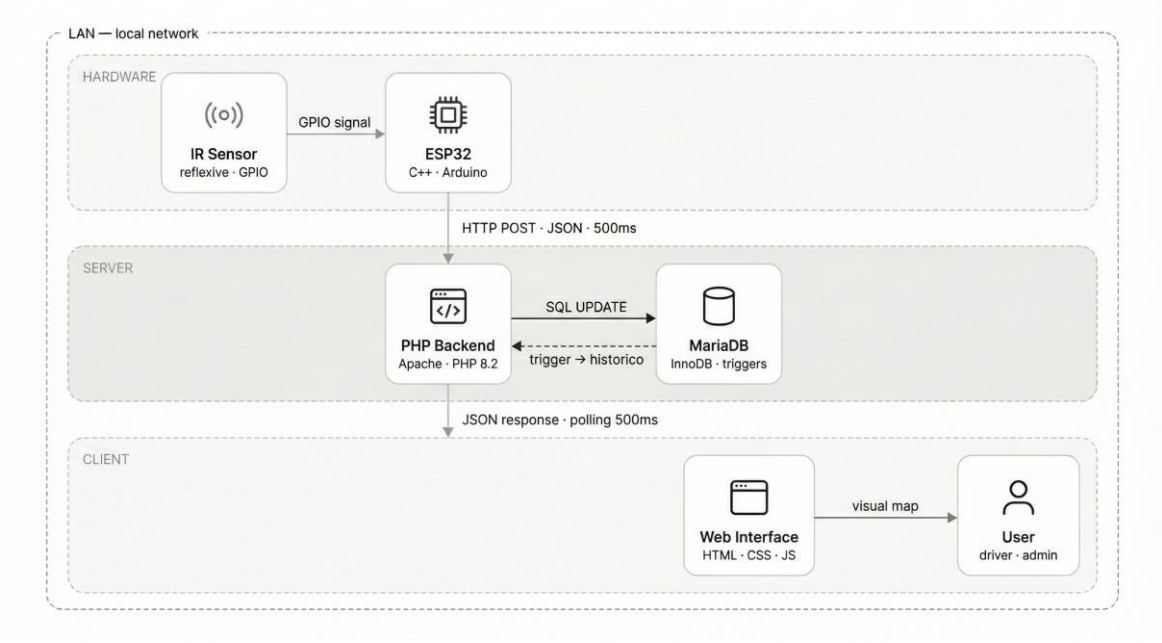
A camada de hardware é composta por sensores infravermelhos reflexivos instalados nas vagas do estacionamento, conectados a um microcontrolador ESP32. Esse microcontrolador é responsável pela coleta contínua dos dados de ocupação de múltiplas vagas e pela transmissão dessas informações ao servidor por meio de requisições HTTP sobre a rede local.

A camada de backend é executada em um computador convencional configurado como servidor na rede interna. Nesse servidor, as requisições provenientes do ESP32 são recebidas e processadas pela aplicação desenvolvida em PHP, que realiza operações de leitura e escrita no banco de dados MariaDB. As informações de estado atual das vagas, histórico de ocupação e dados dos usuários são armazenadas no banco de dados. O controle de acesso é gerenciado por meio de sessões PHP, que garantem que cada perfil de usuário acesse apenas as funcionalidades a ele permitidas.

A camada de apresentação é acessada pelos usuários por meio de navegador web, em dispositivos desktop ou móveis conectados à mesma rede local. A interface é desenvolvida com HTML, CSS e JavaScript, sendo disponibilizadas experiências distintas para os dois perfis de usuário do sistema: o motorista, que visualiza em tempo real a disponibilidade das vagas, e o administrador, que tem acesso a funcionalidades de configuração, gerenciamento de usuários e acompanhamento de indicadores operacionais.

A Figura 9 ilustra a arquitetura geral do sistema, evidenciando o fluxo de dados entre as camadas de hardware, backend e apresentação.

Figura 9 - Diagrama de fluxo de funcionamento do sistema

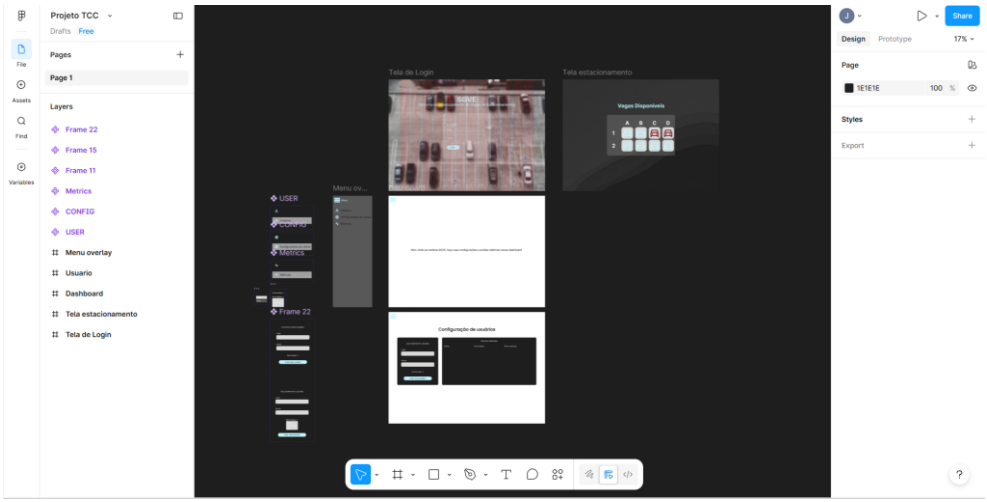


Fonte: Os autores.

3.2 INTERFACE WEB

Antes do desenvolvimento das telas, foi realizada a diagramação das interfaces principais no Figma, ferramenta de prototipação visual utilizada para planejar a estrutura, o layout e o fluxo de navegação do sistema. Essa etapa permitiu definir a organização dos elementos visuais e a disposição das funcionalidades de cada perfil de usuário antes da implementação, reduzindo retrabalho e garantindo maior consistência na interface final. As telas prototipadas abrangeram as principais funcionalidades do SGVE, incluindo a tela de login, o menu de navegação, o mapa de vagas, o dashboard administrativo e a configuração de usuários conforme ilustrado a seguir.

Figura 10 - Protótipo de interface do sistema com fluxo de telas no Figma.



Fonte: Os autores.

