# 🏆🚀 Desafio da Semana: Microserviços! 🚀🏆

E aí, starters! Prontos para mais um desafio? 😏 Desta vez, vamos mergulhar no mundo dos microsserviços! 💡⚙️
Chamem a criatividade, afiemos o código e bora mostrar do que somos capazes!
Preparados para uma aventura cheia de fofura e código? 🐶🐱

## Sistema de Cadastro e Agenda de Cuidados para Pets

O desafio desta semana é desenvolver um sistema com três microsserviços que se comunicam via RabbitMQ e podem ser containerizados com Docker. O primeiro microsserviço será responsável pelo cadastro de pets, armazenando informações detalhadas sobre cada animal, enquanto o segundo gerenciará a agenda de cuidados, organizando vacinações, banhos e consultas veterinárias, o terceiro enviará notificações por email para os tutores. Além disso, a integração entre os serviços garantirá um fluxo eficiente e dinâmico para o gerenciamento dos pets. 🐾

**IMPLEMENTAÇÃO BÁSICA DO SISTEMA:**

1. **Microsserviço de Cadastro de Pets:** 📝

Gerencia o cadastro de pets com informações detalhadas , integra dados externos por meio das APIs TheCatAPI e TheDogAPI, enriquecendo o perfil dos pets com imagens e informações complementares. Além disso, o serviço oferece funcionalidades de busca avançada (por filtros como espécie e raça).

Utiliza comunicação assíncrona via RabbitMQ, ao salvar um novo pet e pode também responder a solicitações de dados para alimentar o serviço de agendamento de cuidados escolhendo por via RabbitMQ ou API REST.

**2. Microsserviço de Agenda de Cuidados:** 📅

Cria agendamentos automáticos quando o microsserviço de cadastro de pets cadastra um novo pet, aplicando regras de negócio como primeira vacina, primeiro banho grátis e checkup inicial, e também permite agendamentos manuais para cuidados específicos, como vacinações, banho e tosa, consultas veterinárias e administração de medicamentos. Gerencia o armazenamento em um banco de dados e buscando dados no microsserviço de cadastros de pets escolhendo por via API REST ou RabbitMQ .

**3. Microsserviços de Notificações (EXCEEDS):**

Escuta eventos no RabbitMQ quando novos agendamentos são criados, enviando e-mails aos tutores dos pets e garantindo a entrega assíncrona das mensagens, de modo a mitigar os impactos de eventuais falhas temporárias no sistema.

# 📌 Microsserviço de Agendamento de Cuidados

- Cria **agendamentos automáticos** quando um novo pet é cadastrado no microsservico de cadastro de pets, aplicando regras de negócio como primeira vacina, primeiro banho gratis, chekup inicial.
- Permite **agendamentos manuais**, onde o usuário pode escolher um cuidado específico, como como vacinações, banho e tosa, consultas veterinárias e administração de medicamentos.
- Gerencia o armazenamento dos agendamentos em um banco de dados.
- Publica eventos no RabbitMQ para que o Microserviço de Notificações envie e-mails aos tutores.

O microsserviço terá obrigatoriamente dois fluxos :

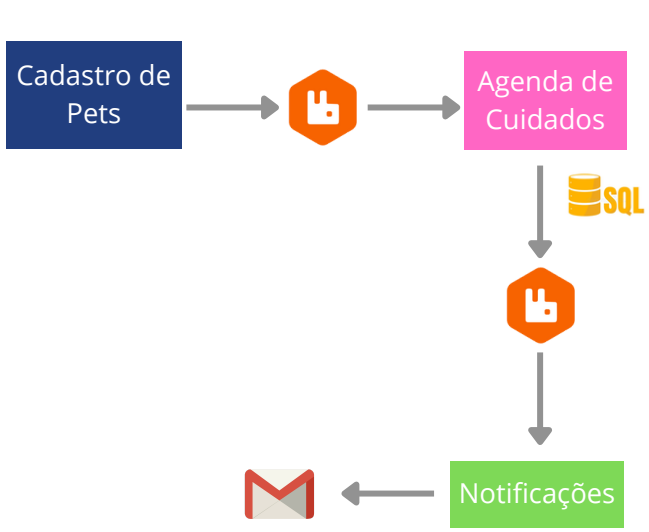
1️⃣ **Criação de Agendamentos Automáticos** → Processa eventos do Cadastro de Pets **via RabbitMQ (assíncrono).**
2️⃣ **Criação de Agendamentos Manuais** →obtém dados do pet do Cadastro de Pets **via REST (síncrono).**

