CARTA-DESAFIO

Auditoria

Descrição

Todas as inserções, atualizações e exclusões de uma entidade devem gerar registro de auditoria acessível em `GET /log`.

Exemplos de implementação

- Auditar `Sensor`, `Reserva`, `Usuário` ou `Veículo`.
- Salvar ação, data/hora, usuário responsável, id do registro e origem da requisição.
- Registrar estado anterior e novo estado, quando aplicável.
- Filtrar com `GET /log?entity=sensor&action=update`.

Passo a passo para implementação

1. Escolher uma entidade real do domínio.
2. Definir quais eventos serão auditados e quais dados serão registrados.
3. Persistir o log a cada `POST`, `PUT/PATCH` e `DELETE`.
4. Expor rota de consulta com filtros básicos.

> ✅ **Critério de validação**
> A equipe deve demonstrar que cada operação de criação, atualização e exclusão gera log persistido e que `GET /log` retorna informações suficientes para identificar entidade, operação e momento do evento.

> ⚠️ **Atenções importantes**
> A funcionalidade deve estar integrada ao domínio principal da aplicação. Não vale criar entidades artificiais apenas para cumprir a carta.

Rota mínima: `GET /log` Log persistido é obrigatório

CARTA-DESAFIO

Ranking de Popularidade

Descrição

Uma entidade deve possuir contador de acessos. A cada `GET`, esse contador deve ser incrementado em 1. O `GET ALL` deve permitir ordenação por popularidade.

Exemplos de implementação

- `Projeto.accessCount`, `Evento.visualizacoes` ou `Produto.totalAcessos`.
- `GET /projetos/5` incrementa o contador.
- `GET /projetos?sort=popularidade_desc` retorna os mais acessados.
- `GET /eventos?sort=popularidade_asc` ordena do menor para o maior.

Passo a passo para implementação

1. Escolher a entidade do domínio que terá popularidade.
2. Adicionar um campo persistido de contagem de acessos.
3. Incrementar esse campo nas requisições `GET` adotadas pela equipe.
4. Implementar ordenação por popularidade no endpoint de listagem.

> ✅ **Critério de validação**
> O contador precisa existir de forma persistida, precisa ser atualizado nas consultas definidas para a entidade e o `GET ALL` deve permitir ordenar pelo valor desse contador, com resultado verificável na prática.

> ⚠️ **Atenções importantes**
> A funcionalidade deve ser aplicada a uma entidade do domínio principal. Não vale criar entidade artificial só para armazenar ranking ou acessos.

Persistência do contador: valor salvo no banco Ordenação no `GET ALL` deve funcionar

CARTA-DESAFIO

Relatório Agregado

Descrição

Uma entidade deve permitir relatório filtrado por período de tempo, retornando a quantidade de registros encontrados e o total de registros no intervalo.

Exemplos de implementação

- `GET /coletas?periodo=30dias`.
- `GET /vendas?dataInicio=2026-05-01&dataFim=2026-05-31`.
- `GET /consultas/relatorio?periodo=semana`.
- Resposta com período, quantidade retornada e total no período.

Passo a passo para implementação

1. Escolher a entidade e a data usada no filtro.
2. Definir parâmetros de período, intervalo ou datas inicial/final.
3. Montar consulta que conte registros filtrados e totalize o período.
4. Expor retorno com dados agregados e parâmetros usados.

> ✅ **Critério de validação**
> Deve existir endpoint de relatório temporal em uma entidade real do domínio, com filtro por período funcionando e resposta contendo pelo menos a contagem dos registros retornados e o total de registros no intervalo analisado.

> ⚠️ **Atenções importantes**
> A agregação deve usar dados da própria entidade escolhida. Não vale criar entidade paralela apenas para gerar relatório e cumprir a carta.

Use datas da própria entidade

Retorno precisa ser agregado

CARTA-DESAFIO

Regras Temporais

Descrição

Uma entidade deve permitir criação ou modificação apenas em dias e horários específicos da semana.

Exemplos de implementação

- `Reservas` só podem ser criadas de segunda a sexta, das 8h às 18h.
- `Agendamentos` só podem ser alterados até as 20h.
- `Ordens de manutenção` só podem ser abertas em horário comercial.
- Fora da janela, a API retorna erro com mensagem clara.

Passo a passo para implementação

1. Escolher a entidade e as operações afetadas.
2. Definir janela permitida de dias e horários.
3. Validar a regra antes de salvar a operação.
4. Bloquear tentativas fora do período com resposta apropriada.

> ✅ **Critério de validação**
> A restrição de dia e horário precisa estar implementada no backend, bloqueando operações fora da janela permitida e retornando código HTTP coerente com mensagem explicativa.

> ⚠️ **Atenções importantes**
> A regra temporal deve fazer sentido para uma entidade do domínio principal. Não vale criar uma entidade fictícia apenas para demonstrar a restrição.

