UNIVERSIDADE
FEDERAL DO CEARÁ

Protótipo de um Estacionamento Inteligente

Equipe:

Beatriz de Sousa Alves - 553556

Maria Eduarda Almeida Rodrigues - 552819

Nathalia de Oliveira Lima - 552319

Pablo Brandão Passos - 539730

Vitória das Graças Silva - 557364

JUNHO DE 2025

Sumário

Introdução

O domínio do projeto está na automação de estacionamentos, com foco na gestão inteligente de vagas e no controle de acesso veicular. O sistema será desenvolvido para resolver problemas comuns em estacionamentos públicos e privados, como a falta de informação em tempo real sobre vagas disponíveis, processos manuais de controle de entrada/saída e a ineficiência na gestão do fluxo de veículos.

O escopo inicial do sistema inclui: Detecção automática da ocupação de vagas através de sensores; Controle automatizado de cancelas de entrada e saída; Interface web em tempo real mostrando vagas disponíveis; Painel informativo na entrada do estacionamento; Registro histórico de movimentação de veículos; Sistema de alertas para administradores.

O sistema será projetado para estacionamentos de pequeno e médio porte (até 100 vagas), com possibilidade de expansão para maiores capacidades. A solução visa substituir completamente os sistemas manuais atuais, proporcionando maior eficiência operacional, redução de custos e melhor experiência para os usuários.

Entrevista

Roteiro da Entrevista

1. Quantas vagas de estacionamento o sistema precisa gerenciar?
2. O sistema será utilizado por funcionários, clientes, visitantes ou todos os tipos de usuários?
3. Como é feito hoje o controle de entrada e saída dos veículos?
4. Haverá necessidade de integração com dispositivos físicos, como cancelas, sensores, câmeras OCR ou leitores de placa?
5. Deseja que o sistema registre horários de entrada/saída e calcule o valor a ser pago automaticamente?
6. Será cobrado a estadia no estacionamento?
7. Deseja que o sistema tenha funcionalidades administrativas, como relatórios, controle de fluxo, ou exportação de dados?
8. Usuários precisarão fazer reservas de vagas antecipadamente ou o controle será apenas por ordem de chegada?
9. Haverá diferentes tipos de acesso (ex: mensalistas, visitantes, funcionários, VIP)?
10. Qual será o ambiente de uso do sistema: web, mobile, totem local, todos?

11. Quais funcionalidades considera essenciais no sistema desde o início (requisitos obrigatórios)?

12. Existem processos manuais hoje que o sistema deverá automatizar? Quais?

13. Possui alguma preferência por tecnologias ou plataformas específicas (ex: banco de dados, linguagem, sistema operacional)?

14. Quem são os usuários do sistema e qual é o nível de familiaridade deles com a tecnologia?

15. Existem normas legais, fiscais ou de acessibilidade que o sistema deve seguir?

16. Deseja relatórios personalizados? Se sim, que tipo de informações devem ser apresentadas?

17. Qual é o prazo desejado para desenvolvimento e implantação do sistema?

18. Como será feito o suporte ao sistema após a implantação? O cliente espera manutenção contínua?

19. Existe algum sistema concorrente ou referência que o cliente gostaria de se basear ou evitar?

20. Quais são os principais problemas que o cliente quer resolver com o novo sistema?

Respostas da Entrevista

Perguntas Realizadas: 1 a 20

1. Duas vagas.

2. Clientes e visitantes.

3. Não tem controle.

4. Sim.

5. Sim.

6. Sim.

7. Não.

8. Por ordem de chegada.

9. Não.

10. Web.

11. O sistema deve detectar a presença de veículos nas vagas por meio de sensores, o sistema deve disponibilizar uma interface web acessível para visualização das vagas ocupadas e disponíveis.

12. Abertura e fechamento da cancela.

13. Não.

14. Cliente, nível médio a alto.

15. Não.

16. Sim, horários de entrada e saída.

17. 4 meses.

18. Ideal seria suporte após entrega e manutenção apenas quando solicitada.

19. Não.

20. Automatização do estacionamento e acesso ao sistema web que possibilite ver a quantidade de vagas.

Requisitos Funcionais

RF01 *- O sistema deve detectar a presença de veículos nas vagas por meio de sensores.*
RF02 *- O sistema deve abrir automaticamente a cancela quando um veículo se aproxima da entrada e há vaga disponível.*
RF03 *- O sistema deve impedir a entrada de veículos se todas as vagas estiverem ocupadas.*
RF04 *- O sistema deve registrar a entrada e a saída de cada veículo.*
RF05 *- O sistema deve atualizar em tempo real a quantidade de vagas disponíveis.*
RF06 *- O sistema deve disponibilizar uma interface web acessível para visualização das vagas ocupadas e disponíveis.*
RF07 *- O sistema deve exibir em tempo real na interface web o status de cada vaga (livre/ocupada).*
RF08 *- O sistema deve exibir uma mensagem na interface web alertando que o estacionamento está cheio quando todas as vagas estiverem ocupadas.*
RF09 *- O sistema deve possuir um painel local (display LCD ou LED) informando o número de vagas disponíveis na entrada do estacionamento.*
RF10 *- O sistema deve controlar automaticamente a cancela de saída.*