A interface web do SGVE foi desenvolvida com HTML, CSS e JavaScript, seguindo o princípio de separação entre estrutura, apresentação e comportamento. O acesso ao sistema é realizado por meio de navegador web, em dispositivos desktop ou móveis conectados à rede local, sem necessidade de instalação de software adicional.

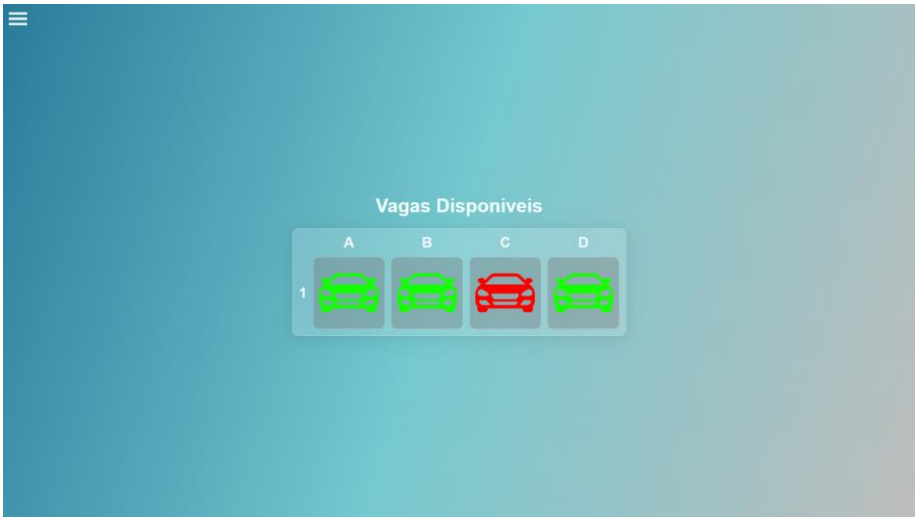
A tela de login constitui o ponto de entrada do sistema para todos os usuários. Nessa tela, as credenciais de acesso são solicitadas e validadas pelo backend PHP por meio do gerenciamento de sessões. Após a autenticação, o perfil do usuário é identificado e o redirecionamento para a interface correspondente é realizado automaticamente, garantindo que cada tipo de usuário acesse apenas as funcionalidades a ele permitidas. A Figura a seguir apresenta a interface da tela de login do SGVE, evidenciando a simplicidade do layout e o foco na usabilidade, aspectos importantes para facilitar o acesso inicial ao sistema.

Figura 11 - Tela de Login

Fonte: Os autores.

Para o perfil de motorista, a principal tela disponível é o mapa visual das vagas. Nessa interface, cada vaga do estacionamento é representada graficamente e identificada por codificação de cores: verde para vagas livres e vermelho para vagas ocupadas. Essa representação permite que a disponibilidade dos espaços seja identificada de forma imediata e intuitiva, reduzindo o tempo necessário para localização de uma vaga disponível. As informações exibidas são atualizadas a partir dos dados coletados pelos sensores e processados pelo backend, refletindo o estado real do estacionamento. A figura abaixo ilustra essa interface, destacando a organização das vagas e a utilização de elementos visuais que priorizam a usabilidade e a eficiência na tomada de decisão por parte do usuário.

Figura 12 - Dashboard de ocupação das vagas



Fonte: Os autores.

Para o perfil de administrador, um conjunto mais amplo de funcionalidades é disponibilizado. O painel de KPIs apresenta os principais indicadores operacionais do estacionamento, entre eles a taxa de ocupação atual, o total de vagas livres e ocupadas, o tempo médio de ocupação e o histórico de uso por período. Esses indicadores permitem que o desempenho do estacionamento seja acompanhado e que decisões sejam tomadas com base em dados.

A Figura 13 apresenta o painel administrativo do sistema, destacando a organização dos indicadores e a utilização de elementos gráficos que facilitam a interpretação das informações e o acompanhamento das métricas operacionais.

Figura 13 - Dashboard de métricas e ocupação do estacionamento.

Fonte: Os autores.

Além do painel de indicadores, a tela de gerenciamento de usuários é disponibilizada ao administrador, onde usuários podem ser cadastrados, editados e removidos, bem como seus perfis de acesso definidos. A tela de configuração de vagas possibilita o registro e o ajuste das vagas monitoradas pelo sistema, permitindo a adaptação do SGVE a diferentes layouts de estacionamento.

A separação clara entre as interfaces dos dois perfis de usuário reforça o princípio de controle de acesso por níveis de permissão, assegurando que informações administrativas e configurações do sistema estejam restritas aos usuários autorizados.

3.3 BACKEND E REGRAS DE NEGÓCIO

O backend do SGVE foi desenvolvido em PHP 8.2 puro, sendo executado sobre um servidor web Apache instalado no computador configurado como servidor da rede local. A conexão com o banco de dados MariaDB é realizada por meio da extensão mysqli do PHP, com os parâmetros de conexão centralizados no arquivo DB_CONFIG.php. A organização dos arquivos segue uma estrutura modular, com separação entre arquivos de configuração, endpoints de API, ações de processamento e views. As funções de autenticação e controle de permissão são definidas no arquivo validar.php, sendo reutilizadas em todas as páginas do sistema.

A autenticação dos usuários é realizada por meio de formulário de login, no qual as credenciais informadas são verificadas em relação aos registros armazenados no banco de dados. As senhas são armazenadas com hash bcrypt, gerado pela função password_hash(), e a verificação no momento do login é realizada pela função password_verify(). Após a validação, uma sessão PHP é iniciada e os dados do usuário autenticado — identificador e nível de permissão — são armazenados nessa sessão. O encerramento da sessão é tratado pelo arquivo logout.php, que limpa todas as variáveis de sessão e destrói a sessão antes de redirecionar o usuário para a tela de login.

O controle de acesso é implementado por meio de três níveis de permissão. O nível 1 permite acesso exclusivo ao dashboard de vagas. O nível 2 amplia o acesso para o painel de métricas e KPIs. O nível 3 concede acesso total ao sistema, incluindo o gerenciamento de usuários e a configuração de sensores. Em cada página, a função exigirLogin() verifica se a sessão está ativa, redirecionando para a tela de login caso contrário, e a função exigirPermissaoMinima() valida se o nível de permissão do usuário é suficiente para o acesso ao recurso solicitado, retornando HTTP 403 em caso de acesso não autorizado.

