Briefing – Projeto Integrador

Grupo 3 – Plataforma de Monitoramento de Vagas em Estacionamentos Urbanos

Objetivo:

Criar uma **solução completa de controle e consulta de vagas** em estacionamentos públicos e privados, permitindo:

- **Motoristas** encontrarem vagas disponíveis em tempo real.
- **Donos de pátio** gerirem suas vagas, usuários, tarifas e relatórios.
- **Integração IoT** para atualização automática das vagas via sensores.

Contexto:

No centro urbano, motoristas gastam tempo e combustível dando voltas em busca de vagas, enquanto estacionamentos perdem clientes por falta de informação.

- **Problema do motorista:** dificuldade de localizar vaga livre.
- **Problema do dono (ex.: Marcos):** falta de transparência sobre ocupação e faturamento.

Solução proposta:

- Pequenos sensores de presença nas vagas enviam status (livre/ocupada) ao backend.
- O **app mobile** mostra disponibilidade em tempo real para motoristas.
- O **painel web** permite ao dono configurar preços, horários, políticas de uso e gerar relatórios.
- Tudo sincronizado via **API única**, garantindo consistência entre sistemas.

Público-Alvo

- **Motoristas urbanos:**
  Pessoas que utilizam carro diariamente em centros urbanos e precisam encontrar vagas com rapidez e segurança.
- **Donos de estacionamentos privados:**
  Pequenos e médios empreendedores (como Marcos) que buscam mais controle sobre ocupação, faturamento e gestão de clientes.
- **Prefeituras e gestores de estacionamento público/rotativo:**
  Organizações que desejam modernizar o monitoramento, reduzir congestionamentos e aumentar a eficiência do uso do espaço urbano.

- **Usuários corporativos (futuro):**
  Empresas que desejam oferecer vagas monitoradas para funcionários, clientes ou parceiros.

Estilo e Design

O que queremos passar com o projeto em design:

Profissionalismo, dashboard institucional, aplicativo humanizado e natural.

Paleta de Cores

Cor Principal #4A4A4A (cinza neutro médio, versátil para fundo e textos principais);

Cor Secundária #FFFFFF (limpo, base para contraste);

Cor Destaque #FFD600 (amarelo vibrante, chama atenção para botões, links, ícones);

Variações De Apoio

#e0e0e0 (para fundos suaves, bordas, separadores);

#1c1c1c (para textos fortes e títulos);

#fff59d (hover, estados secundários, detalhes sutis);

#f9a825 (variação de destaque para contraste);

DarkMode (talvez)

Planejamento e Prazos

- **Data de entrega oficial:** 04/12 (92 dias a partir de hoje).
- **Meta interna do grupo:** finalizar em **80 dias**.
- **Objetivo da meta antecipada:** garantir **~12 dias de folga** para:
    - Resolver imprevistos técnicos.
    - Realizar testes completos.
    - Revisar documentação e apresentação.

Plataformas e Integrações:

- **Web (Admin):**
  Painel administrativo para donos de estacionamentos (gestão de vagas, usuários, relatórios).
- **Mobile (Motoristas):**
  Aplicativo para busca, consulta e reserva de vagas em tempo real.
- **IoT:**
  Sensores de presença enviando status (ocupada/livre) via MQTT → backend → atualização automática.

Requisitos Funcionais (RF)

Cadastro e autenticação

- RF01 – Cadastro de usuários (motoristas e donos de pátio).
- RF02 – Login com autenticação JWT + refresh tokens.
- RF03 – Recuperação de senha (talvez).
- RF04 – Gestão de perfis (dados pessoais, contato, pagamento).

Gestão de estacionamentos

- RF05 – Cadastro de estacionamento (nome, endereço, mapa, horários, tarifas).
- RF06 – Cadastro de vagas (quantidade, identificação por número/área).
- RF07 – Políticas de preço (hora, fração, diária, descontos).
- RF08 – Permissões (quem pode ver, reservar, administrar).

Disponibilidade de vagas

- RF09 – Atualização automática via sensores IoT.
- RF10 – Simulação de sensores via broker MQTT.

- RF11 – Consulta em tempo real no app.
- RF12 – Filtros/Tags (distância, preço, horário, estacionamento).

Reserva e permanência

- RF13 – Reserva de vagas (opcional).
- RF14 – Cancelamento de reservas.
- RF15 – Notificações push (vaga liberada, tempo expirar).

Pagamentos (futuro)

- RF16 – Pagamento via app (Via simulação).
- RF17 – Emissão de recibo digital (talvez).

Painel Administrativo (Web)

- RF18 – Ocupação em tempo real.
- RF19 – Gestão de usuários vinculados.
- RF20 – Relatórios (ocupação, faturamento, tempo médio).
- RF21 – Exportação de relatórios (PDF/CSV).
- RF22 – Dashboard com gráficos (Recharts).

Comunicação e integração

- RF23 – API única para Web, Mobile e IoT.
- RF24 – Integração MQTT para sensores.
- RF25 – WebSockets para dados em tempo real.

Requisitos Não Funcionais (RNF)

Desempenho

- RNF01 – Atualização ≤ 3s após mudança no sensor.
- RNF02 – Backend suporta uma grande quantidade de requisições simultâneas.

Segurança

- RNF03 – Senhas criptografadas (bcrypt).
- RNF04 – Tokens JWT com refresh e expiração curta.
- RNF05 – Controle de acesso baseado em papéis/tipos de login (RBAC).

Disponibilidade & Confiabilidade

- RNF06 – Uptime ≥ 99%.
- RNF07 – Logs de erros e auditoria.
- RNF08 – Backup automático diário.

Usabilidade & UX

- RNF09 – App responsivo (mobile-first).
- RNF10 – Acessibilidade (WAI-ARIA).
- RNF11 – UX padronizada (shadcn + Tailwind).

Compatibilidade

- RNF12 – Android ≥ 8.
- RNF13 – Navegadores modernos (Chrome, Edge, Firefox, Opera).

Escalabilidade

- RNF14 – Suporte a múltiplos estacionamentos e milhares de vagas.

IoT & Simulação

- RNF15 – Suporte a sensores reais via MQTT.
- RNF16 – Modo simulação (mock com atualização periódica).

Portabilidade

- RNF1 – Dockerização backend e frontend.

IoT & Simulação:

- Simulador em Node.js gera mensagens de "vaga livre/ocupada" a cada 2s → envia pro broker MQTT.
- **MQTTX** para testes manuais.
- Backend com listener MQTT para atualizar banco.
- Painel de simulação (botão *Ocupar/Desocupar*) dispara mensagens para o broker.

Tecnologias:

Frontend Web (Admin)

- Next.js 14 (SSR/ISR).
- TailwindCSS + Shadcn (design system).
- Axios + React Query (cache e sincronização real-time).

Mobile

- React Native + Expo.
- Integração com WebSockets e MQTT.

Backend

- Node.js (NestJS recomendado / Express.js opcional).
- PostgreSQL (preferido) ou MySQL.
- Prisma ORM.
- WebSockets para tempo real.

IoT

- MQTTX para testes.
- Mosquitto Broker.

Gráficos e Relatórios

- Recharts.

Infraestrutura

- Docker para deploy padronizado.

Diferenciais

- Atualização em tempo real (sensores + WebSockets).
- Integração IoT completa (hardware real + simulação).
- Maquete (talvez).
- Experiência unificada: painel web para gestores e app mobile para motoristas/clientes.
- Design system consistente (shadcn + Tailwind).
- Escalabilidade prevista para múltiplos estacionamentos.