Requisitos Não Funcionais

RNF01 *- O sistema deve ter resposta em tempo real para atualizar o status das vagas na interface web.*
RNF02 *- O sistema deve ser acessível via dispositivos móveis e desktops, com interface responsiva.*
RNF03 *- O sistema deve manter a disponibilidade durante o horário de funcionamento do estacionamento.*
RNF04 *- A comunicação entre sensores e o sistema central deve ser segura e imune a interferências.*
RNF05 *- O código fonte do sistema deve ser modular para facilitar futuras atualizações*

RNF06 *- Deve haver controle de posição para garantir que a cancela nunca fique entre aberta por falha.*

Histórias de Usuário

Nas histórias de usuário, buscamos mapear o maior número possível de cenários relacionados à operação do estacionamento, considerando diferentes perfis de usuários e variações no fluxo de entrada, saída e consulta de vagas. Além dos fluxos padrão, também contemplamos situações excepcionais, como falhas em sensores ou indisponibilidade do sistema, que exigem comportamentos diferenciados por parte da aplicação.

As histórias mais relevantes foram destacadas com base em uma votação interna, seguindo os critérios da métrica INVEST (Independente, Negociável, Valiosa, Estimável, Pequena e Testável). No entanto, todas as histórias descritas representam requisitos importantes e refletem contextos reais que o sistema deve ser capaz de atender.

US001	
Cartão	Como motorista, quero ver na entrada do estacionamento quantas vagas estão disponíveis para saber se posso entrar.
Conversa	● As vagas serão exibidas em um painel físico ou via aplicativo? ● A atualização das vagas será em tempo real? ● O que deve acontecer se o número de vagas for zero? ● É necessário indicar a localização das vagas livres ou apenas a quantidade? ● Como os sensores identificam a ocupação das vagas?
Confirmação	● Um display visível na entrada do estacionamento exibe o número total de vagas disponíveis. ● O número de vagas disponíveis é atualizado em tempo real conforme entrada e saída de veículos. ● Se não houver vagas, uma mensagem como “Estacionamento Lotado” deve ser exibida. ● O sistema deve funcionar corretamente com sensores simulados ou reais. ● A contagem de vagas é precisa com margem de erro inferior a 1 vaga. **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✓ **T** - ✓

US002	
Cartão	Como motorista, quero que a cancela abra automaticamente quando houver vaga disponível, para não precisar sair do carro.
Conversa	● Como o sistema identifica se há vaga disponível (sensor, sistema central)? ● Como o carro será detectado (sensor de presença, leitura de placa, cartão RFID)? ● A cancela deve permanecer fechada se não houver vagas? ● É necessário algum tipo de autenticação antes da abertura? ● Qual o tempo de resposta esperado entre a detecção e a abertura da cancela?
Confirmação	● A cancela se abre automaticamente quando houver pelo menos uma vaga disponível e um carro for detectado na entrada. ● A cancela não abre se o estacionamento estiver lotado. ● O tempo entre a detecção do carro e a abertura da cancela não ultrapassa 3 segundos. ● O sistema registra cada abertura da cancela para fins de auditoria. ● O funcionamento é testado com diferentes simulações de entrada (vagas disponíveis / lotadas). **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✓ **T** - ✓

US003	
Cartão	Como motorista, quero ser impedido de entrar se não houver vagas, para evitar frustrações.
Conversa	● Como o sistema irá impedir a entrada? (Ex: mantendo a cancela fechada, sinalização visual ou sonora?) ● O sistema deve informar o motivo para a entrada negada? ● Como o sistema lida com erros na contagem de vagas (ex: erro de sensor)? ● É necessário ter uma fila ou zona de espera para os veículos que chegarem sem vaga? ● O sistema deve registrar tentativas de entrada com o estacionamento lotado?
Confirmação	● Se não houver nenhuma vaga disponível, a cancela permanece fechada. ● Um aviso claro (painel ou mensagem) informa: “Estacionamento Lotado”. ● Nenhum comando de abertura é enviado para a cancela quando o estacionamento está lotado. ● O sistema impede múltiplas tentativas de entrada indevidas durante o tempo de lotação. ● O funcionamento é testado simulando a ausência de vagas. **I** - ✗ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✓ **T** - ✓

US004	
Cartão	Como motorista, quero acessar uma página web e ver quais vagas estão livres, antes de chegar ao estacionamento.
Conversa	● A página mostrará quantidade de vagas ou um mapa visual com vagas específicas? ● A informação será atualizada em tempo real ou com intervalo? ● A página será responsiva para dispositivos móveis? ● Será necessário autenticar o usuário ou o acesso será público? ● Haverá diferenciação por tipo de vaga (carro comum, PCD, moto etc.)?
Confirmação	● A página web exibe o número total de vagas livres no estacionamento. ● A informação é atualizada no máximo a cada 10 segundos. ● A página pode ser acessada de dispositivos móveis sem falhas de visualização. ● Caso todas as vagas estejam ocupadas, é exibida a mensagem: “Estacionamento Lotado”. ● Testes confirmam que os dados exibidos refletem corretamente a ocupação em tempo real. **I - ✓ N - ✓ V - ✓ E - ✓ S - ✓ T- ✓**

