Apresentação do Projeto Final para
Engenharia de Software

UNIVERSIDADE FEDERAL DO CEARÁ

Protótipo de um Estacionamento Inteligente

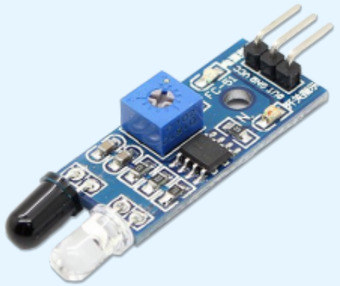
Professora: Carla Ilane Moreira Bezerra

Alunos:
Beatriz de Sousa Alves - 553556
Maria Eduarda Almeida Rodrigues - 552819
Nathalia de Oliveira Lima - 552319
Pablo Brandão Passos - 539730
Vitória das Graças Silva - 557364

Quixadá-CE
30 de Junho de 2025

Descrição do Projeto

- Automatização do controle de entrada e saída em estacionamentos;
- Visualização das vagas disponíveis em tempo real via web;
- Utilização de sensores para detectar veículos nas vagas e para abrir as cancelas;
- Sensor de obstáculo;
- Servo motor.



Usuários e Clientes

Cliente:
- Avante Tech Jr

Usuários:
- Clientes da Avante Tech Jr

Sobre a empresa:
- Empresa júnior de tecnologia da nossa faculdade.
- Ainda não possui estacionamento, mas planeja estruturar um futuramente como sede física e operacional.

Técnicas de Elicitação

- 1. Quantas vagas de estacionamento o sistema precisa gerenciar?
- 2. Quais funcionalidades considera essenciais no sistema desde o início?
- 3. Usuários precisarão fazer reservas de vagas antecipadamente ou o controle será apenas por ordem de chegada?
- 4. Quais são os principais problemas que o cliente quer resolver com o sistema?
- 5. Como será feito o suporte ao sistema após a implantação? O cliente espera manutenção contínua?

Requisitos

Total: 16 requisitos;

- 10 requisitos funcionais, sendo 7 requisitos já implementados;
- 6 requisitos não funcionais, sendo 4 requisitos já implementados;

Requisitos Funcionais

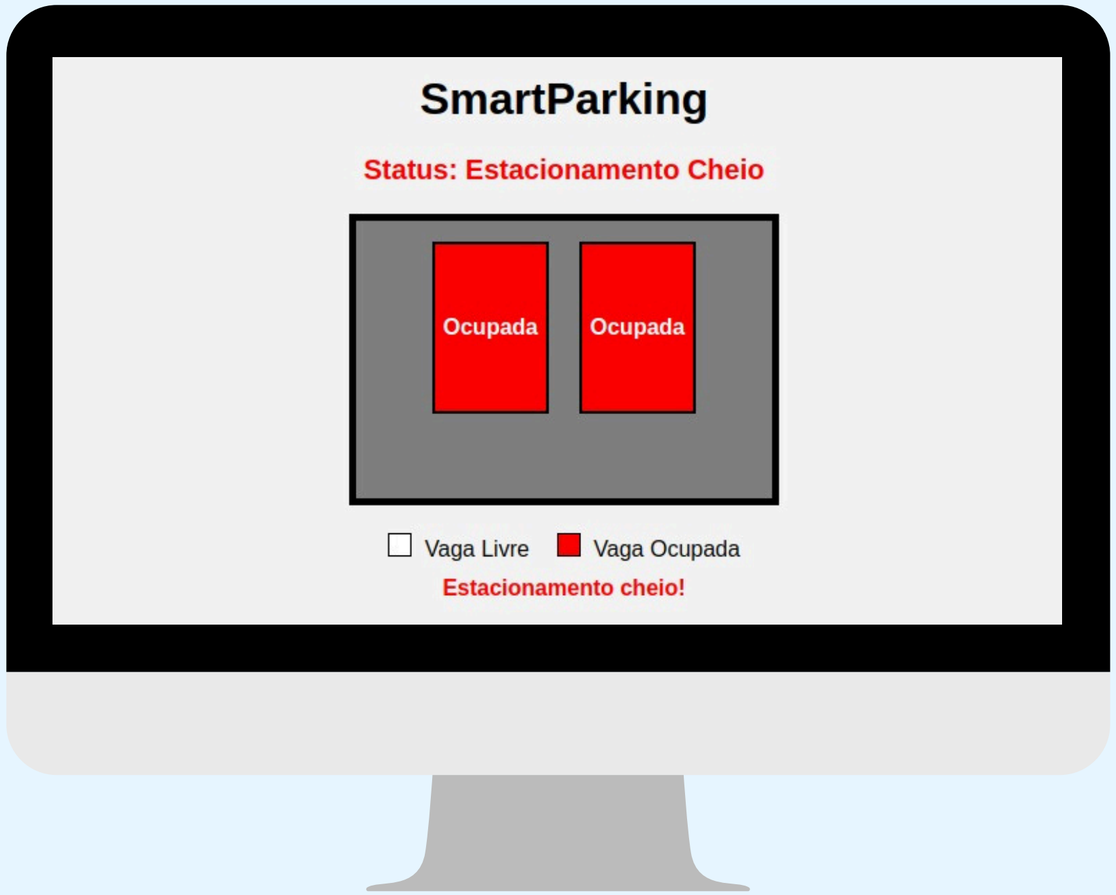
- **RF01** - O sistema deve detectar a presença de veículos nas vagas por meio de sensores.
- **RF02** - O sistema deve abrir automaticamente a cancela quando um veículo se aproxima da entrada e há vaga disponível.
- **RF03** - O sistema deve atualizar em tempo real a quantidade de vagas disponíveis.
- **RF04** - O sistema deve disponibilizar uma interface web acessível para visualização das vagas ocupadas e disponíveis.