A comunicação com o hardware é tratada por três endpoints PHP dedicados, localizados no diretório API/. O endpoint api-receber.php recebe requisições HTTP POST enviadas pelo ESP32 contendo um array JSON com o estado de cada pino GPIO monitorado. Os dados recebidos são validados, e o status de cada sensor é atualizado no banco de dados somente quando uma mudança real é detectada, evitando escritas desnecessárias. O endpoint utiliza prepared statements para prevenção de injeção de SQL e retorna uma resposta JSON detalhada por item processado. O endpoint endpoint-arduino-envio.php responde a requisições GET do ESP32 com a lista de portas GPIO cadastradas no sistema, permitindo que o firmware configure dinamicamente os pinos a serem monitorados. O endpoint endpoint-frontend.php atende às requisições da interface web, retornando o estado atual de todas as vagas em formato JSON, sendo seu acesso restrito a usuários com sessão ativa.

As páginas da interface são servidas diretamente pelo backend PHP, que processa as requisições dos navegadores, aplica as regras de controle de acesso e retorna o conteúdo HTML correspondente ao perfil do usuário autenticado. O gerenciamento de usuários e sensores é tratado pelos arquivos do diretório Actions/,

que recebem os dados submetidos via formulário, realizam as operações no banco de dados por meio de prepared statements e redirecionam o usuário com mensagens de retorno sobre o resultado da operação.

3.4 BANCO DE DADOS

O banco de dados do SGVE foi implementado no MariaDB, denominado sgve, utilizando o conjunto de caracteres UTF-8 e o mecanismo de armazenamento InnoDB.

3.4.1 MODELO CONCEITUAL

O modelo conceitual do banco de dados do SGVE foi elaborado utilizando a notação entidade-relacionamento (ER), com o objetivo de representar de forma abstrata as informações gerenciadas pelo sistema e as relações existentes entre elas. O diagrama resultante é composto por três entidades: Sensor, Histórico_Ocupação e Usuário.

A entidade Sensor representa cada dispositivo físico instalado nas vagas do estacionamento, responsável por detectar a presença ou ausência de veículos. Seus atributos são: idSensor, chave primária que identifica unicamente cada sensor; linha e coluna, que definem a posição física da vaga na matriz do estacionamento; GPIO, que indica o pino do ESP32 ao qual o sensor está conectado; status, que armazena o estado atual da vaga (ocupada ou livre); ultima_ocupacao, que registra o momento em que a vaga foi ocupada pela última vez; e ultima_saida, que registra o momento da última liberação da vaga.

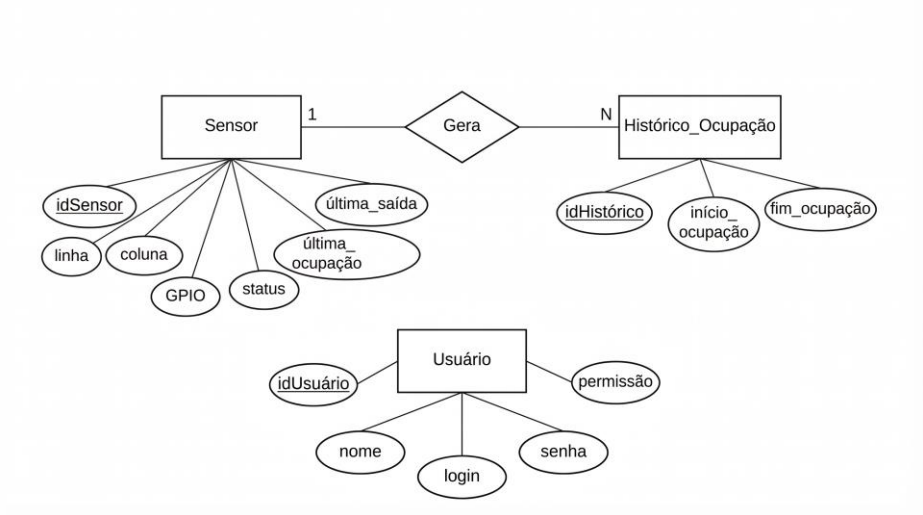
A entidade Histórico_Ocupação tem a finalidade de registrar o histórico completo de uso de cada vaga ao longo do tempo, permitindo consultas e análises futuras sobre a utilização do estacionamento. Seus atributos são: idHistórico, chave primária com incremento automático; inicio_ocupacao, que armazena a data e hora em que determinada vaga foi ocupada; e fim_ocupacao, que registra quando a vaga foi liberada, podendo ser nulo caso a ocupação ainda esteja em andamento.

A entidade Usuário representa as pessoas que têm acesso ao sistema de gerenciamento. Seus atributos são: idUsuário, chave primária; nome, que armazena o nome completo do usuário; login, atributo com restrição de unicidade utilizado para autenticação; senha, que armazena a senha de forma criptografada; e permissao, que define o nível de acesso do usuário no sistema.

O relacionamento modelado entre Sensor e Histórico_Ocupação é denominado Gera, com cardinalidade 1:N: um sensor pode gerar múltiplos registros de ocupação ao longo do tempo, enquanto cada registro de histórico pertence a exatamente um sensor. A entidade Usuário não possui relacionamento direto com as demais entidades no modelo conceitual, pois sua função é restrita ao controle de acesso à interface administrativa do sistema.

A Figura 14 apresenta o modelo conceitual do banco de dados do SGVE, elaborado por meio da notação entidade-relacionamento (ER), representando de forma abstrata as entidades do sistema, seus atributos e os relacionamentos existentes entre elas.

Figura 14 - Modelo conceitual do banco de dados



Fonte: Os autores.

3.4.2 MODELO LÓGICO

O modelo lógico foi derivado do modelo conceitual, representando a estrutura relacional do banco de dados com suas tabelas, chaves e restrições, sem especificação de tipos de dados físicos.

A tabela **sensores** possui como chave primária *idSensores*. Seus atributos são *Coluna*, *Linha*, *GPIO*, *Status*, *Ultima_ocupacao* e *Ultima_saida*. O atributo *GPIO* possui restrição de unicidade (UNIQUE), garantindo que nenhum pino seja atribuído a mais de um sensor simultaneamente.

A tabela **historico_ocupacao** possui como chave primária *id*. Seus atributos são *inicio_ocupacao* e *fim_ocupacao*, sendo este último opcional (permite valor nulo),

pois representa o encerramento de uma ocupação que pode ainda estar em andamento. O atributo *sensor_id* é chave estrangeira referenciando *idSensores* na tabela **sensores**, concretizando o relacionamento **1:N** entre as duas tabelas: um sensor pode estar associado a múltiplos registros de histórico, mas cada registro pertence a exatamente um sensor.