US005	
Cartão	Como motorista, quero que a saída seja automática após estacionar, para maior comodidade.
Conversa	● Como o sistema detecta que o motorista deseja sair? (Sensor de presença, leitura de placa, botão no app?) ● Haverá algum controle para evitar saídas indevidas ou fraudes? ● A cancela abre automaticamente sempre que um carro for detectado na saída? ● O sistema precisa registrar o horário de saída e o tempo de permanência? ● O pagamento (se houver) será integrado antes da liberação da saída?
Confirmação	● Ao ser detectado na zona de saída, o veículo tem sua presença verificada e a cancela abre automaticamente. ● O tempo de resposta entre a detecção e a abertura da cancela não ultrapassa 3 segundos. ● O sistema registra a saída do veículo para controle interno. ● Se o sistema de controle exigir, a saída só ocorre após a verificação de pendências (ex: pagamento). ● Testes confirmam que a saída ocorre de forma fluida sem intervenção humana. **I- ✗ N- ✓ V- ✓ E- ✓ S- ✓ T- ✓**

US006	
Cartão	Como motorista, quero saber em tempo real se o estacionamento está cheio ou liberado para entrada.
Conversa	● Onde essa informação será exibida? (Painel na entrada, app, site, semáforo?) ● Qual é a definição de "liberado"? (Há pelo menos uma vaga ou mais de um número mínimo?) ● Qual será a frequência de atualização da informação? ● A informação considera apenas as vagas comuns ou também específicas (PCD, moto, etc.)? ● O sistema continua funcionando em caso de falha de sensores ou comunicação?
Confirmação	● O sistema exibe, em tempo real, o status “Liberado” ou “Cheio”. ● O status é atualizado automaticamente conforme a entrada e saída de veículos. ● A informação é visível ao motorista antes da entrada no estacionamento. ● Testes simulando entrada e saída de veículos atualizam corretamente o status. ● Em caso de sistema lotado, é exibido um aviso claro: “Estacionamento Cheio”. **I -✓ N - ✓ V - ✓ E - ✓ S - ✓ T - ✓**

US007	
Cartão	Como administrador, quero ver um painel com o histórico de ocupações diárias para analisar o fluxo de veículos.
Conversa	● Gráficos de ocupação por hora/dia/semana; ● Filtros por período e tipo de dia (útil, fim de semana, feriado); ● Exportação para Excel/PDF; ● Identificação de horários de pico; ● Comparativo com períodos anteriores ● Alertas para padrões incomuns
Confirmação	● Gráficos gerados corretamente; ● Filtros funcionando como esperado; ● Exportação gerando arquivos utilizáveis; ● Identificação automática de picos; ● Comparativos precisos; ● Alertas configuráveis; **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✗ **T** - ✓

US008	
Cartão	Como administrador, quero acessar remotamente os sensores para verificar o estado operacional.
Conversa	● Status online/offline de cada sensor ● Histórico de funcionamento ● Alertas para sensores com problemas ● Mapa mostrando localização física dos sensores ● Tempo de atividade (uptime) de cada componente ● Interface web segura com autenticação
Confirmação	● Status de todos os sensores visíveis ● Histórico armazenado por pelo menos 30 dias ● Alertas configuráveis por email/SMS ● Mapa preciso do layout físico ● Autenticação segura implementada ● Testado com múltiplos sensores simultâneos **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✗ **T** - ✓

US009	
Cartão	Como administrador, quero reiniciar remotamente os sensores em caso de falha.
Conversa	● Qual será o meio utilizado para o reinício remoto? (Painel web, app, terminal?) ● Os sensores enviam algum status indicando falha? ● O reinício será individual por sensor ou em grupo? ● Será necessário autenticação para executar essa ação? ● Haverá log de quando e quem executou o reinício?
Confirmação	● O administrador consegue acessar uma interface remota e visualizar o status dos sensores. ● Ao identificar falhas, é possível selecionar o sensor e reiniciá-lo remotamente. ● Após o comando, o sensor volta ao estado de operação normal (se funcional). ● A ação de reinício é registrada em log com data, hora e usuário. ● Testes simulam falha e confirmam que o comando remoto restabelece a função do sensor. **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US010	
Cartão	Como administrador, quero receber alertas de falha nos sensores para agir rapidamente.
Conversa	● Notificações por email e/ou SMS ● Níveis de alerta (informação, aviso, crítico) ● Possibilidade de configurar horários para notificações ● Lista de destinatários configurável ● Template de mensagem personalizável ● Opção de silenciar alertas temporariamente
Confirmação	● Alertas sendo enviados pelos canais configurados ● Diferenciação entre níveis de alerta ● Configuração de horários funcionando ● Lista de destinatários editável ● Personalização de mensagens possível ● Silenciamento temporário implementado **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✗ **T** - ✓

US011	
Cartão	Como administrador, quero ter controle de acesso à interface de administração para garantir a segurança.
Conversa	● Autenticação obrigatória com usuário e senha; ● Níveis de acesso (operador, administrador, superusuário); ● Registro de todas as ações administrativas; ● Bloqueio após tentativas fracassadas; ● Expiração de senha periódica; ● Possibilidade de autenticação em dois fatores
Confirmação	● Autenticação obrigatória implementada; ● Diferentes níveis de acesso funcionando; ● Log de atividades sendo gerado; ● Bloqueio após tentativas configurável; ● Expiração de senha operacional; ● 2FA implementado (opcional) **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✗ **T** - ✓