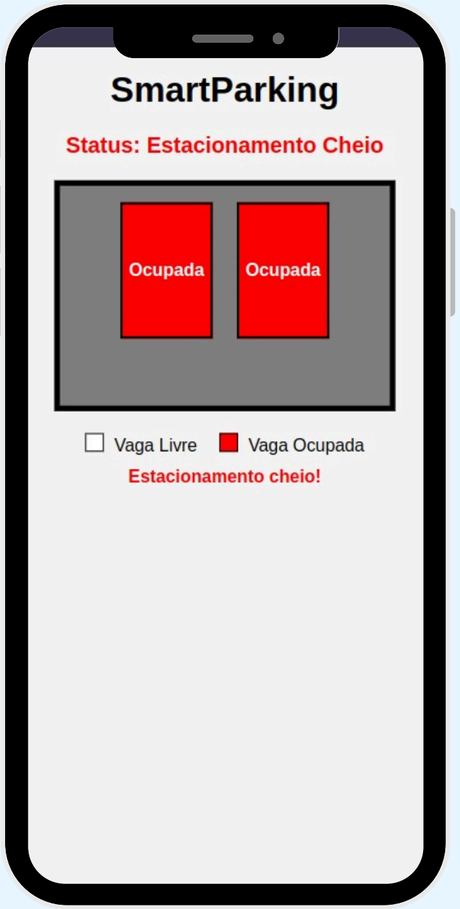
Requisitos Funcionais

- **RF05** - O sistema deve exibir em tempo real na interface web o status de cada vaga (livre/ocupada).
- **RF06** - O sistema deve possuir um painel local (display LCD ou LED) informando o número de vagas disponíveis na entrada do estacionamento.
- **RF07** - 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** - O código fonte do sistema deve ser modular para facilitar futuras atualizações.


SmartParking
Status: Estacionamento Cheio
Ocupada
Ocupada
Vaga Livre
Vaga Ocupada
Estacionamento cheio!


SmartParking
Status: Estacionamento Cheio
Ocupada
Ocupada
Vaga Livre
Vaga Ocupada
Estacionamento cheio!

Histórias de Usuário

US001 - Visualização de Vagas:
- Painel físico e web com atualização em tempo real.
- Exibe "LOTADO" quando sem vagas.