A tabela **usuarios** possui como chave primária *idUsuarios*. Seus atributos são *Nome*, *Login*, *Senha* e *Permissao*. O atributo *Login* possui restrição de unicidade (UNIQUE). Esta tabela não possui relacionamento com as demais, pois é responsável exclusivamente pelo controle de acesso ao sistema.

3.4.3 MODELO FÍSICO

O modelo físico representa a implementação concreta do banco de dados no sistema gerenciador MariaDB, definindo os tipos de dados, índices, restrições e rotinas automáticas utilizadas pelo SGVE.

A tabela sensores é implementada com *idSensores* como INT(11) NOT NULL AUTO_INCREMENT, sendo a chave primária da tabela. Os atributos *Coluna*, *Linha*, *GPIO* e *Status* são do tipo INT(11), sendo *GPIO* definido com restrição UNIQUE, impedindo duplicidade de pinos entre os sensores cadastrados. Os atributos *Ultima_ocupacao* e *Ultima_saida* são do tipo TIMESTAMP e aceitam valor nulo, sendo atualizados automaticamente por um trigger descrito adiante.

A tabela historico_ocupacao possui *id* como INT(11) NOT NULL AUTO_INCREMENT, chave primária da tabela. O atributo *sensor_id* é INT(11) NOT NULL e chave estrangeira referenciando *idSensores* em sensores. Os atributos *inicio_ocupacao* e *fim_ocupacao* são do tipo DATETIME, sendo *fim_ocupacao* nullable. Para otimização de consultas, foram criados índices sobre *sensor_id*, *inicio_ocupacao* e *fim_ocupacao*, dado que essas colunas são frequentemente utilizadas em filtros de histórico.

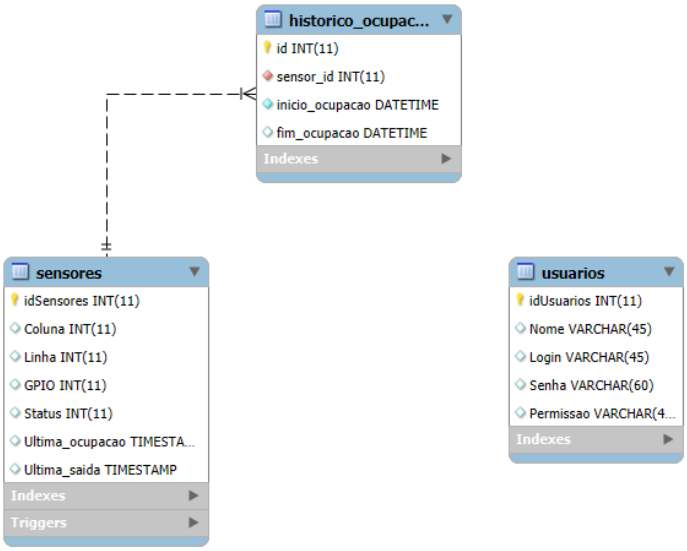
A tabela usuarios possui *idUsuarios* como INT(11) NOT NULL AUTO_INCREMENT. Os atributos *Nome*, *Login* e *Permissao* são VARCHAR(45), e *Senha* é VARCHAR(60), dimensionado para armazenar o hash gerado na autenticação. O atributo *Login* possui restrição UNIQUE.

Todas as tabelas utilizam o mecanismo de armazenamento InnoDB, que oferece suporte a chaves estrangeiras e controle transacional, e o conjunto de caracteres utf8mb3.

Para manter a consistência dos dados de ocupação sem depender da aplicação, foi implementado o trigger atualizar_timestamp_status, executado automaticamente antes de cada atualização na tabela sensores. Quando o campo *Status* é alterado para o valor 1, indicando que a vaga foi ocupada, o trigger atribui automaticamente o timestamp atual ao campo *Ultima_ocupacao*. Quando alterado para o valor 2, indicando liberação da vaga, o campo *Ultima_saida* é atualizado com o momento corrente. Dessa forma, os registros de tempo são gerenciados diretamente pelo banco de dados, independentemente do ESP32 ou da camada de aplicação.

A Figura 15 apresenta o modelo físico do banco de dados do SGVE, exibindo as tabelas implementadas no MariaDB com seus respectivos atributos, tipos de dados e relacionamentos.

Figura 15 - Modelo físico do banco de dados



Fonte: Os autores.

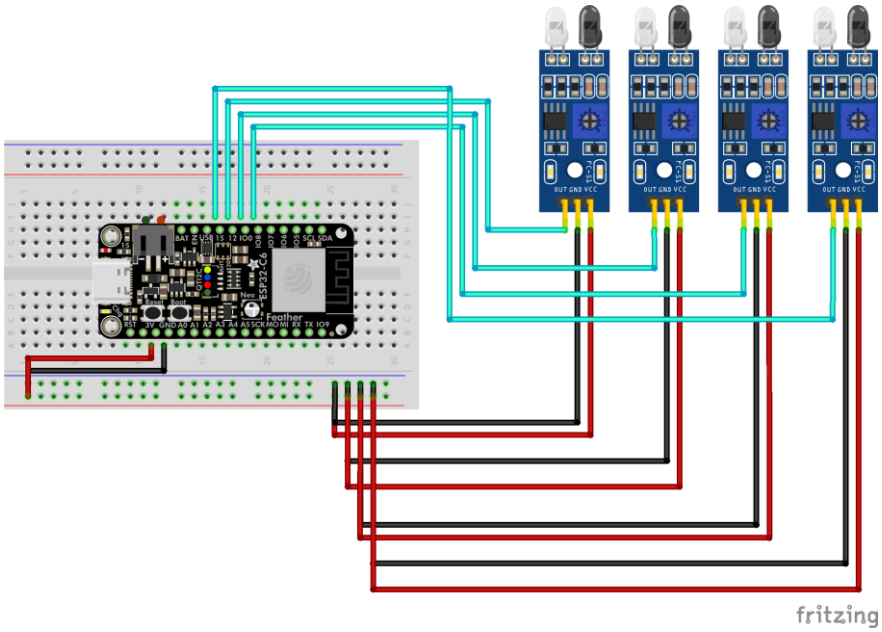
3.5 HARDWARE E COLETA DE DADOS

A camada de hardware do SGVE é composta por sensores infravermelhos reflexivos e pelo microcontrolador ESP32, responsáveis pela detecção da ocupação das vagas e pela transmissão dos dados ao backend do sistema.

