**B-Health**

Você é o novo tech lead da squad que cuida do sistema de agendamento de exames, que está com os seguintes problemas:

1. Diariamente todos os hospitais enviam via FTP um arquivo para atualizar os exames, que são processados por um robô que faz a classificação e inclusão no banco de dados. Neste processo ocorrem diversos erros, como processamento de arquivos muito grandes, concorrência de I/O, falha no upload/download do arquivo, que deixam os exames desatualizados. Ficou definido com os hospitais que a comunicação não deverá mais ser feito por arquivos.

◦ Qual solução você daria para essa integração? Descreva a solução e a arquitetura que você faria.

2. Existe um front-end que o cliente final pode agendar exames, realizar o pagamento e consultar os dados do hospital. Da forma que foi construída, cada consulta dos exames abre uma conexão com o banco e nosso banco permite até 100 conexões simultâneas. Com novos hospitais sendo integrados, hoje já tem mais de 1000 usuários acessando o site ao mesmo tempo, com muitas reclamações de performance e erros de consulta.

◦ Qual solução você daria para que suporte o aumento contínuo de usuários simultâneos, sem envolver alguma alteração de hardware (servidores e bancos)? Descreva a solução e a arquitetura que você faria.

3. Pensando nas soluções propostas dos itens anteriores, quais boas práticas você aplicaria no processo de entrega em produção, pensando nos conceitos de disponibilidade, qualidade e segurança?

**01 - Problema de integração/atualização por troca de arquivos via FTP.**

• [Desenho da Solução](https://whimsical.com/01-b-health-STtPbqzDjYUZ3cjogaUYYs)

**Solução 01 - Centralização de gerenciamento de informações - Integração via Api**

• Substituir a integração diária via arquivo por integrações periódicas efetuando sincronização das bases de registro de exame com o sistema que é responsável pelo gerenciamento central das informações no banco de dados via API.

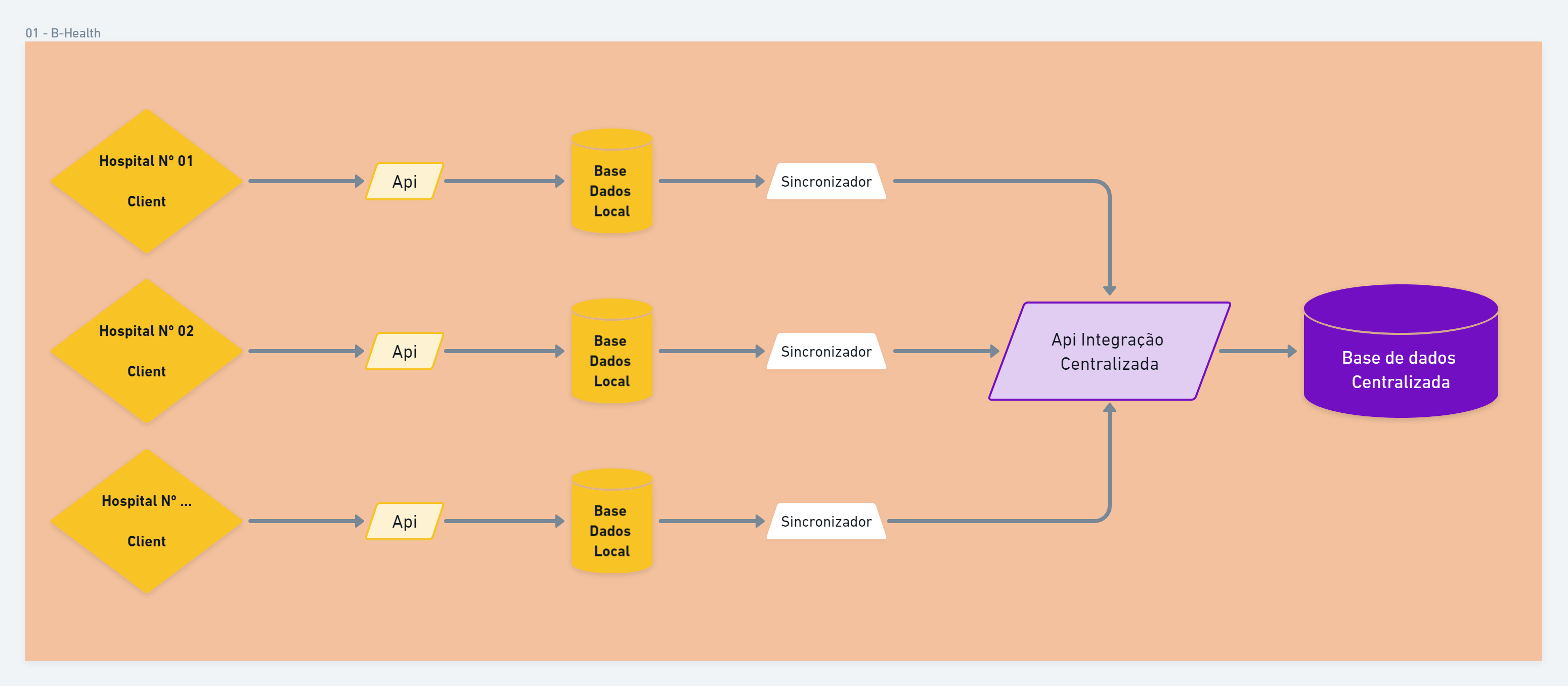
◦ Cada Hospital teria uma base de dados local para gerenciar os exames.