US002 - Abertura Automática:
- Cancela abre em 3s ao detectar veículo + vaga livre.

US004 - Interface Web:
- Consulta remota de vagas (atualização a cada 10s).

US005 - Saída Automática:
- Liberação por sensor sem ação do motorista.

Diagrama de Classes

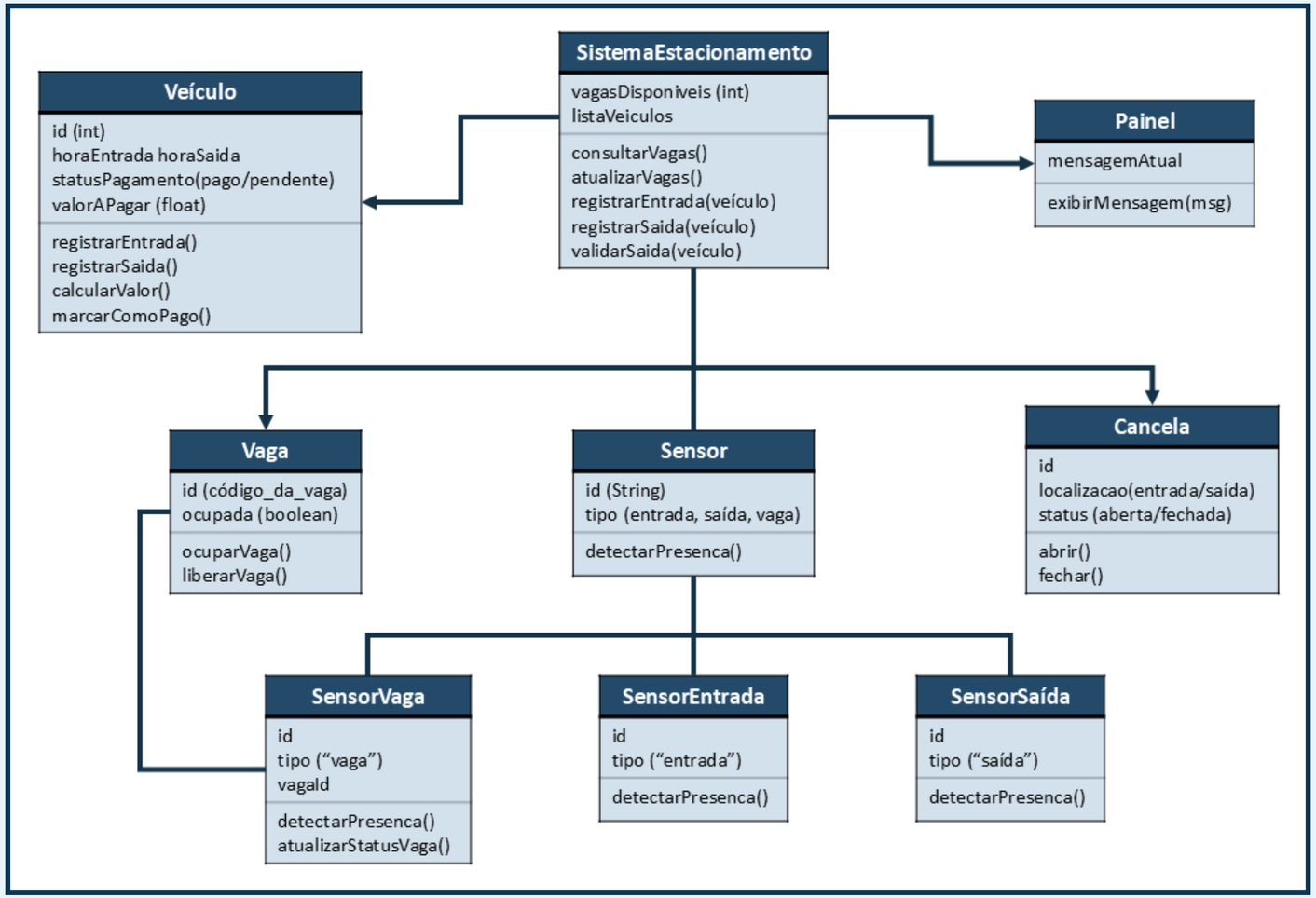


Diagrama de Classes

Responsabilidades das Principais Classes:

SistemaEstacionamento:
- Controla o funcionamento geral e a lógica do sistema.