A montagem do circuito foi realizada utilizando uma protoboard, na qual o ESP32 foi conectado aos sensores por meio de suas portas GPIO. Cada sensor possui três conexões principais: alimentação (VCC), terra (GND) e saída digital (OUT), sendo esta última responsável por enviar o sinal de detecção ao microcontrolador. A alimentação dos sensores foi distribuída de forma paralela, enquanto os sinais de saída foram conectados individualmente às portas GPIO do ESP32, permitindo a leitura independente de cada vaga.

A Figura 16 apresenta o diagrama de ligação dos componentes utilizados no sistema, evidenciando a conexão entre o microcontrolador ESP32 e os sensores infravermelhos reflexivos. No esquema, é possível observar a distribuição da alimentação elétrica, bem como a conexão individual dos sinais de saída dos sensores às portas GPIO do microcontrolador, responsáveis pela leitura dos estados de cada vaga.

Figura 16 - Diagrama de ligação do ESP32 com sensores infravermelhos.



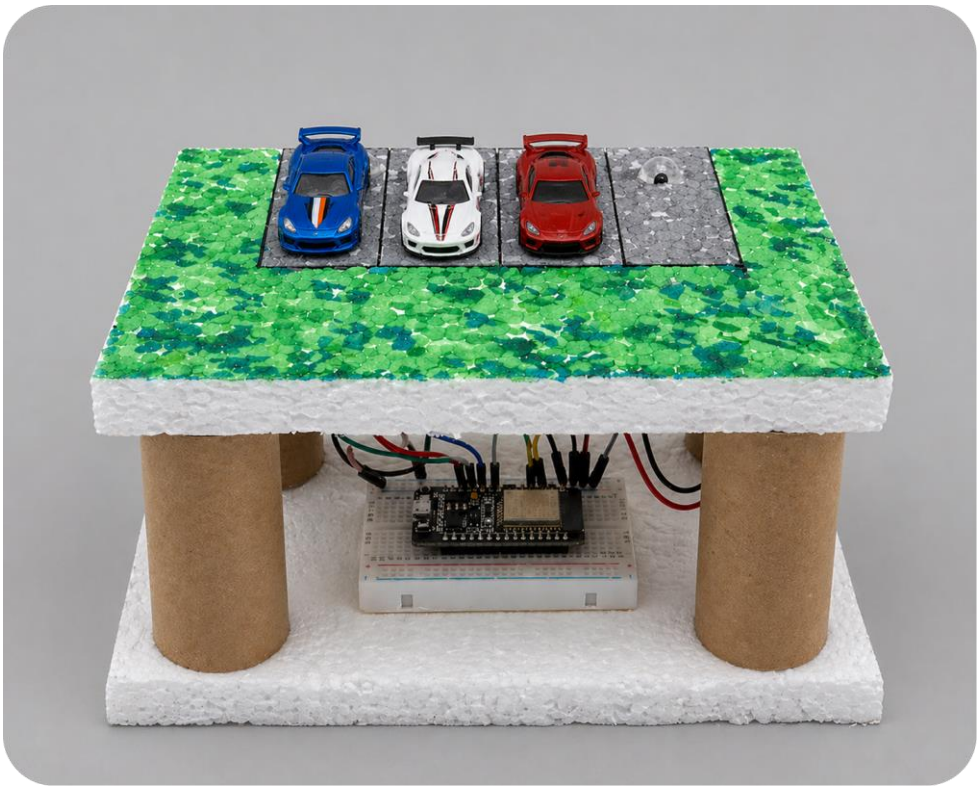
Fonte: Os autores.

Para validação prática do sistema, foi desenvolvida uma **maquete em escala reduzida**, representando um estacionamento com múltiplas vagas. Os sensores foram posicionados em cada vaga da maquete, permitindo simular a presença e ausência de veículos. Essa estrutura possibilitou testar o funcionamento do sistema em condições controladas, aproximando o protótipo de um cenário real de aplicação.

A Figura X ilustra a maquete em escala reduzida desenvolvida para a validação do sistema. Nela, é possível observar a disposição física das vagas de

estacionamento e o posicionamento dos sensores, permitindo a simulação do funcionamento do sistema em um ambiente controlado e representativo da aplicação real.

Figura 17 - Maquete física do sistema de estacionamento



Fonte: Os autores.

Os sensores infravermelhos reflexivos operam por meio da emissão de um feixe de luz infravermelha e da detecção do sinal refletido. Na ausência de um veículo, o sensor mantém um nível lógico correspondente ao estado livre. Quando um veículo ocupa a vaga, ocorre a reflexão do feixe, alterando o sinal de saída e indicando o estado de ocupação. Esse sinal é interpretado pelo ESP32 por meio de suas portas GPIO configuradas como entradas digitais.

O microcontrolador ESP32 atua como unidade central de processamento, sendo responsável pela leitura dos sensores e pela comunicação com o sistema backend. As portas GPIO utilizadas são previamente cadastradas no sistema e obtidas dinamicamente pelo firmware durante a inicialização, por meio de uma requisição HTTP GET ao endpoint correspondente.

O comportamento do ESP32 é definido por meio de um **firmware desenvolvido em linguagem C++**, utilizando o ambiente Arduino IDE. Esse programa realiza leituras periódicas dos sensores a cada 500 milissegundos,

processa os dados e os organiza em formato JSON. Em seguida, essas informações são enviadas ao servidor por meio de requisições HTTP do tipo POST, permitindo a atualização em tempo real do estado das vagas.

3.6 COMUNICAÇÃO ENTRE HARDWARE E BACKEND

A comunicação entre o microcontrolador ESP32 e o backend do SGVE é realizada por meio do protocolo HTTP sobre a rede local, seguindo o modelo arquitetural REST. Os dados são trafegados no formato JSON, permitindo que as informações coletadas pelos sensores sejam transmitidas e interpretadas de forma padronizada e eficiente entre o hardware e a aplicação PHP.

O fluxo de comunicação é iniciado pelo ESP32, que atua como cliente HTTP, realizando requisições ao servidor a cada 500 milissegundos. Em cada ciclo, um array JSON contendo o número do pino GPIO e o status de ocupação de cada vaga monitorada é enviado ao endpoint api-receber.php por meio de uma requisição POST. O endpoint recebe o conteúdo JSON, valida o formato e os campos presentes e realiza a atualização do estado de cada sensor no banco de dados, somente quando uma mudança real de status é detectada. Em caso de dados inválidos ou ausência de campos obrigatórios, respostas HTTP com códigos de erro apropriados são retornadas ao microcontrolador, permitindo o registro e o tratamento de falhas no lado do hardware.