Validação no servidor

Erro fora da janela é obrigatório

CARTA-DESAFIO

Inativos

Descrição

Uma entidade deve possuir um atributo que indique a última atualização e uma rota para listar os registros inativos por mais de uma semana.

Exemplos de implementação

- `Equipamento.updatedAt`, `Usuário.lastUpdate` ou `Lote.ultimaMovimentacao`.
- `GET /equipamentos/inativos`.
- `GET /usuarios/inativos?dias=7`, se a equipe quiser generalizar.
- Filtrar registros sem atualização há mais de 7 dias.

Passo a passo para implementação

1. Escolher a entidade e garantir o campo de última atualização.
2. Atualizar esse campo sempre que o registro mudar.
3. Criar rota para buscar itens acima do limite de inatividade.
4. Retornar apenas os registros realmente inativos.

> ✅ **Critério de validação**
> A equipe deve persistir a data da última atualização na entidade e implementar rota que retorne somente os registros acima do período mínimo de inatividade definido na carta.

> ⚠️ **Atenções importantes**
> O campo de atualização deve pertencer a uma entidade real do domínio. Não vale criar cadastro artificial apenas para listar itens inativos.

Mínimo: mais de 1 semana

Baseado na data atual

CARTA-DESAFIO

Limite de Requisições

Descrição

Implementar rate limit para retornar uma entidade ou uma lista, com configuração alterável por minuto e por hora.

Exemplos de implementação

- Limitar `GET /leituras` para 30 requisições por minuto e 300 por hora.
- Aplicar em `GET /sensores` e `GET /sensores/:id`.
- Definir limites via `.env`, banco ou rota administrativa.
- Bloquear excessos com `429 Too Many Requests`.

Passo a passo para implementação

1. Escolher a rota ou entidade que terá limitação.
2. Definir chaves de controle, como IP, usuário ou token.
3. Configurar limites por minuto e por hora.
4. Bloquear excessos e expor forma de alterar os valores.

> ✅ **Critério de validação**
> A API deve bloquear excessos no endpoint escolhido usando pelo menos dois limites configuráveis, um por minuto e outro por hora, e a equipe deve conseguir demonstrar a alteração desses valores e o retorno apropriado quando forem ultrapassados.

> ⚠️ **Atenções importantes**
> A limitação deve ser aplicada a uma funcionalidade real da aplicação. Não vale criar endpoint artificial apenas para provar o rate limit.

Dois limites: minuto e hora

Configuração deve ser alterável

CARTA-DESAFIO

Cache

Descrição

Implementar cache em uma entidade para reduzir o tempo de resposta, com tempo de expiração configurável.

Exemplos de implementação

- Cachear `GET /sensores` por 60 segundos.
- Usar Redis, memória local ou cache distribuído.
- Invalidar cache em criação, atualização e exclusão.
- Permitir configurar o TTL via `.env`.

Passo a passo para implementação

1. Escolher o endpoint de leitura mais adequado para cache.
2. Definir chave de cache e tempo de expiração.
3. Servir respostas repetidas a partir do cache.
4. Invalidar ou atualizar o cache quando os dados mudarem.

> ✅ **Critério de validação**
> Deve existir cache funcional em pelo menos um endpoint de entidade, com expiração configurável, possibilidade de alterar esse tempo e demonstração de que leituras repetidas usam o cache sem manter dados desatualizados após mudanças relevantes.

> ⚠️ **Atenções importantes**
> O cache deve otimizar uma rota real da aplicação. Não vale criar entidade ou endpoint artificial apenas para mostrar a técnica.

TTL configurável

Invalidação é parte da solução

CARTA-DESAFIO

Exportação

Descrição

Implementar rota para exportar os dados de uma entidade no formato CSV e retornar o arquivo gerado para download.

Exemplos de implementação

- `GET /coletas/export?format=csv`.
- `GET /usuarios/export?format=csv`.
- Definir `Content-Type` e `Content-Disposition` corretamente.
- Gerar CSV com cabeçalhos e linhas da entidade.

Passo a passo para implementação

1. Escolher a entidade que poderá ser exportada.
2. Definir quais campos irão para o arquivo.
3. Gerar o conteúdo CSV a partir dos dados persistidos.
4. Retornar o arquivo para download pela rota de exportação.

> ✅ **Critério de validação**
> A carta é validada quando a rota gera um CSV real com dados da entidade do domínio e o cliente consegue baixar o arquivo corretamente no formato solicitado.

> ⚠️ **Atenções importantes**
> A exportação deve usar uma entidade real da aplicação. Não vale criar entidade artificial somente para atender à carta.

Saída mínima: CSV

Download deve funcionar

CARTA-DESAFIO

Importação

Descrição

Implementar importação de dados via arquivo CSV. O upload deve responder imediatamente e o processamento precisa ocorrer em segundo plano.