US012	
Cartão	Como administrador, quero poder gerar relatórios semanais do uso do estacionamento.
Conversa	● Relatório padrão com ocupação média, picos, receita (se aplicável); ● Personalização de colunas e métricas; ● Agendamento de envio automático por email; ● Comparativo com semanas anteriores; ● Gráficos e tabelas exportáveis; ● Marcadores para eventos especiais (feriados, etc.)
Confirmação	● Relatório padrão gerando dados corretos; ● Personalização de campos funcionando; ● Agendamento automático operacional; ● Comparativos precisos; ● Exportação para formatos comuns; ● Marcadores de eventos visíveis **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓ **S** - ✗ **T** - ✓

US013	
Cartão	Como visitante, quero acessar a interface web e ver em tempo real as vagas ocupadas e livres.
Conversa	● Atualização automática sem necessidade de refresh; ● Visualização simples e intuitiva; ● Legenda clara para os status; ● Funcionamento em navegadores modernos; ● Tempo de carregamento inferior a 3 segundos; ● Versão mobile otimizada
Confirmação	● Atualização automática implementada; ● Interface testada com usuários reais; ● Legenda clara e visível; ● Compatibilidade com principais navegadores; ● Performance dentro do esperado; ● Versão mobile validada **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US014	
Cartão	Como visitante, quero que a página web seja responsiva para visualizar em qualquer dispositivo.
Conversa	● Adaptação a telas de 5" a 24"; ● Layout reorganizado conforme tamanho da tela; ● Elementos touch com tamanho adequado em mobile; ● Performance consistente em diferentes dispositivos; ● Testado em iOS, Android e desktop; ● Nenhum elemento cortado ou sobreposto
Confirmação	● Testado em pelo menos 5 tamanhos de tela diferentes; ● Reorganização do layout funcionando; ● Elementos touch com tamanho adequado; ● Performance consistente; ● Compatibilidade cruzada verificada; ● Nenhum problema de sobreposição **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US015	
Cartão	Como visitante, quero atualizar os dados sem recarregar a página.
Conversa	● Atualização a cada 10 segundos no máximo; ● Indicador visual discreto durante atualização; ● Manutenção da posição de scroll; ● Nenhum flicker ou piscar de tela; ● Funcionamento mesmo em conexões lentas; ● Fallback para refresh manual se tecnologia não suportada
Confirmação	● Intervalo de atualização respeitado; ● Indicador de atividade visível; ● Posição do scroll mantida; ● Transições suaves; ● Testado em conexão 3G simulada; ● Fallback implementado **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US016	
Cartão	Como visitante, quero que o mapa das vagas seja fácil de entender e visualmente claro.
Conversa	● Representação fiel do layout físico; ● Cores contrastantes para vagas livres/ocupadas; ● Legenda sempre visível; ● Zoom sem perda de qualidade; ● Navegação intuitiva; ● Acessível para daltônicos
Confirmação	● Mapa corresponde ao layout real; ● Cores com contraste adequado; ● Legenda acessivel; ● Zoom funcionando corretamente; ● Testado com usuários reais; ● Opções para daltônicos **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US017	
Cartão	Como visitante, quero ser informado se o sistema estiver fora do ar.
Conversa	● Mensagem clara sobre o status; ● Tempo estimado para retorno (se conhecido); ● Alternativas (como contato telefônico); ● Design que não sugere erro do usuário; ● Página de status separada, se necessário; ● Restauração automática da funcionalidade quando resolvido;
Confirmação	● Mensagens claras e não técnicas; ● Estimativas quando possível; ● Alternativas fornecidas; ● Design amigável; ● Página de status dedicada; ● Recuperação automática testada **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US018	
Cartão	Como técnico, quero ver os logs do sistema para diagnosticar falhas nos sensores.
Conversa	● Acesso seguro aos logs completos; ● Filtros por data, tipo de evento, sensor; ● Exportação para análise externa; ● Correlação entre eventos; ● Marcadores para eventos críticos; ● Busca textual nos logs
Confirmação	● Acesso apenas para técnicos; ● Filtros funcionando corretamente; ● Exportação gerando arquivos válidos; ● Correlação entre eventos implementada; ● Marcadores visíveis; ● Busca textual operacional **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