Além do envio de dados, o ESP32 também realiza uma requisição GET ao endpoint endpoint-arduino-envio.php durante sua inicialização. Essa requisição retorna a lista de portas GPIO cadastradas no sistema em formato JSON, permitindo que o firmware configure dinamicamente os pinos a serem monitorados sem necessidade de alterações no código embarcado.

A interface web, por sua vez, realiza requisições GET ao endpoint endpoint-frontend.php a cada 500 milissegundos por meio da Fetch API do JavaScript, obtendo o estado atualizado de todas as vagas em formato JSON. Esse mecanismo de polling garante que o mapa visual exibido ao usuário reflita em tempo real as alterações detectadas pelos sensores, com latência compatível com o intervalo de atualização adotado.

A operação exclusiva em rede local elimina a necessidade de exposição do sistema a redes externas, reduzindo a superfície de ataque e garantindo baixa latência

nas comunicações entre os componentes. Toda a troca de informações entre hardware, backend e interface web ocorre dentro do ambiente interno da rede, mantendo o controle sobre o fluxo de dados e a segurança da infraestrutura.

4 VALIDAÇÃO DO SISTEMA

A validação do SGVE foi estruturada com o objetivo de verificar a relevância da problemática abordada, analisar a percepção dos usuários em relação à proposta e relacionar os resultados obtidos com estudos já existentes sobre sistemas inteligentes de estacionamento. Dessa forma, a validação não se limita apenas ao funcionamento técnico do protótipo, mas também considera a aplicabilidade prática da solução no contexto de uso real.

4.1 METODOLOGIA DA PESQUISA

Com o objetivo de validar a proposta do sistema desenvolvido, foi realizada uma pesquisa com indivíduos que utilizam estacionamentos em seu cotidiano. A amostra contemplou tanto usuários ocasionais quanto usuários frequentes, permitindo a obtenção de uma visão abrangente sobre as dificuldades enfrentadas na busca por vagas.

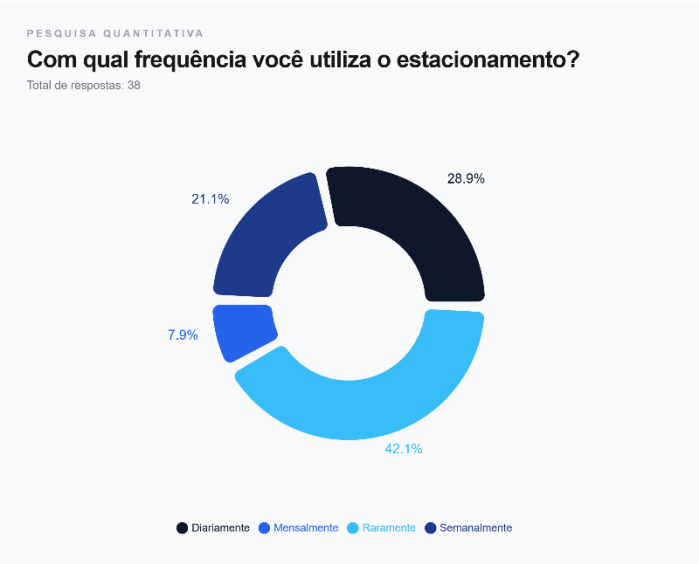
A coleta de dados ocorreu por meio de um questionário estruturado, composto por perguntas fechadas e objetivas. As questões foram elaboradas com foco na identificação de padrões de uso, tempo médio de procura por vagas, nível de satisfação com a organização dos estacionamentos e percepção dos usuários quanto à utilidade de um sistema de monitoramento em tempo real.

A pesquisa contou com 37 respostas válidas, sendo esse o total considerado para a análise dos resultados apresentados. Os dados foram organizados em gráficos estatísticos, possibilitando uma visualização clara e objetiva das respostas obtidas.

4.2 APRESENTAÇÃO DOS RESULTADOS

Os resultados da pesquisa permitiram identificar aspectos relevantes sobre a experiência dos usuários em estacionamentos, especialmente em relação à frequência de uso, dificuldade na localização de vagas e aceitação da solução proposta.

Figura 18 – Frequência de uso de estacionamentos pelos respondentes



Fonte: Os autores.

Observa-se que 50,0% dos respondentes utilizam estacionamentos com frequência, sendo 28,95% diariamente e 21,05% semanalmente. Em contrapartida, 42,11% afirmaram utilizar raramente e 7,89% mensalmente. Esse resultado indica que a amostra inclui usuários com experiência prática relevante, fortalecendo a confiabilidade dos dados coletados.

Figura 19 – Tempo médio de procura por vaga

Fonte: Os autores.

Verifica-se que 45,95% dos respondentes levam mais de 5 minutos para encontrar uma vaga, enquanto 24,32% levam entre 3 e 5 minutos e 27,03% entre 1 e 3 minutos. Apenas 2,7% conseguem encontrar uma vaga em menos de 1 minuto. Esses dados evidenciam que o tempo de procura é elevado para a maioria dos usuários.

Figura 20 – Frequência de sucesso ao encontrar vaga na primeira tentativa

Fonte: Os autores.

Apenas 5,41% dos usuários afirmaram encontrar vaga na primeira tentativa, enquanto 51,35% relataram não conseguir e 43,24% indicaram que isso ocorre apenas ocasionalmente. Esse resultado demonstra baixa eficiência no processo atual de localização e ocupação das vagas.

Figura 21 – Nível de satisfação com a organização do estacionamento

Fonte: Os autores.

Observa-se que 29,73% dos usuários estão insatisfeitos, sendo 8,11% muito insatisfeitos e 21,62% insatisfeitos. Além disso, 37,84% se mostram indiferentes e apenas 32,43% satisfeitos. Esse cenário evidencia margem significativa para melhorias na organização dos estacionamentos.

Figura 22 – Desistência de estacionar devido à dificuldade

Fonte: Os autores.