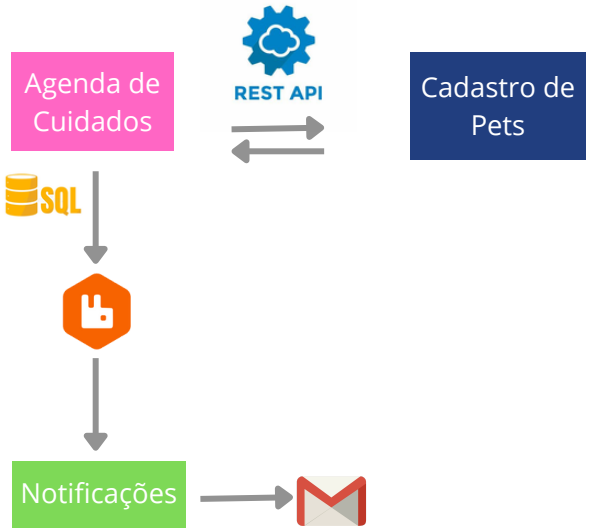
**📌 Fluxo do Agendamento Automático via RabbitMQ:**

1️⃣O Microsserviço de Cadastro de Pets ao salvar os dados do um novo pet no banco de dados publica um evento pet_created no RabbitMQ.
2️⃣O Agenda de Cuidados escuta o evento e aplica regras de negócio para definir o agendamento automático, ex:
Se idade < 6 meses → Criar um agendamento de vacinação inicial.
Se idade >= 6 meses → Criar um agendamento de check-up inicial.
3️⃣O Agenda de Cuidados salva o agendamento no banco de dados.
4️⃣O Agenda de Cuidados publica um evento appointment_created no RabbitMQ.
5️⃣O Microserviço de Notificações escuta esse evento e envia um e-mail ao tutor do pet.

**📌Fluxo do Agendamento Manual via REST:**

1️⃣ O Agenda de Cuidados faz uma chamada REST síncrona ao Cadastro de Pets para obter as informações do pet antes de confirmar o agendamento.
2️⃣O Cadastro de Pets responde imediatamente com os dados do pet.
3️⃣O Agenda de Cuidados cria o agendamento e salva no banco de dados.
4️⃣O Agenda de Cuidados publica um evento appointment_created no RabbitMQ.
5️⃣ O Microserviço de Notificações escuta esse evento e envia um e-mail ao tutor do pet.



# 📌 Microsserviço de Agendamento de Cuidados

- Cria **agendamentos automáticos** quando um novo pet é cadastrado no microsservico de cadastro de pets, aplicando regras de negócio como primeira vacina, primeiro banho gratis, chekup inicial.
- Permite **agendamentos manuais**, onde o usuário pode escolher um cuidado específico, como como vacinações, banho e tosa, consultas veterinárias e administração de medicamentos.
- Gerencia o armazenamento dos agendamentos em um banco de dados.
- Publica eventos no RabbitMQ para que o Microserviço de Notificações envie e-mails aos tutores.

