Compatibilidade de API

Preserve o contrato das versões instaladas

O app e o servidor evoluem em ritmos diferentes

Uma versão mobile já instalada pode chamar a API meses após o lançamento. Antes de mudar um endpoint, liste os campos, tipos, valores e comportamentos que os consumidores existentes usam. Inclua erros, ordenação e paginação se fizerem parte do contrato.

Omissão também tem significado

Ao adicionar um parâmetro opcional, mantenha o comportamento anterior quando ele vier ausente. Se a listagem antes devolvia todos os agendamentos, impor um limite por padrão pode ocultar registros em clientes que desconhecem a paginação.

Campos novos precisam de verificação

Um consumidor que ignora propriedades desconhecidas tende a aceitar um campo a mais. Um validador estrito pode rejeitá-lo. Teste os consumidores suportados, inclusive quando a mudança adiciona um valor de enum à resposta.

Compatibilidade de API

Adicione o campo e mantenha o original

Exemplo em JavaScript

```
function respostaV1(reserva) {
  return {
    id: reserva.id,
    horario: reserva.inicio,
    inicio: reserva.inicio,
  };
}
function lerHorarioV1(resposta) {
  if (typeof resposta.horario !== "string") {
    throw new Error("Resposta incompatível");
  }
  return resposta.horario;
}
```

Leitura do caso

O servidor passou a chamar o atributo interno de inicio. A resposta de GET /v1/reservas/:id continua expondo horario como texto no mesmo formato, com início da reserva no mesmo fuso convencionado. O campo inicio serve à migração se os consumidores de v1 aceitarem propriedades adicionais.

Evolução

Se algum consumidor de v1 rejeitar campos extras, preserve integralmente a resposta anterior e ofereça a nova representação em v2. Rode testes de contrato a cada mudança. Para retirar v1, anuncie o prazo e acompanhe as versões de aplicativo ainda em uso.

Compatibilidade de API

Exercício resolvido

Cenário

Uma versão entregue à loja lê horario. A API nova responde apenas com inicio. O cliente recebe HTTP 200 e falha ao renderizar. Como restaurar o fluxo sem aguardar a atualização pela loja?

Resolução

Restaure horario na representação v1 com o formato e significado anteriores. Execute o leitor antigo com uma resposta publicada e confirme a hora exibida. Teste a ausência de horario como caso de regressão. Se o leitor antigo rejeitar chaves extras, inclua inicio somente na representação v2.

Verificação de contrato

No exemplo, o leitor antigo deve produzir a mesma data tanto com a resposta original quanto com a resposta aditiva. O teste com apenas inicio deve falhar no leitor. Para cobrir integração, repita com uma versão instalada apontada para a API de homologação.

Referências primárias

Google AIP-180 — Compatibilidade

Pact — Verificação do provedor

Compatibilidade de API

Teste a API com versões instaladas diferentes

Considere valores novos em campos conhecidos

Adicionar um estado pode quebrar um cliente que enumera todos os casos possíveis. Teste a resposta com os aplicativos suportados. Em clientes novos, trate um estado desconhecido explicitamente e deixe ações sensíveis indisponíveis até consultar o estado correto.

Exercício adicional

O aplicativo antigo reconhece pendente e confirmado. A API quer devolver em_analise. Como publicar?

Resolução

Mantenha a representação que o cliente antigo entende se houver um estado equivalente para as ações permitidas. Se a equivalência não existir, use uma versão nova do contrato. Teste o comportamento nas duas versões do aplicativo.

Ao assistir ao vídeo

Observe como versões distintas podem continuar disponíveis durante uma migração.

Vídeo complementar

Implementing Modern API Versioning in .NET (em inglês)