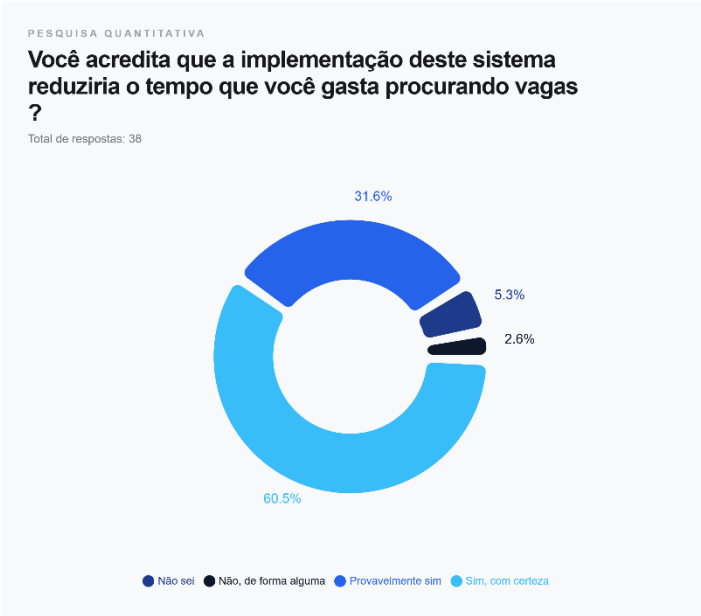
Destaca-se que 70,27% dos respondentes já desistiram de estacionar devido à dificuldade em encontrar vagas. Esse dado demonstra que o problema não se limita ao desconforto do usuário, mas pode afetar diretamente o uso do serviço e a eficiência do estacionamento.

Figura 23 – Percepção de utilidade de um sistema de monitoramento

Fonte: Os autores.

A maioria dos respondentes, correspondente a 62,16%, considera útil a existência de um sistema capaz de informar a disponibilidade de vagas em tempo real. Outros 13,51% se mostram indiferentes, enquanto apenas 5,4% apresentam avaliação negativa. Esse resultado indica elevada aceitação da solução proposta.

Figura 24 – Redução do tempo de procura com o sistema



Fonte: Os autores.

Verifica-se que 92,11% dos respondentes acreditam que o sistema reduziria o tempo de busca por vagas, sendo 60,53% com certeza e 31,58% provavelmente. Esse resultado reforça a viabilidade prática da solução proposta, pois demonstra que os usuários reconhecem o potencial do sistema para melhorar sua experiência.

Figura 25 - Impacto na fluidez do trânsito interno

Fonte: Os autores.

Observa-se que 86,84% dos respondentes concordam que o sistema melhoraria a fluidez do trânsito interno, sendo 57,89% concordando totalmente e

28,95% parcialmente. Esse dado indica que a proposta não é percebida apenas como uma ferramenta de consulta individual, mas também como um recurso capaz de contribuir para a organização do fluxo interno do estacionamento.

4.3 ANÁLISE DOS RESULTADOS

A análise dos dados evidencia que a dificuldade na localização de vagas é um problema recorrente e significativo. O elevado tempo de procura, associado à baixa taxa de sucesso na primeira tentativa, demonstra que muitos usuários enfrentam ineficiência no processo atual de busca por vagas.

O índice de desistência de 70,27% reforça a relevância da problemática, pois indica que a dificuldade em encontrar vagas pode levar o usuário a abandonar o uso do estacionamento. Esse fator impacta tanto a experiência do condutor quanto a eficiência operacional do espaço, uma vez que a circulação interna sem direcionamento tende a aumentar o fluxo de veículos e a desorganização do ambiente.

A percepção dos usuários em relação à solução proposta é majoritariamente positiva. A maior parte dos respondentes considera útil um sistema de monitoramento em tempo real e acredita que ele poderia reduzir o tempo gasto na busca por vagas. Além disso, a maioria também reconhece que a solução poderia melhorar a fluidez do trânsito interno.

Dessa forma, os dados obtidos não apenas validam a existência do problema, como também demonstram aceitação e viabilidade da solução proposta pelo SGVE.

4.4 RELAÇÃO COM ESTUDOS SOBRE ESTACIONAMENTO INTELIGENTE

Os resultados obtidos na pesquisa aplicada aos usuários estão alinhados com estudos sobre sistemas inteligentes de estacionamento. Khan et al. (2020), ao analisarem a solução ParkUs, demonstram que a disponibilização de informações sobre a disponibilidade de vagas pode contribuir para a redução do tempo de busca por estacionamento. No estudo, a solução utiliza um aplicativo móvel e um mapa de calor para indicar áreas com maior probabilidade de vagas disponíveis, auxiliando o motorista na tomada de decisão. Os autores observaram que, quando havia dados disponíveis no mapa de calor, a mediana do tempo de busca foi reduzida de 150 segundos para pouco mais de 100 segundos, representando uma redução aproximada de 33% no tempo de procura.

Embora o ParkUs utilize uma abordagem diferente da proposta neste trabalho, baseada em smartphones, GPS e dados colaborativos dos usuários, o princípio central é semelhante ao do SGVE: coletar informações sobre ocupação e apresentá-las ao usuário de forma visual e acessível. No caso do SGVE, essa coleta é realizada por sensores infravermelhos conectados ao ESP32, enquanto a apresentação das informações ocorre por meio de uma interface web que exibe o estado das vagas em tempo real.

Essa relação reforça a fundamentação da proposta, pois demonstra que sistemas capazes de informar a disponibilidade de vagas podem reduzir a incerteza do usuário durante a procura por estacionamento. A redução aproximada de 33% observada no estudo do ParkUs indica que a apresentação de dados de ocupação pode gerar impacto direto na eficiência da busca por vagas. Assim, o SGVE aplica esse mesmo princípio em um contexto controlado, com infraestrutura de baixo x'

4.5 VALIDAÇÃO DA PROPOSTA

Com base nos resultados obtidos, é possível afirmar que a proposta do SGVE apresenta aplicabilidade prática. A solução atende diretamente às necessidades identificadas na pesquisa, especialmente no que se refere à redução do tempo de procura por vagas, à melhoria da organização do estacionamento e ao apoio à tomada de decisão do usuário.

Os dados coletados corroboram a hipótese inicial do trabalho, evidenciando que a utilização de tecnologias baseadas em Internet das Coisas (IoT) pode contribuir para a otimização da gestão de vagas. A integração entre sensores, ESP32, backend e interface web permite que informações de ocupação sejam coletadas, processadas e exibidas em tempo real, reduzindo a incerteza na busca por vagas.

Conclui-se, portanto, que o sistema proposto aborda um problema real, apresenta aceitação por parte dos usuários e possui fundamentação compatível com estudos sobre estacionamento inteligente. Dessa forma, o SGVE demonstra relevância e viabilidade como solução tecnológica para o monitoramento e gerenciamento de vagas de estacionamento.

