🏆🚀 Desafio da Semana: Angular! 🚀🏆

E aí, starters! 🚀 Estão prontos para mais um desafio? 😏 Desta vez, vamos trazer o front-end à vida com Angular! 🎨💻
Chegou a hora de conectar nossos microsserviços e criar uma experiência incrível para os usuarios s e seus pets! 🐶🐱
Vamos construir telas interativas, consumir APIs e deixar tudo fluindo como um bom código bem escrito!
Afie o TypeScript, prepare os componentes e bora mostrar do que somos capazes! 🔥

Front end em Angular para sistema de cadastro e agendamento de cuidados de pets

O desafio da semana será desenvolver um front-end em Angular com módulos de Cadastro de Pets, que registra e enriquece os dados via TheCatAPI e TheDogAPI; Agendamentos, para criar e visualizar serviços; e módulos opcionais , um para notificações, que exibe os e-mails enviados aos tutores e outro para login dos usuários. O sistema consumirá os microsserviços desenvolvidos no desafio anterior.🐾

IMPLEMENTAÇÃO BÁSICA DO SISTEMA:

📌O front-end do petshop será dividido em 2 módulos principais e 2 opcionais:

1️⃣ **Módulo de Cadastro de Pets** – Permite o registro de pets, consumindo o microsserviço de cadastro para salvar os dados e buscar informações adicionais via APIs externas (TheCatAPI e TheDogAPI).
2️⃣ **Módulo de Agendamentos** – Exibe os agendamentos automáticos (como primeira vacina e checkup inicial) e permite a criação manual de serviços, comunicando-se com o microsserviço de agendamentos.
3️⃣ **Módulo de Notificações (Exceeds)** – Lista os e-mails enviados aos tutores sobre os agendamentos, consumindo o microsserviço de notificações.
4️⃣ **Módulo de Login (Exceeds)** – Permite a autenticação do usuário, garantindo acesso seguro às funcionalidades do sistema.
A interface será desenvolvida em Angular, utilizando API REST para comunicação com os microsserviços.

📌Tecnologias e Bibliotecas:

🅰️ Angular (versão mais recente) - Framework principal do front-end.
📦 Angular Material ou PrimeNG - Para componentes de UI (tabelas, formulários, diálogos, etc.).
🌍 Angular Router - Para navegação entre telas.
🎨 Figma/Miro (opcional) - Para prototipação e desenvolvimento das ideias das telas antes da implementação.
🐳 Docker (opcional) - Para facilitar a execução do front-end em ambiente local.

🐾 Módulo de Cadastro de Pets

Possibilita o cadastro de pets, integrando-se ao microsserviço de cadastro de pets para armazenar os dados e obter informações extras por meio das APIs externas TheCatAPI e TheDogAPI.

📌Cadastro de Pets:

- Formulário para inserir informações do pet (ex: nome, espécie, raça, idade, peso, descrição, tutor e e-mail do tutor).
- Busca automática de imagem e informações adicionais ao selecionar a raça (consulta às APIs externas via microsserviço, ou via API REST diretamente do front).
- Feedback visual indicando o sucesso ou falha no cadastro.

Fluxo do Cadastro de Pets

1️⃣ Usuário acessa a tela de cadastro e preenche o formulário com as informações básicas do pet.
2️⃣ Usuário confirma o cadastro e envia os dados.
3️⃣ O sistema chama o microsserviço de cadastro de pets para salvar as informações no banco de dados.
4️⃣ O front-end exibe um feedback visual (sucesso ou erro).
5️⃣ Em caso de sucesso, o usuário pode:
- Cadastrar outro pet
- Ir para a lista de pets
- Ver detalhes do pet contendo a imagem das APIs externas (TheCatAPI ou TheDogAPI).

📌Lista de Pets:

- Exibe os pets cadastrados pelo usuário.
- Filtros avançados por espécie, raça, idade, etc.
- Opção para visualizar detalhes do pet.

Fluxo da Lista de Pets

1️⃣ O usuário acessa a tela de pets cadastrados, que carrega os dados chamando o microsserviço de cadastro de pets.
2️⃣ O sistema exibe todos os pets cadastrados com imagens, nomes e informações básicas.
3️⃣ O usuário pode utilizar filtros avançados para buscar pets por espécie, raça, idade, peso, cor, etc.
4️⃣ Ao clicar em um pet, o usuário é redirecionado para a tela de detalhes do pet.