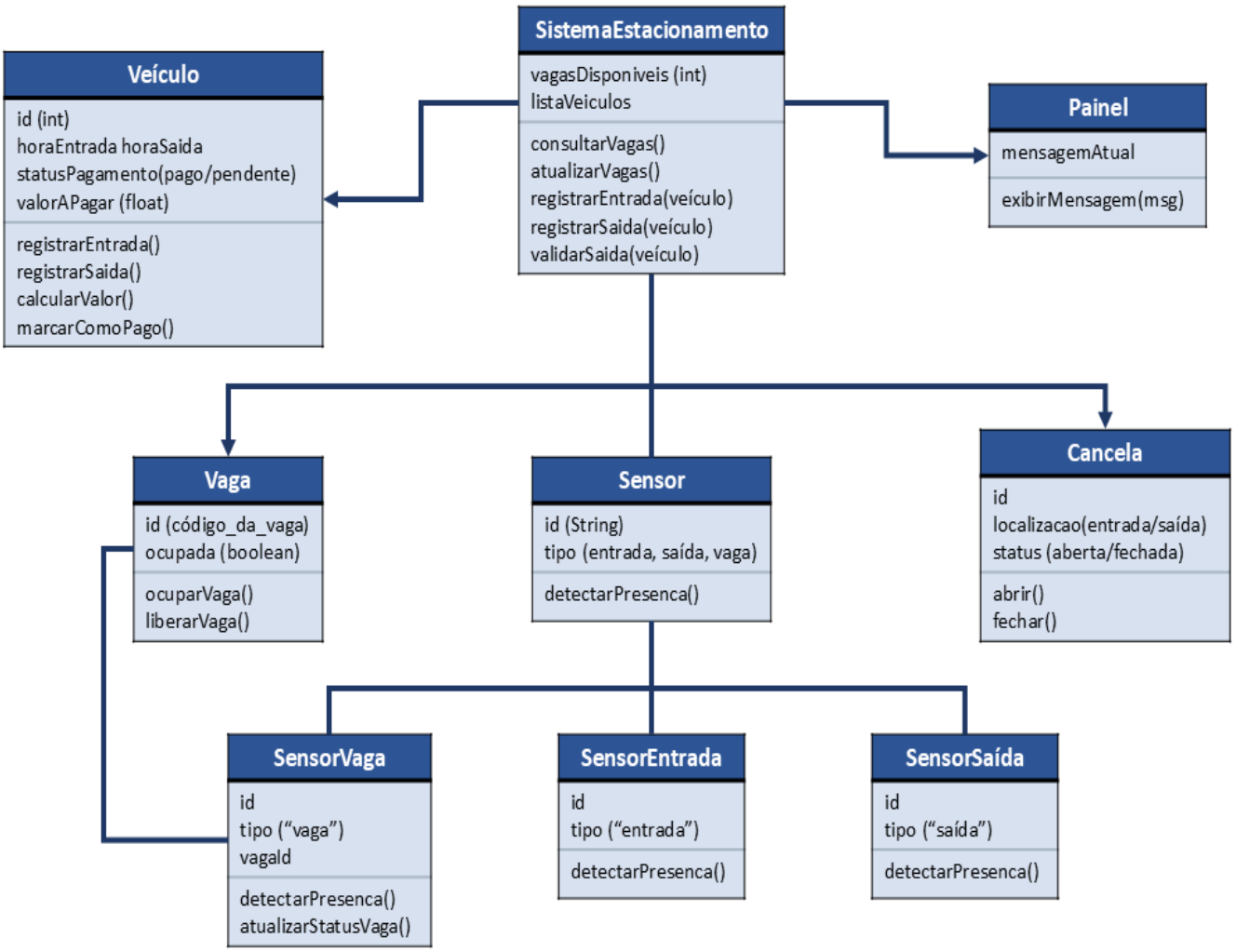
US019	
Cartão	Como técnico, quero simular a entrada e saída de veículos para testes.
Conversa	● Interface dedicada para testes; ● Simulação individual ou em massa; ● Parâmetros configuráveis (intervalos, etc.); ● Não afetar dados reais de produção; ● Log detalhado das simulações; ● Integração com ambiente de testes
Confirmação	● Interface separada da produção; ● Simulações individuais e em massa; ● Parâmetros ajustáveis; ● Isolamento dos dados reais; ● Logs de teste gerados; ● Ambiente de testes integrado **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

US020	
Cartão	Como técnico, quero visualizar qual sensor está apresentando problemas.
Conversa	● Destaque visual para sensores com falha; ● Detalhes do último estado conhecido; ● Histórico de falhas do sensor; ● Localização física no mapa; ● Priorização por criticidade; ● Ações rápidas (reiniciar, desativar)
Confirmação	● Destaque visual claro; ● Detalhes do estado disponíveis; ● Histórico de falhas acessível; ● Mapa de localização preciso; ● Priorização funcionando; ● Ações rápidas implementadas **I** - ✓ **N** - ✓ **V** - ✓ **E** - ✓**S** - ✓ **T** - ✓

Diagrama de Classes

O **Diagrama de Classes** representa as principais entidades envolvidas no processo automatizado de controle de estacionamento.

Abaixo está o Diagrama de Classes com atributos, métodos e os relacionamentos entre as classes:



Classe: Veículo

- **Atributos:**
    - horaEntrada e horaSaida: controlam os horários de entrada e saída.
- **Métodos:**
    - registrarEntrada(), registrarSaida(): armazenam os horários no sistema.

Classe: Sensor *(Classe genérica/pai)*

- **Atributos:**
    - id: identificação do sensor.
    - tipo: tipo de sensor (ex: entrada, saída, vaga).
- **Métodos:**
    - registrarPresenca(): detecta presença de um veículo.

As classes **SensorEntrada**, **SensorSaida** e **SensorVaga** herdam de **Sensor**:

🔹 **SensorEntrada**

- Detecta veículos chegando ao estacionamento.

🔹 **SensorSaida**

- Detecta veículos saindo.

🔹 **SensorVaga**

- Detecta se uma vaga está ocupada ou livre.
- Atributo extra: vagaId.

Classe: Vaga

- **Atributos:**
    - id: código da vaga.
    - ocupada: indica se está sendo usada.
- **Métodos:**
    - ocuparVaga(), liberarVaga(): mudam o status da vaga.

Classe: SistemaEstacionamento

- Classe principal que gerencia o sistema.
- **Atributos:**
    - vagasDisponiveis: número de vagas livres.
    - listaVeiculos: lista de veículos presentes.
- **Métodos:**
    - consultarVagas(), atualizarVagas()
    - registrarEntrada(veiculo) / registrarSaida(veiculo)
    - validarSaida(veiculo): verifica se o veículo pode sair (ex: pagamento ok).