◦ Cada registro de exame seria gravado nessa base de dados no momento do lançamento e geraria um evento de gravação.

◦ Periodicamente os registros seriam enviados para o serviço responsável pela base de dados geral dos hospitais.

◾ As integrações periódicas teriam um curto período de dessincronismo.

◾ Em caso de indisponibilidade do serviço centralizado, a próxima tentativa de integração resolveria a falha anterior.



**Solução 02 - Centralização de gerenciamento de informações - Geração de**

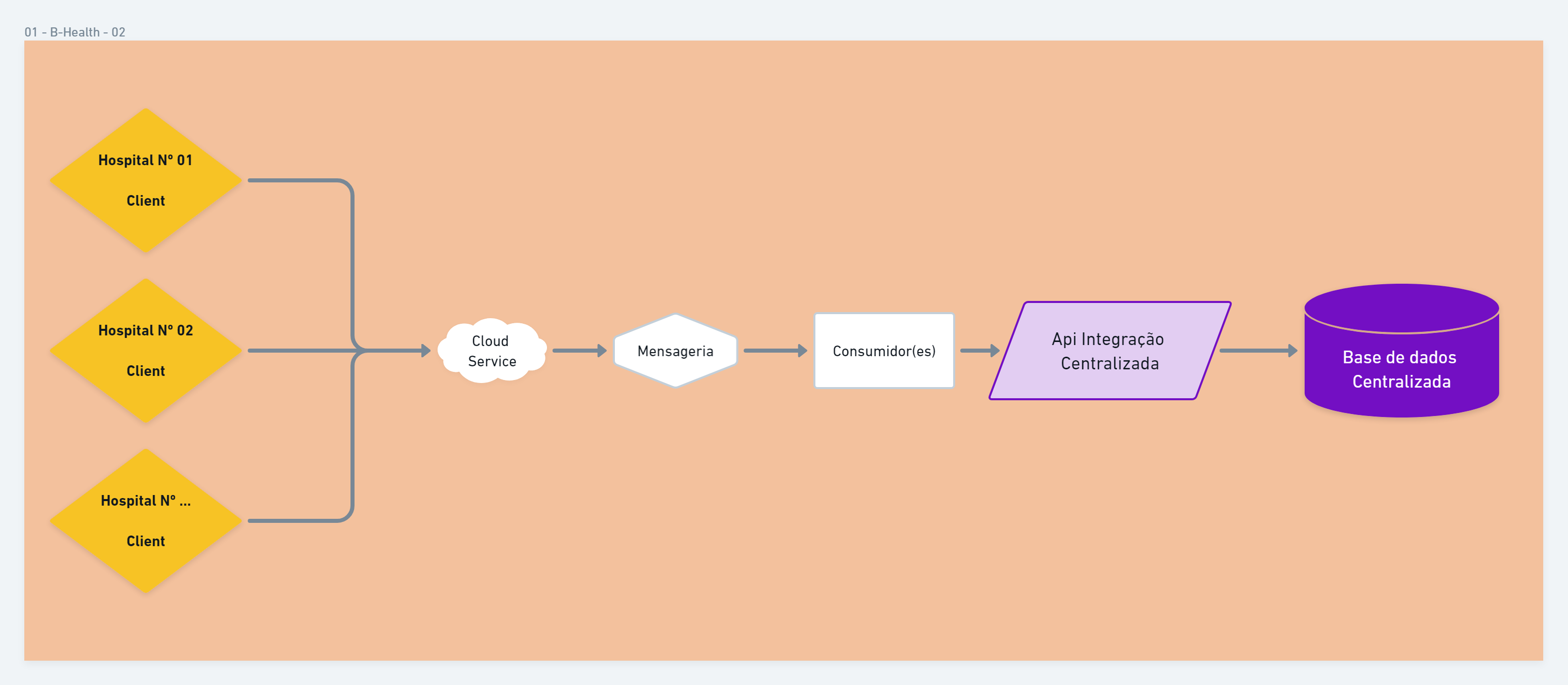
• Substituir a integração diária via arquivo por integrações periódicas onde cada registro de exame geraria um evento para um serviço de mensageria que seria responsável por integrar os registros com a base de dados centralizada via API.

◦ Cada Registro de exame geraria um evento que seria disparado para um consumidor ser responsável por integrar diretamente com o serviço centralizador.

◦ Periodicamente os registros seriam enviados para o serviço responsável pela base de dados geral dos hospitais.

◾ As integrações periódicas teriam um curto período de dessincronismo.

◾ Em caso de indisponibilidade do serviço centralizado, retentativas seriam feitas até que o serviço fosse Reestabelecido.