📌Detalhes do Pet:

- Exibe informações completas do pet e a imagem obtida via API.
- Opção de editar ou excluir o pet.

Fluxo da Tela de Detalhes do Pet

1️⃣ O usuário acessa a tela com as informações completas e a imagem do pet.
2️⃣ Pode editar os dados, salvando as alterações via microsserviço.
3️⃣ Pode excluir o pet, com uma confirmação antes da remoção.
4️⃣ Após excluir, é redirecionado para a lista de pets atualizada.

📅 Módulo de Agendamento de cuidados

Apresenta os agendamentos automáticos, como a primeira vacina e checkup inicial, além de possibilitar a criação manual de serviços, interagindo com o microsserviço de agendamentos.

📌 Tela de Agendamento Cuidados Manuais

- A tela de agendamento manual permitirá que o tutor selecione um pet e agende um serviço específico, como vacinação, banho e tosa, consulta veterinária ou administração de medicamentos.

Fluxo da Tela de Agendamento Manual

1️⃣ O usuário acessa a tela e seleciona um pet da lista.
2️⃣ Escolhe o tipo de serviço (vacinação, banho e tosa, consulta veterinária ou administração de medicamentos).
3️⃣ Seleciona a data e horário disponíveis.
4️⃣ Confirma o agendamento clicando em "Agendar".
5️⃣ O sistema envia os dados ao microsserviço de agendamentos e exibe um feedback de sucesso ou erro.
6️⃣ O novo agendamento é adicionado à lista de agendamentos do pet.

📌 Lista de Agendamentos:

- Uma tela dedicada onde o tutor pode visualizar todos os agendamentos automáticos e manuais.
- Pode incluir filtros por pet, status, data e tipo de serviço para facilitar a busca.
- Diferenciar visualmente os agendamentos automáticos dos manuais, como usando ícones, cores diferentes ou tags (exemplo: "Automático" e "Criado pelo Usuário").

Fluxo da Lista de Agendamentos

1️⃣ O usuário acessa a tela de agendamentos, que carrega todos os serviços agendados.
2️⃣ Pode aplicar filtros por pet, status, data e tipo de serviço para facilitar a busca.
3️⃣ O sistema diferencia visualmente agendamentos automáticos (ex: cor diferente, tag "Automático") dos manuais (ex: tag "Criado pelo Usuário").
4️⃣ O usuário pode confirmar, reagendar ou cancelar um agendamento conforme a necessidade.

🎨 Melhorias Extras :

- Calendário que mostra os agendamentos automáticos e manuais
- Validação de campos obrigatórios antes de enviar.
- Animações sutis ao selecionar um pet ou serviço.
- Exibir horários disponíveis dinamicamente com base na data escolhida.
- Botão de "Voltar" para melhorar a navegação.
- Mensagem de alerta se houver um agendamento conflitante.

📨 Módulo de Notificações (Exceeds)

Lista os e-mails enviados aos tutores sobre os agendamentos, consumindo o microsserviço de notificações.

Fluxo da Tela de Notificações

1️⃣ O usuário acessa a tela de notificações.
2️⃣ O sistema faz uma requisição GET /notificacoes para listar os e-mails enviados aos tutores.
3️⃣ As notificações são exibidas com informações como data, assunto e status de envio.
4️⃣ O usuário pode filtrar por pet, tipo de notificação ou status (enviado, pendente, falha).
5️⃣ Opcionalmente, pode marcar notificações como lidas ou solicitar o reenvio em caso de falha.

🎨 **Melhorias Extras :**

- Diferenciação visual por status – Exibir cores ou ícones para indicar notificações lidas, pendentes ou com erro.
- Filtro avançado – Permitir busca por data, tipo de notificação e status.
- Opção de reenvio – Para notificações que falharam no envio, incluir um botão de "Reenviar".
- Modo detalhado – Ao clicar em uma notificação, exibir o conteúdo do e-mail enviado.
- Marcar como lido em lote – Opção para selecionar múltiplas notificações e marcar todas como lidas.

🔐Módulo de Login (Exceeds)

Controla o acesso ao sistema com autenticação segura.