Classe: Cancela

- Controla o acesso ao estacionamento.
- **Atributos:**
    - localizacao: "Entrada" ou "Saída".
    - status: "Aberta" ou "Fechada".
- **Métodos:**
    - abrir(), fechar(): controlam a cancela fisicamente.

Classe: Painel

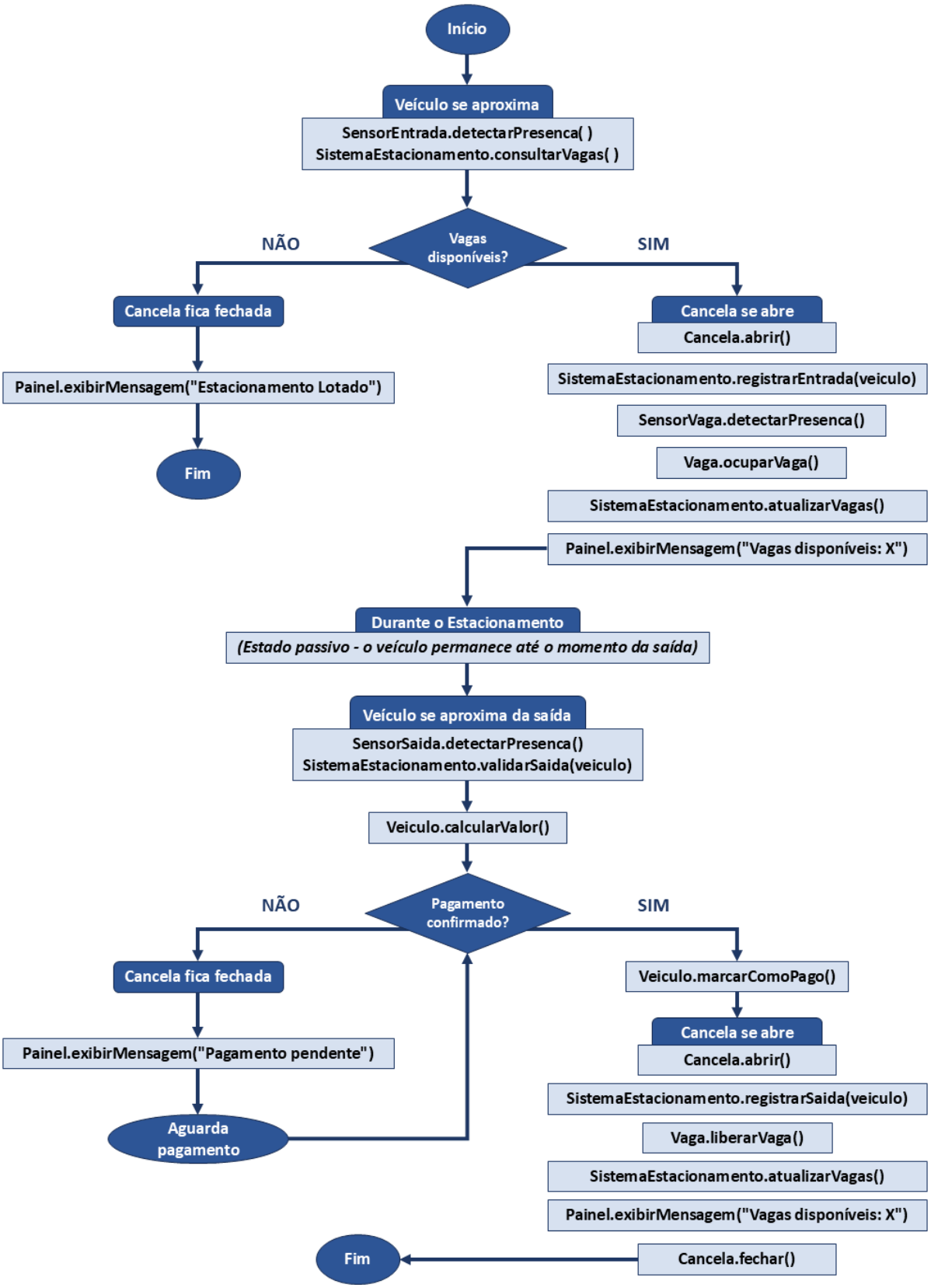
- Mostrar mensagens ao motorista.
- **Atributos:**
    - mensagemAtual: mensagem sendo exibida.
- **Métodos:**
    - exibirMensagem(msg): atualiza o conteúdo no painel.

Relacionamentos

- **Herança**: SensorEntrada, SensorSaida, SensorVaga herdam de Sensor.
- **Associação**:
    - SistemaEstacionamento se relaciona com Veiculo, Vaga, Cancela, Painel.
    - SensorVaga está ligado a Vaga.
    - Painel é usado por SistemaEstacionamento e Sensor.

Diagrama de Atividades

O **Diagrama de Atividades** representa graficamente o fluxo de processos que ocorrem no sistema desde a chegada do veículo até a saída, considerando decisões automáticas, interações com sensores e módulos do sistema. Esse fluxo representa a automação desejada, otimizando a entrada, ocupação e saída de veículos, com mínimo contato humano.