**02 - Problema de concorrência de acesso a base de dados.**

• [Desenho da Solução](https://whimsical.com/02-b-health-GfgJXG1GCFiGLg8FncJPrK)

**Solução 01 - Centralizar acesso a base e aplicar 'Repository Pattern' com gerenciamento 'Singleton' de acesso a base de dados.**

• Desacoplar o front-end do back-end.

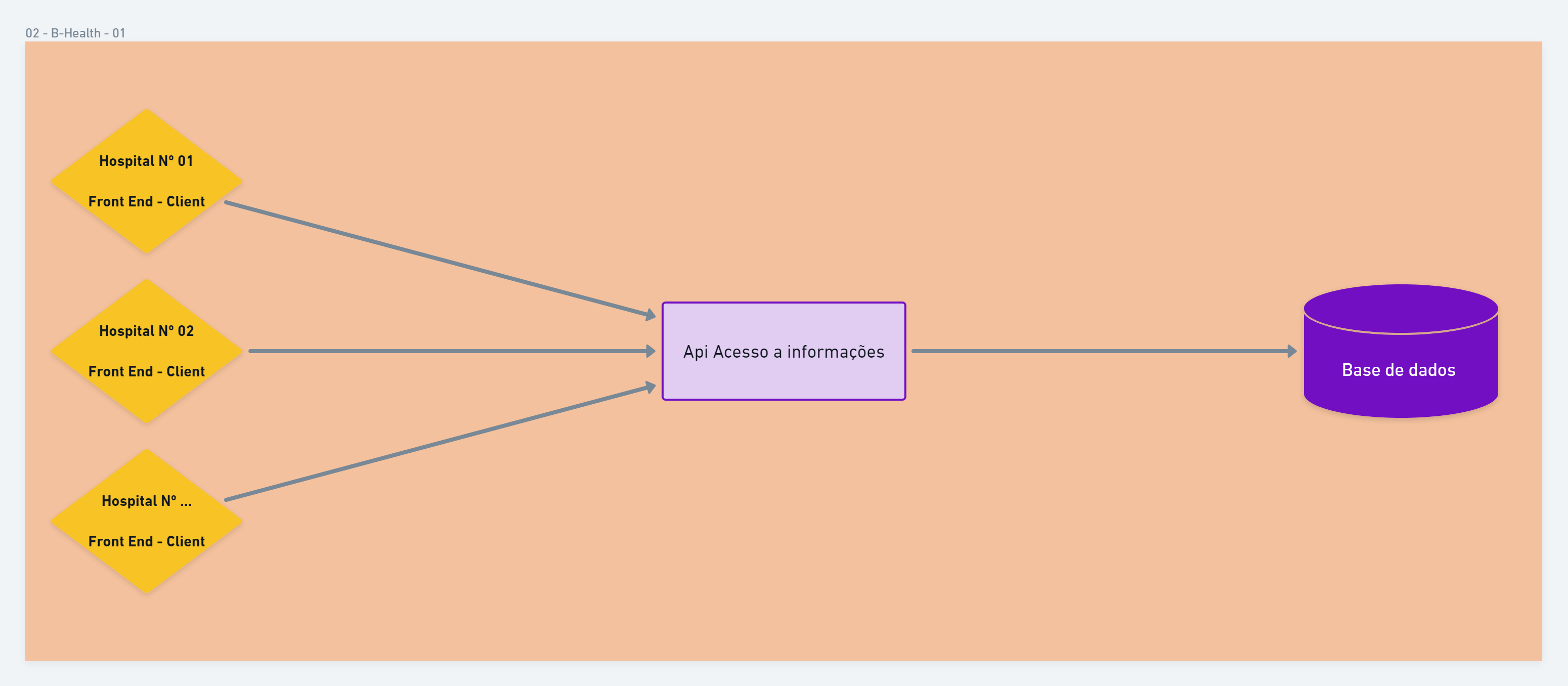
• Criar uma API única para acesso a base de dados.

◦ Aplicar padrão de acesso a dados de forma gerenciada.

◾ Limitar o número de conexões abertas por requisição.

◾ Implementar padrão singleton com gerenciamento de abertura e fechamento de seções.

• Front-end deverá consultar as informações através da API criada.



**03 - Boas práticas para disponibilidade, qualidade e segurança.**

**Disponibilidade**

• Solução publicada em Núvem (quando possível).

• Aplicar solução em multicloud e multiregion (quando possível).

• Servidores de aplicação com redundância e balanceador de cargas.

• Aplicações conteinerizada com auto scaling.

• Base de dados com redundância e balanceador de cargas.

• Monitoramento ativo de aplicações com alertas.

**Qualidade**

• Testes unitários nas aplicações.

• Análise de código estática e humana. (Code Review).

• Esteira de publicação com testes automatizados (Unitário, regressivo, carga).

**Segurança**

• Ambiente com acesso limitado a equipes especificas.

• Integrações autenticada entre aplicações.

• Integrações HTTPs criptografadas.