Fluxo do Login no Front-End

1️⃣ O usuário acessa a tela de login e insere suas credenciais.
2️⃣ O sistema faz uma requisição POST /login para autenticar o usuário.
3️⃣ Se válido, o back-end retorna um JWT que é armazenado no localStorage ou sessionStorage.
4️⃣ O usuário é redirecionado para o painel principal do sistema.
5️⃣ Todas as requisições protegidas enviam o JWT no cabeçalho Authorization.
6️⃣ O sistema usa AuthGuard para restringir acesso a páginas protegidas.
7️⃣ O usuário pode fazer logout, removendo o token e redirecionando para a tela de login.

🎨 **Melhorias Extras :**

- Login com redes sociais – Permitir login via Google, Facebook ou Apple ID.
- Feedback em tempo real – Exibir mensagens como "E-mail não cadastrado" ou "Senha incorreta" .
- Validação de caracteres.
- Exibir senha temporariamente – Ícone para visualizar a senha digitada antes de enviar.
- Login persistente – Opção "Lembrar-me" para manter a sessão ativa.
- Animação suave – Transições suaves para melhorar a experiência ao alternar entre login, cadastro e recuperação de senha.

📌 Dicas de Estruturação:

✅ Modularização – Cada funcionalidade deve ter seu próprio módulo (auth, pets, agendamentos, notificações) para melhor organização e reutilização.
✅ Separação de Camadas – Serviços (.service.ts) gerenciam a lógica e comunicação com APIs, enquanto componentes (.component.ts) lidam com a interface.
✅ Routing Module – Cada módulo deve ter suas próprias rotas , mantendo o app-routing.module.ts mais limpo.
✅ Pasta Core e Shared – Core armazena funcionalidades globais (Auth, Interceptors, Guards) e Shared contém componentes reutilizáveis (Botões, Modais, Pipes).
✅ Organização Limpa – Código estruturado para facilitar manutenção e escalabilidade.

🎨 Dicas de Estilização:

✅ Separação de HTML e CSS – Evitar estilos inline e manter CSS organizado em arquivos separados.
✅ Navegação (Menu, Header, Footer)

- Menu fixo ou expansível para facilitar o acesso.
- Header com nome do usuário e logout.

✅ Botões e Interatividade

- Cores contrastantes para ações principais (verde para confirmar, vermelho para excluir).
- Efeitos sutis de hover e clique para melhor feedback.

✅ Harmonia Visual

- Paleta de 3-4 cores para consistência no design.
- Hierarquia tipográfica bem definida (títulos destacados, subtítulos intermediários, texto normal).
- Flexbox ou Grid CSS para responsividade.

✅ Feedback Visual

- Mensagens claras de sucesso (verde), erro (vermelho) e avisos (amarelo).
- Loaders ou Skeleton Screens para melhorar a experiência de carregamento.

✅ Modo Escuro (Opcional)

- Implementação via CSS Variables, permitindo troca dinâmica de temas.
- Baixo contraste para evitar fadiga visual.

CRITÉRIOS DE AVALIAÇÃO:

✅1. Boa organização do código e modularização.

✅2. Separação de HTML e CSS.

✅ 3. Integração bem-sucedida com as API´s externas.

✅4. Comunicação bem-sucedida com os Microsserviços .

✅5. Qualidade dos tratamentos de erro e mensagens amigáveis no retorno.

✅6. Navegação intuitiva e experiência do usuário (UX)

✅7. Qualidade da estilização e harmonia visual (UI Design)

✅8. Implementação de filtros e busca eficientes

✅9. Readme

✅10. Tratamento de erros na comunicação com APIs

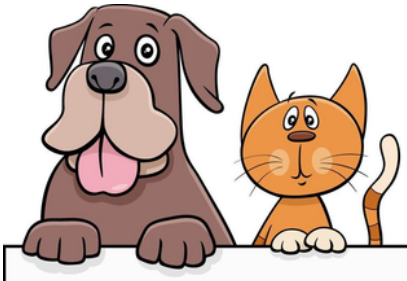
✅11. Telas em funcionando adequadamente.

✅12. Containerização com Docker (EXCEEDS)

✅13. Login (EXCEEDS)

OBS: Utilize IA de forma responsável no desafio de microsserviço, compreenda o problema, valide as respostas e entregue códigos testados e em pleno funcionamento. A IA é uma aliada, mas a qualidade e confiabilidade dependem de você!

⭐Sucesso no Desafio!⭐



Que a força, o foco e o código bem escrito estejam com vocês! 💪💻 Bora encarar esse desafio e mostrar do que somos capazes! 🚀🔥