Veículo:
- Armazena horários, status de pagamento e valor devido.

Vaga:
- Controla se está ocupada ou livre.

SensorVaga:
- Detecta se a vaga está ocupada.

SensorEntrada:
- Detecta a chegada de veículos.

SensorSaída:
- Detecta a saída e dispara verificação de pagamento.

Cancela e **Painel:**
- Controla entrada e saída física dos veículos.
- Exibe mensagens visuais ao usuário.

Diagrama de Atividades

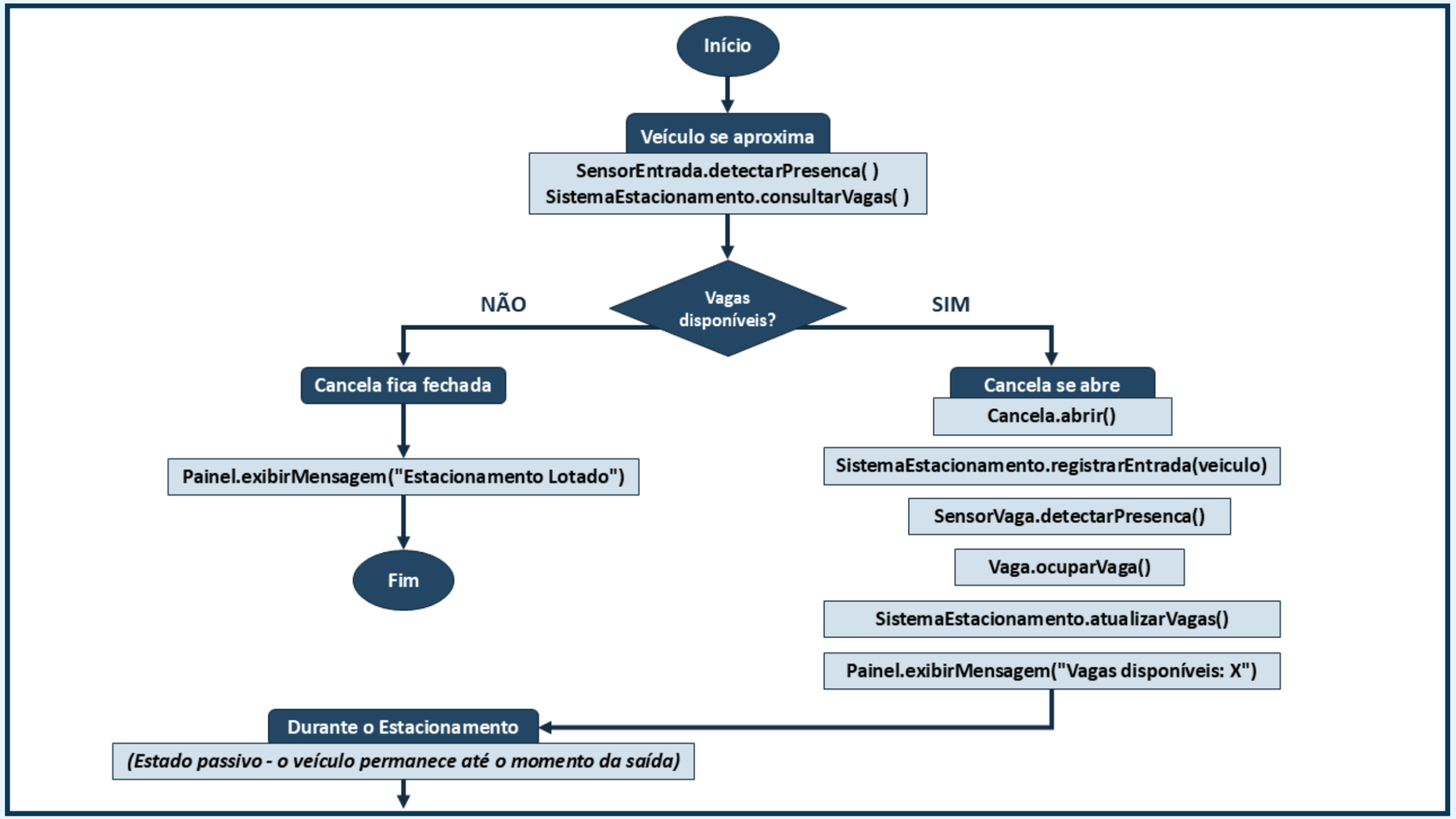
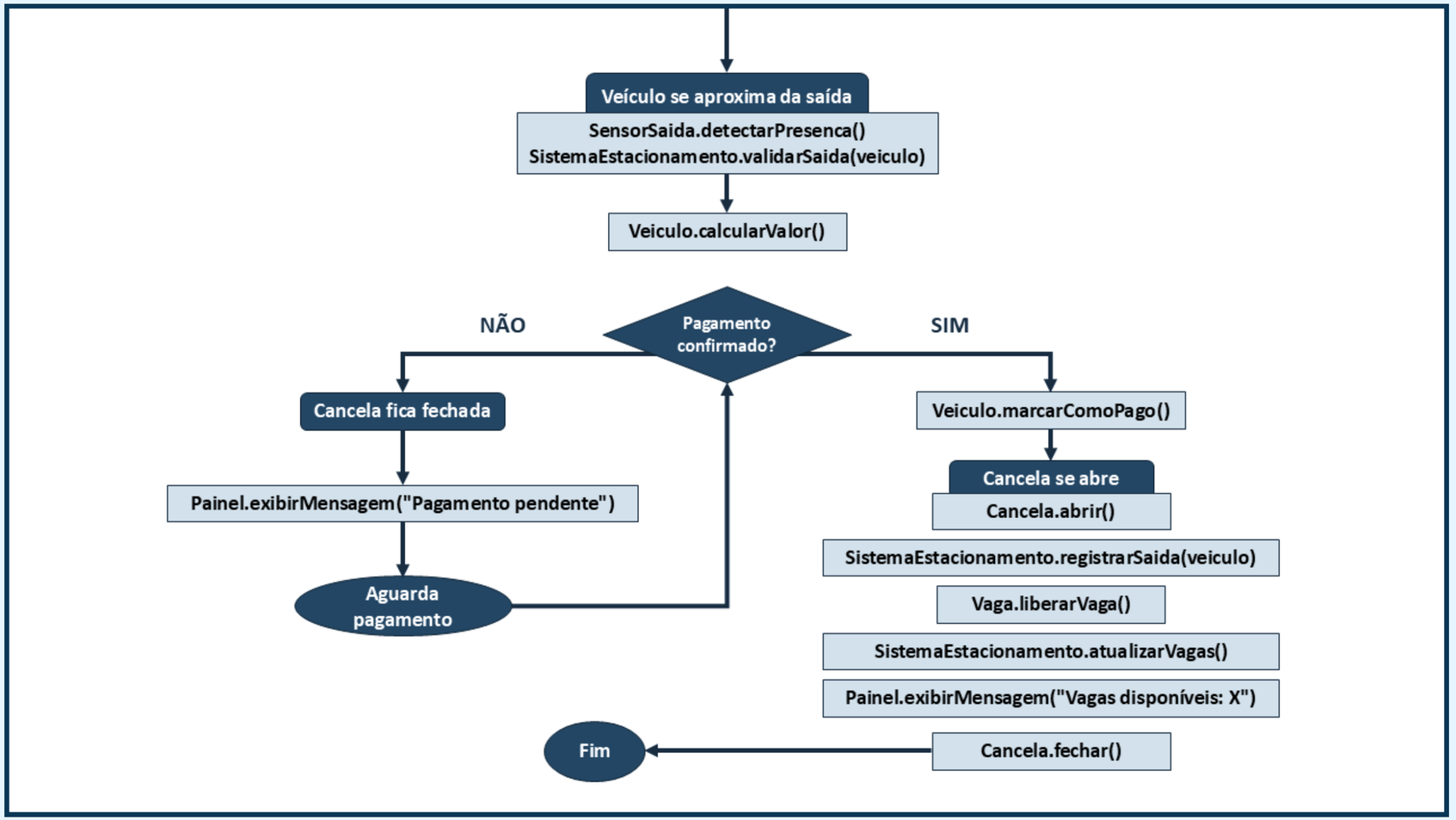
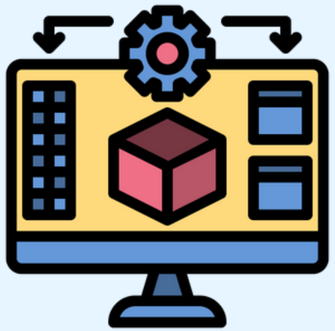