Início
Veículo se aproxima
SensorEntrada.detectarPresenca( )
SistemaEstacionamento.consultarVagas( )
Vagas disponíveis?
NÃO
SIM
Cancela fica fechada
Painel.exibirMensagem("Estacionamento Lotado")
Fim
Cancela se abre
Cancela.abrir()
SistemaEstacionamento.registrarEntrada(veiculo)
SensorVaga.detectarPresenca()
Vaga.ocuparVaga()
SistemaEstacionamento.atualizarVagas()
Painel.exibirMensagem("Vagas disponíveis: X")
Durante o Estacionamento
(Estado passivo - o veículo permanece até o momento da saída)
Veículo se aproxima da saída
SensorSaida.detectarPresenca()
SistemaEstacionamento.validarSaida(veiculo)
Veiculo.calcularValor()
Pagamento confirmado?
NÃO
SIM
Cancela fica fechada
Painel.exibirMensagem("Pagamento pendente")
Aguarda pagamento
Veiculo.marcarComoPago()
Cancela se abre
Cancela.abrir()
SistemaEstacionamento.registrarSaida(veiculo)
Vaga.liberarVaga()
SistemaEstacionamento.atualizarVagas()
Painel.exibirMensagem("Vagas disponíveis: X")
Cancela.fechar()
Fim

Abaixo está o detalhamento das principais atividades divididas em quatro etapas:

1. Detecção de Entrada

- **Início**
- Veículo se aproxima da entrada
- Sensor de entrada detecta presença
- Sistema consulta o número de vagas disponíveis

2. Decisão de Acesso

- [Decisão] Há vaga disponível?
    - **Sim**:
        - Abrir cancela
        - Veículo entra e estaciona
        - Registrar horário de entrada
        - Atualizar banco de dados (vaga ocupada)
        - Exibir número atualizado de vagas
    - **Não**:
        - Manter cancela fechada
        - Exibir aviso “Estacionamento Cheio” no painel
        - Encerrar fluxo

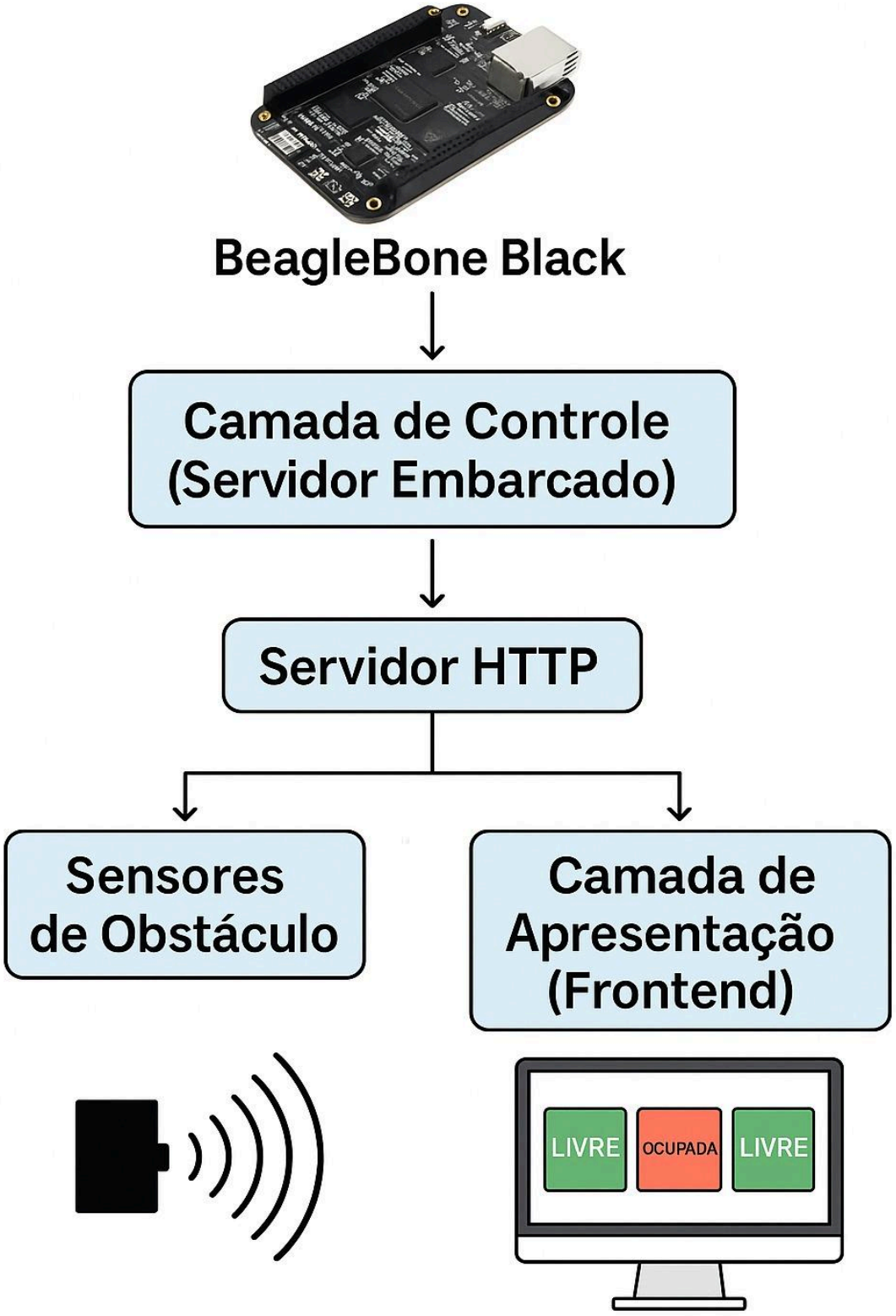
3. Estacionamento

- Vaga é ocupada
- Sensor de vaga marca como **ocupada**
- Sistema atualiza visualização web e display externo