O microsserviço terá obrigatoriamente dois fluxos :

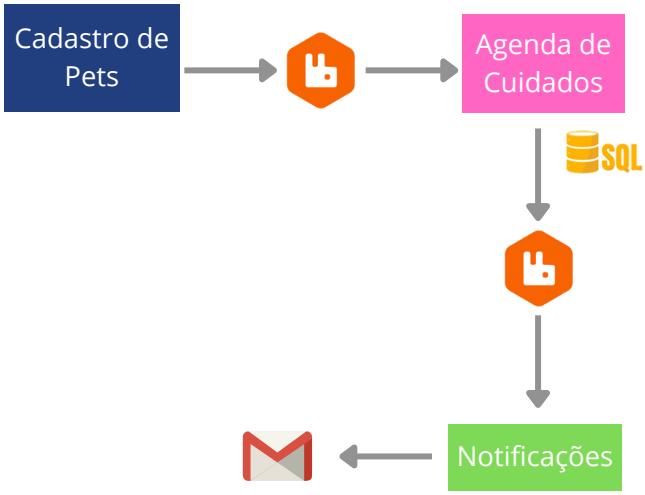
1️⃣ **Criação de Agendamentos Automáticos** → Processa eventos do Cadastro de Pets **via RabbitMQ (assíncrono).**
2️⃣ **Criação de Agendamentos Manuais** → obtém dados do pet do Cadastro de Pets **via RabbitMQ (assíncrono).**

**📌 Fluxo do Agendamento Automático via RabbitMQ:**

1️⃣O Microsserviço de Cadastro de Pets ao salvar os dados do um novo pet no banco de dados publica um evento pet_created no RabbitMQ.
2️⃣O Agenda de Cuidados escuta o evento e aplica regras de negócio para definir o agendamento automático, ex:
Se idade < 6 meses → Criar um agendamento de vacinação inicial.
Se idade >= 6 meses → Criar um agendamento de check-up inicial.
3️⃣O Agenda de Cuidados salva o agendamento no banco de dados.
4️⃣O Agenda de Cuidados publica um evento appointment_created no RabbitMQ.
5️⃣O Microserviço de Notificações escuta esse evento e envia um e-mail ao tutor do pet.

**📌 Fluxo do Agendamento Manual via RabbitMQ**

1️⃣ O Agenda de Cuidados verifica se já tem informações do pet no seu banco de dados.
2️⃣ Caso não tenha, publica um evento pet_info_request no RabbitMQ para buscar dados do pet no microsservico de cadastro de pets.
3️⃣O Cadastro de Pets escuta esse evento e publica um pet_info_response com os dados do pet.
4️⃣ O Agenda de Cuidados recebe a resposta, salva o agendamento no banco e publica um evento appointment_created para o Microserviço de Notificações.
5️⃣O Microserviço de Notificações escuta esse evento e envia um e-mail ao tutor confirmando o agendamento.




## Microsserviço de Cadastro de Pets: 📝

- Permitir o registro de pets com informações detalhadas : nome, espécie (cachorro, gato), tutor, emailTutor, raça, idade, peso, cor, descrição.
- Integrar com as APIs externas TheCatAPI e TheDogAPI para enriquecer os dados dos nossos amigos peludos (como consulta de imagens, dados adicionais ).
- Funcionalidade de busca para encontrar pets cadastrados: Filtros por espécie, raça, etc.

### Comunicação com RabbitMQ:

- Ao salvar os dados do um novo pet no banco de dados publica um evento pet_created no RabbitMQ para o microsservico de agendamento de cuidado consumir.

**📌Fluxo para cadastro de pets:**

1️⃣ O usuário cadastra um novo pet via API REST (POST /api/pets).
2️⃣ O Cadastro de Pets salva os dados do pet no banco de dados.
3️⃣ O Cadastro de Pets integra com APIs externas (TheCatAPI/TheDogAPI) para obter uma imagem do pet.
4️⃣ O Cadastro de Pets cria um evento pet_created com os dados do pet.
5️⃣ O evento é publicado no RabbitMQ para ser consumido pelo Microsserviço de Agenda de Cuidados.
6️⃣ O Agenda de Cuidados escuta o evento, aplica regras de negócio e cria agendamentos automáticos para o pet.