Diagrama de Atividades



Arquitetura



CAMADAS PRINCIPAIS:

1. Sensoriamento:

- Sensor de obstáculos: detecta objetos no ambiente;
- Servo motor: recebe comandos para realizar movimentos;
- Conexão via GPIO.

2. Controle (BeagleBone Black):

- Servidor HTTP em C;
- Porta 8080.

3. Apresentação final ao usuário:

- Interface HTML/CSS/JS;
- Responsiva para dispositivos móveis.

Apresentação parcial das funcionalidades implementadas

- **Leitura dos Sensores GPIO:**
  - Detecta se a vaga está ocupada (0) ou livre (1) através dos arquivos /sys/class/gpio/gpioXX/value.
  - Lê valores dos sensores usando código em C com fopen() e fgets().
- **Servidor HTTP em C:**
  - Cria servidor socket que executa na porta 8080.
  - Retorna a página HTML e o JSON com status das vagas.

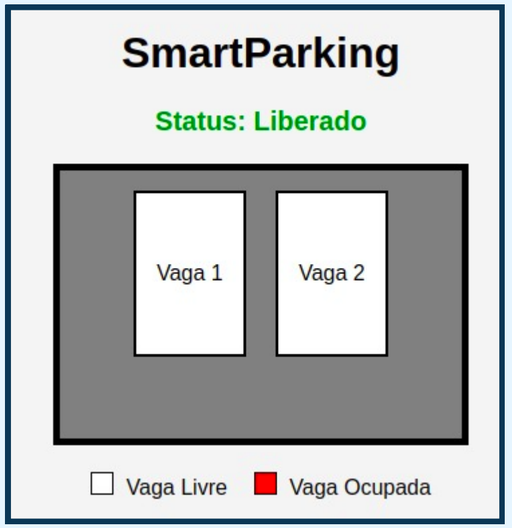
```c
#define PORT 8080
#define GPIO12 "/sys/class/gpio/gpio12/value" // Vaga 1
#define GPIO13 "/sys/class/gpio/gpio13/value" // Vaga 2
```

```c
char response[256];
snprintf(response, sizeof(response),
    "HTTP/1.1 200 OK\r\n"
    "Content-Type: application/json\r\n"
    "Access-Control-Allow-Origin: *\r\n"
    "\r\n"
    "{\"vaga1\": %d, \"vaga2\": %d}",
    vaga1_ocupada, vaga2_ocupada
);
```

Apresentação parcial das funcionalidades implementadas

- **Interface Web Dinâmica:**
  - Página HTML atualiza automaticamente o status das vagas usando JavaScript.
  - Utiliza fetch() com setInterval() para atualização a cada 2 segundos.
  - Mostra visualmente se a vaga está livre (branca) ou ocupada (vermelha).
- **Alerta Visual de Estacionamento Cheio:**
  - Exibe aviso "Estacionamento Cheio!" quando todas as vagas estão ocupadas.
  - Status do estacionamento muda de cor e texto dinamicamente.

```
setInterval(atualizarVagas, 2000);\n"
atualizarVagas();\n"
```



FIM!

OBRIGADO PELA ATENÇÃO!

UNIVERSIDADE FEDERAL DO CEARÁ