**PARTE III - ARQUITETURA**

Como o case é bem genérico (por se tratar de um teste), algumas "decisões" vão tender a um lado mais agnóstico de tecnologia, plataforma ou outras peculiaridades do projeto.

Para novas aplicações sendo desenvolvidas para compor esse grupo de aplicações, elas poderiam seguir arquiteturas como a Clean Architecture ou Arquitetura Hexagonal. Ambas podem ser implementadas em diversas plataformas, e o que esses dois modelos possuem em comum é que mantém em seu "centro" o domínio e seus casos de usos livres de dependências pois são componentes de alto nível. Os componentes de mais baixo nível, relacionados a infraestrutura, clientes web para comunicação com outros sistemas, repositórios de dados e componentes de IO em geral (sistema de arquivos) são mantidos nas camadas mais "externas" da arquitetura.

A forma de integração / comunicação entre os sistemas seria baseada em Web APIs pois estão disponíveis em todas as plataformas e alinhadas com a arquitetura definida acima, seriam mínimas as necessidades de reescrita ou modificações de código. Dependendo da demanda, da natureza dos projetos e da quantidade de aplicações fazendo parte desses ambientes. Talvez fosse interessante aplicar Event Sourcing e/ou CQRS.

Com o Event Sourcing, cada ação dentro do sistema seria gerado um evento que posteriormente seria consumido por uma outra aplicação ou serviço responsável pelo processamento.

O modelo lida bem com escalabilidade pois permite pools de workers para fazer o processamento de um determinado evento conforme a demanda e se torna uma peça central de mensageria ou orquestração do sistema como um todo.

Falando sobre o CQRS essa abordagem determina um conjunto separado de comandos para leitura e escrita na aplicação, fazendo que cada modalidade passe por um fluxo com tecnologias diferentes para escrita e leitura.

A leitura pode ser feita em bancos de dados não relacionais ou bancos em memória (algo que seja mais rápido no geral que executar queries em bancos rllacionais).

A escrita pode ser direta em bases relacionais ou gerar um evento por exemplo para que outra aplicação faça a leitura de uma fila e de fato consolide a alteração.

Essas bases de dados (de leitura e escrita) devem ser sincronizadas, a frequência e o modo que essa sincronização aconteceria pode depender bastante das necessidades de negócio da aplicação.

Sobre a comunicação ou integração com Mainframe, não obtive experiencia com esse cenário. Eu optaria por "abstrair / isolar" o Mainframe através de uma aplicação que fique na frente dele servindo de gateway para outras aplicações consumirem os serviços do Mainframe. Essa app seria quase que um adaptador, implementando a forma necessária para comunicação com o Mainframe e disponibilizando para as outras aplicações através do modelo de integração especificado acima (Web APIs). Isso caso já não fosse nativo da plataforma do Mainframe.

Considerando que a empresa e o projeto não coloquem restrições quanto a utilização de cloud pública.

Por se tratar de sistemas críticos e de alta performance, a utilização da nuvem torna muito fácil e ágil a implementação desse ambiente garantindo HA, tolerância a falhas e escalabilidade conforme a demanda.

Características que tornam bem atrativas considerando que o projeto é de um sistema crítico para o negócio. Serviços Paas de host e aplicações ou de containers da nuvem podem rodar as aplicações isoladas e de forma escalável acompanhando a demanda.

Considera que nesse cenário, podemos utilizar ferramentas de Deploy automático CI/CD para automatizar pipelines de execuções de testes, compilação e deploy e também o rollback em caso de problemas.