Exemplos de implementação

- `POST /produtos/import` com `multipart/form-data`.
- `POST /sensores/import` enviando CSV com leituras históricas.
- Retorno `202 Accepted` com identificador do processo.
- Job assíncrono valida linhas e insere registros válidos.

Passo a passo para implementação

1. Escolher a entidade que receberá dados importados.
2. Criar rota de upload de arquivo CSV.
3. Responder imediatamente e enfileirar o processamento.
4. Processar o arquivo em segundo plano e persistir os dados válidos.

> ✅ **Critério de validação**
> A rota deve aceitar CSV, responder sem processar tudo de modo bloqueante e demonstrar que o backend continua o processamento assíncrono em segundo plano, persistindo os dados importados.

> ⚠️ **Atenções importantes**
> A importação deve alimentar uma entidade real do domínio. Não vale criar cadastro artificial apenas para demonstrar upload assíncrono.

Resposta imediata

Processamento assíncrono é obrigatório

CARTA-DESAFIO

Snapshot

Descrição

Implementar snapshots de uma entidade em intervalos regulares ou manualmente, permitindo criar, listar e restaurar um snapshot específico.

Exemplos de implementação

- `POST /sensores/snapshot` cria backup manual.
- `GET /sensores/snapshots` lista backups existentes.
- `POST /sensores/restore?snapshotId=123` restaura o estado salvo.
- Agendar snapshot diário com tarefa automatizada.

Passo a passo para implementação

1. Escolher a entidade que terá snapshots.
2. Definir o formato e o local de armazenamento das cópias.
3. Implementar criação manual, listagem e restauração.
4. Adicionar rotina periódica para gerar snapshots automáticos.

> ✅ **Critério de validação**
> A equipe precisa demonstrar a criação manual, a listagem, a restauração e a existência de geração periódica de snapshots da entidade do domínio, com recuperação comprovável de um estado anterior identificável.

> ⚠️ **Atenções importantes**
> Os snapshots devem refletir uma entidade real da aplicação. Não vale criar entidade artificial apenas para cumprir o requisito.

Manual + periódico

Restaurar deve recuperar estado real

CARTA-DESAFIO

Upload de Arquivos

Descrição

Permitir upload seguro de arquivo associado a uma entidade, como foto ou documento, e possibilitar o download posterior pelo registro específico.

Exemplos de implementação

- `POST /usuarios/5/photo` envia uma foto.
- `POST /equipamentos/10/manual` envia um PDF técnico.
- `GET /usuarios/5/photo` retorna o arquivo associado.
- Validar extensão, tipo MIME e tamanho máximo.

Passo a passo para implementação

1. Escolher a entidade que receberá anexos.
2. Criar rota de upload e rota de recuperação do arquivo.
3. Validar tipo, tamanho e nome seguro do arquivo.
4. Armazenar o arquivo vinculado a um registro já existente.

> ✅ **Critério de validação**
> A implementação precisa vincular o arquivo a um registro real da entidade, armazenar o arquivo com segurança e permitir sua recuperação por rota específica. Segurança mínima obrigatória: validar tipo/extensão, validar tamanho, usar nome ou caminho seguro e rejeitar uploads inválidos com erro apropriado.

> ⚠️ **Atenções importantes**
> O upload deve estar ligado a uma entidade real da aplicação. Não vale criar entidade artificial só para anexar arquivo e cumprir a carta.

Upload e download são obrigatórios

Segurança mínima deve ser demonstrada

CARTA-DESAFIO

Notificações via Email

Descrição

Implementar envio de emails quando determinadas ações ocorrerem na API. A rota pode permitir configurar o envio e gerenciar preferências, se a equipe desejar.

Exemplos de implementação

- Enviar email ao criar ou atualizar um `Agendamento`.
- Notificar alteração de status de `Pedido` ou `Chamado`.
- `PUT /usuarios/5/notificacoes` pode definir preferências de recebimento.
- Usar provedor real, sandbox ou logger de emails para demonstração.

Passo a passo para implementação

1. Escolher quais eventos do domínio irão disparar emails.
2. Definir destinatários e conteúdo básico da mensagem.
3. Integrar com serviço de envio real ou ambiente de testes.
4. Registrar ou demonstrar o disparo do email após o evento.

> ✅ **Critério de validação**
> A equipe deve demonstrar o disparo de email em eventos definidos de uma entidade do domínio. Configuração extra de preferências pode existir, mas não é obrigatória para validar a carta.

> ⚠️ **Atenções importantes**
> O envio deve estar integrado a um fluxo real da aplicação. Não vale criar entidade artificial apenas para disparar emails e cumprir a carta.

Disparo de email é o núcleo da carta

Preferências são opcionais