**Para a escolha do Fluxo Assíncrono de Agendamento de Cuidados:**

- Escuta do RabbitMQ para responder a solicitações do Agenda de Cuidados sobre informações do pet (pet_info_request)

**📌Fluxo para Responder a Solicitações do Agenda de Cuidados:**

1️⃣ O Microserviço de Agenda de Cuidados precisa buscar informações de um pet para criar um agendamento manual.
2️⃣ O Agenda de Cuidados publica um evento pet_info_request no RabbitMQ, solicitando os dados do pet ao Cadastro de Pets.
3️⃣ O Cadastro de Pets escuta a fila pet.requests, busca os dados no banco e cria uma resposta.
4️⃣ O Cadastro de Pets publica um evento pet_info_response no RabbitMQ, contendo as informações do pet.
5️⃣ O Agenda de Cuidados escuta a fila pet.responses, recebe os dados e cria o agendamento manual.

## Microsserviço de Notificações(Exceeds) 📧

- Escuta eventos no RabbitMQ quando um novo agendamento é criado.
- Envia notificações por e-mail para os tutores dos pets.
- Garante a entrega assíncrona das notificações, evitando que falhas temporárias afetem o sistema

### 📌 Fluxo do Microsservico de Notificações

1️⃣ O Microserviço de Agenda de Cuidados publica um evento appointment_created no RabbitMQ após criar um agendamento.
2️⃣ O Microserviço de Notificações escuta esse evento e processa a notificação.
3️⃣ Ele recupera os detalhes do agendamento e formata a mensagem de e-mail.
4️⃣ O serviço envia a notificação ao tutor do pet via e-mail.
5️⃣ Caso o envio falhe, a mensagem pode ser reencaminhada automaticamente.

**IMPLEMENTAÇÕES ADICIONAIS (EXCEEDS)**

**Dockerfiles e Docker Compose:**

- Criar Dockerfiles para cada microsserviço, incluindo as dependências necessárias.
- Garantir que os microsserviços sejam construídos e executados corretamente em containers.
- Criar um arquivo docker-compose.yml para orquestrar a subida dos containers.
- Configurar as redes e volumes necessários para garantir a comunicação e persistência dos dado**s.**

**Configuração do Eureka Server / Consul Server:**

- Criar um serviço para registrar e descobrir os microsserviços.
- Configurar os microsserviços para se registrarem no Eureka Server / Consul Server.

**Autenticação e Autorização:**

- Configurar o Keycloak para gerenciar a autenticação e autorização dos usuários.
- Proteger os endpoints dos microsserviços com OAuth2 e JWT.
- Definir roles (ex.: ADMIN, USER) para controlar o acesso às funcionalidades.

**CRITÉRIOS DE AVALIAÇÃO:**

✅1. Boa organização do código (camadas separadas para controller, service e repository)

✅2. Utilização de boas práticas de API REST.

✅ 3. Funcionalidade completa e funcionamento do CRUD.

✅4. Integração bem-sucedida com a API externa.

✅5. Comunicação bem-sucedida entre Microsserviços .

✅6. Uso do Swagger para documentação e collections do Postman

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

✅8. Mais de 4 status code

✅9. Readme

✅10. RabbitMQ

✅11. Uso de Conventional Commits

✅12. Containerização com Docker (EXCEEDS)

✅13. Testes Unitários (EXCEEDS)

✅14. Keycloak (EXCEEDS)

✅15. OAuth2 e JWT (EXCEEDS)

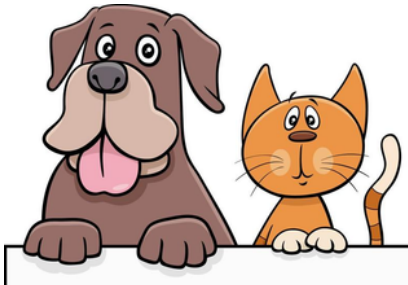
✅16. Eureka / Consul (EXCEEDS)

✅17. Dockerfiles e Docker Compose (EXCEEDS)

✅18. Envio de e-mail (EXCEEDS)

OBS: Para o microsserviços de agendamento de cuidados escolha entre o Fluxos Síncrono e Assíncrono (p. 2) ou Fluxos Assíncronos (p. 3) .

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!



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

# ⭐Sucesso no Desafio!⭐