5 Coniderações finais

O desenvolvimento do SGVE permitiu a criação de uma solução baseada em Internet das Coisas (IoT) para monitoramento de vagas de estacionamento em tempo real, integrando sensores infravermelhos, microcontrolador ESP32, backend em PHP, banco de dados MariaDB e interface web. A arquitetura adotada, estruturada em rede local com comunicação sem fio entre o ESP32 e o servidor, demonstrou ser adequada ao contexto proposto, oferecendo baixa latência, simplicidade de implantação e controle centralizado das informações.

Os resultados obtidos na pesquisa aplicada reforçam a relevância da problemática abordada. A dificuldade na localização de vagas, o elevado tempo de procura e o índice de desistência identificado evidenciam limitações presentes em estacionamentos convencionais. Em contrapartida, a elevada aceitação da solução proposta pelos respondentes demonstra que sistemas de monitoramento em tempo real são percebidos como ferramentas capazes de melhorar a experiência do usuário e contribuir para a organização do fluxo interno.

A implementação do protótipo em maquete permitiu validar o funcionamento da comunicação entre hardware, backend, banco de dados e interface web, demonstrando a viabilidade técnica da solução em ambiente controlado. Além disso, os resultados observados mostraram alinhamento com estudos relacionados a estacionamentos inteligentes, indicando que a disponibilização de informações em tempo real pode contribuir para a redução do tempo de busca por vagas e para a melhoria da eficiência operacional dos estacionamentos.

Dessa forma, conclui-se que o SGVE atingiu os objetivos propostos no contexto do protótipo desenvolvido. Como trabalhos futuros, sugere-se a ampliação da acessibilidade do sistema para os usuários do estacionamento por meio de aplicações para dispositivos móveis, permitindo a visualização remota da disponibilidade das vagas em tempo real. Além disso, podem ser implementadas funcionalidades como reserva antecipada de vagas, notificações automáticas e integração entre o sistema web e aplicativos móveis, ampliando a praticidade e a experiência do usuário.

REFERÊNCIAS

BENTO, Evaldo Junior. Desenvolvimento web com PHP e MySQL. São Paulo: Casa do Código, 2017.

BIFFI, Kelvin Baumhardt. JavaScript: básico ao avançado. [S. l.]: Kelvin Baumhardt Biffi, 2021. Disponível em: https://www.kufunda.net/publicdocs/Javascript%20B%C3%A1sico%20ao%20Avan%C3%A7ado%20%28Kelvin%20Baumhardt%20Biffi%29.pdf. Acesso em: 7 mar. 2026.

BRIONES, Fernie. Level of Service for Parking Facilities Based on Search Time. 2021. 66 f. Dissertação (Master of Science em Civil Engineering) – The University of Texas at El Paso, El Paso, 2021. Disponível em: https://scholarworks.utep.edu/open_etd/3391. Acesso em: 28 fev. 2026.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de banco de dados. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

FACHINI, Moisés Panegassi; MESQUITA, Nathalia Pinheiro; OLIVEIRA, Rafael Padovani; FRANÇA, Patricia Gallo de. Internet das Coisas: uma breve revisão bibliográfica. Conexões: Ciência e Tecnologia, Fortaleza, v. 11, n. 6, p. 85–90, dez. 2017. DOI: 10.21439/conexoes.v11i6.1007.

FAHIM, Abrar; HASAN, Mehedi; CHOWDHURY, Muhtasim Alam. Smart parking systems: comprehensive review based on various aspects. Heliyon, v. 7, n. 5, e07050, 2021. Disponível em: https://doi.org/10.1016/j.heliyon.2021.e07050. Acesso em: 2 mar. 2026.

IBM. O que é uma API REST (API RESTful)? IBM Think, 2023. Disponível em: https://www.ibm.com/br-pt/think/topics/rest-apis. Acesso em: 4 mar. 2026.

JOY-IT. SBC-NodeMCU-ESP32. Disponível em: https://joy-it.net/en/products/SBC-NodeMCU-ESP32. Acesso em: 3 mar. 2026.

K19 TREINAMENTOS. Desenvolvimento Web com HTML, CSS e JavaScript. São Paulo: K19 Treinamentos, 2013. Material didático. Acesso em: 7 mar. 2026.

KHAN, Aftab et al. Reducing Parking Space Search Time and Environmental Impacts: A Technology Driven Smart Parking Case Study. IEEE Technology and Society Magazine, v. 39, n. 3, p. 62-75, set. 2020. DOI: 10.1109/MTS.2020.3012329. Disponível em: https://doi.org/10.1109/MTS.2020.3012329. Acesso em: 12 abr. 2026.

LAST MINUTE ENGINEERS. ESP32 Pinout Reference. Disponível em: https://lastminuteengineers.com/esp32-pinout-reference/. Acesso em: 3 mar. 2026.

MARIADB FOUNDATION. MariaDB. 2023. Disponível em: https://mariadb.org/. Acesso em: 10 mar. 2026.

MITSUHASHI, Marlos Kenjy; CAVAMURA, Humberto Fernando Massaharu. Sistema de gerência de vagas de estacionamento. 2014. 107 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Industrial Elétrica com ênfase em Eletrônica e Telecomunicações) – Universidade Tecnológica Federal do Paraná, Curitiba, 2014.

NASCIMENTO E SILVA, Daniel; BREMGARTNER DA FROTA, Vitor; DE JESUS DOS SANTOS, Alyson. Internet das Coisas Arquiteturas Teóricas e Metodológicas. [S. l.: s. n.], 2023.

PEREIRA LIMA FRANÇA, Cicero Tadeu; CELESTINO JÚNIOR, Joaquim. Computação-Banco de Dados. Fortaleza - Ceará: Editora da Universidade Estadual do Ceará, 2015.

PHP. PHP: Hypertext Preprocessor. Disponível em: https://www.php.net/. Acesso em: 7 mar. 2026.

RALL, Ricardo; LEITE, Luan Guilherme da Silva; MIRANDA, Davi Rodrigo de. Protótipo de domótica com microcontrolador ESP32. Revista EduFatec: educação, tecnologia e gestão, Franca, v. 2, n. 6, ago./dez. 2023. Disponível em: https://revistaedufatec.fatecfranca.edu.br. Acesso em: 3 mar. 2026.

SOUZA, Isaac Felisberto de. Camadas. Guia.dev, 01 fev. 2021. Disponível em: https://guia.dev/pt/pillars/software-architecture/layers.html. Acesso em: 03 mar. 2026.