4. Saída do Veículo

- Veículo se aproxima da saída
- Sensor de saída detecta presença
- Sistema valida se o veículo pode sair (se necessário, verificar pendência de pagamento)
- Abrir cancela de saída
- Registrar horário de saída
- Atualizar banco de dados (vaga livre)
- Exibir número atualizado de vagas
- **Fim**

Arquitetura do Sistema



A arquitetura do sistema de estacionamento inteligente é baseada em um modelo **cliente-servidor embarcado**, utilizando a **placa BeagleBone Black** como unidade central de controle. O sistema permite a detecção de veículos nas vagas e a visualização em tempo real do status por meio de uma interface web.

1. Visão Geral

O sistema é dividido em três camadas principais:

- **Camada de Sensoriamento (Hardware):** Responsável por detectar a presença de veículos nas vagas utilizando sensores conectados à BeagleBone Black.

- **Camada de Controle (Servidor embarcado):** Um programa em C executado na BeagleBone Black lê os sensores e fornece uma página HTML com os dados via HTTP.

- **Camada de Apresentação (Frontend):** Interface web acessível pelo navegador que mostra a ocupação das vagas em tempo real.

2. Componentes e Funcionalidades

A. Módulo de Sensoriamento

- Sensores de obstáculo (infravermelho ou ultrassônicos) detectam a presença do veículo em cada vaga.

- Conectados aos pinos GPIO da BeagleBone Black.

- A leitura dos sensores é feita diretamente via arquivos do sistema Linux (/sys/class/gpio/gpioX/value).

B. Módulo de Controle (Servidor HTTP)

- Implementado em linguagem C, executado na BeagleBone Black.

- Atua como um servidor web simples, escutando requisições na porta 8080.

- Ao receber uma conexão, lê o estado dos sensores e responde com uma página HTML atualizada.

- Utiliza sockets TCP/IP e protocolo HTTP para comunicação.

C. Módulo de Interface Web (Frontend)

- Desenvolvido com HTML, CSS e JavaScript.

- Exibe o estado das vagas (livre ou ocupada) em tempo real.

- Pode ser acessado por qualquer dispositivo na mesma rede local da

BeagleBone Black.

- Suporte a atualizações periódicas para refletir mudanças nos sensores (ex: via JavaScript + setInterval()).

3. Comunicação

- A comunicação ocorre via HTTP sobre TCP/IP, com a BeagleBone Black atuando como servidor.
- A interface web é carregada diretamente da BeagleBone, sem necessidade de backend externo.
- Toda lógica de controle está embarcada no microcontrolador.

4. Tecnologias Utilizadas

Componente	Tecnologia	Justificativa
Microcontrolador	**BeagleBone Black**	Alta capacidade de controle GPIO e conectividade de rede integrada
Leitura de sensores	**GPIO via arquivos do Linux**	Interface simples e direta com sensores
Servidor web	**C com sockets TCP/IP**	Leve e eficiente para sistemas embarcados
Interface web	**HTML, CSS, JavaScript**	Visualização dinâmica e compatível com navegadores
Protocolo de rede	**HTTP sobre TCP/IP**	Comunicação padrão entre cliente e servidor

Testes Unitários

Como o código é em C, a biblioteca de testes unitários recomendada é o Unity (ou *cmocka*, *Check*). Abaixo, segue um teste unitário com Unity para a função *read_sensor*:

```c
#include "unity.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Mock da função de leitura de sensor
int read_sensor_mock(const char *path) {
    if (strcmp(path, "/sys/class/gpio/gpio12/value") == 0) return 1; // simula carro presente
    if (strcmp(path, "/sys/class/gpio/gpio13/value") == 0) return 0; // simula vaga vazia
    return -1;
}

void test_read_sensor_presenca_carro(void) {
    TEST_ASSERT_EQUAL_INT(1, read_sensor_mock("/sys/class/gpio/gpio12/value"));
}

void test_read_sensor_vaga_vazia(void) {
    TEST_ASSERT_EQUAL_INT(0, read_sensor_mock("/sys/class/gpio/gpio13/value"));
}

int main(void) {
    UNITY_BEGIN();
    RUN_TEST(test_read_sensor_presenca_carro);
    RUN_TEST(test_read_sensor_vaga_vazia);
    return UNITY_END();
}
```

Saída esperada ao rodar *./test_read_sensor*:

```
test_read_sensor.c:10:test_read_sensor_presenca_carro:PASS
test_read_sensor.c:14:test_read_sensor_vaga_vazia:PASS
-----------------------
2 Tests 0 Failures
OK

```

Testes Sistêmicos

Realizados testes manuais com base em 5 requisitos principais do sistema. Abaixo segue a planilha:
https://docs.google.com/spreadsheets/d/1uYGbSjb0stZT6uL7ZEmDTvmU09QDp-TE/edit?usp=sharing&ouid=114553037091077502137&rtpof=true&sd=true