Caso de Uso: Modelo de Arquitetura para um Sistema de Cadastro de Clientes

Objetivo

O arquiteto de TI deve projetar uma arquitetura de solução segura, eficiente e escalável a fim de implementar um sistema de cadastro de clientes extremamente crítico. O sistema deve atender aos requisitos funcionais e não funcionais, como por exemplo transações ACID e conformidade com a legislação de proteção de dados. O departamento de marketing também deve receber notificações quando um novo registro for adicionado.

Contexto

Uma empresa de metalurgia tinha uma grande demanda de clientes e projetos em andamento, mas enfrentava dificuldades para gerenciar as informações e manter um registro atualizado dos dados dos clientes. O processo de cadastro era realizado de forma manual e desorganizada, com cada departamento mantendo suas próprias planilhas e registros, o que levava a inconsistências e erros nos dados.

Com o passar do tempo, a empresa percebeu que essa falta de organização e controle estava afetando seus negócios, pois não conseguia identificar facilmente quais clientes estavam em dia com pagamentos, quais projetos estavam em andamento, quais eram as preferências e necessidades de cada cliente, entre outras informações importantes. A empresa também percebeu que, sem um sistema de cadastro de clientes, estava perdendo oportunidades de negócios, pois não conseguia realizar campanhas de marketing personalizadas e não tinha uma visão completa do histórico de cada cliente.

Foi então que a empresa decidiu implementar um sistema de cadastro de clientes, que deve permitir o registro de todas as informações relevantes de cada cliente, incluindo dados pessoais, histórico de compras, preferências, pagamentos, entre outros.

Tendo esse objetivo, a empresa precisa contar com a expertise de um arquiteto de TI que possa modelar esse caso de uso de forma eficiente e escalável. O arquiteto deve projetar uma solução que leve em consideração segurança, desempenho, escalabilidade e facilidade de manutenção. Dessa forma, será possível garantir que o sistema atenda às necessidades da empresa e possa crescer junto com o negócio, sem comprometer a qualidade e a segurança dos dados dos clientes.

Atores

Administradores: são responsáveis por gerenciar o sistema como um todo, incluindo cadastro de usuários, configurações gerais, permissões de acesso, entre outros. Eles possuem acesso completo ao sistema e podem realizar qualquer operação disponível.

Atendentes: são usuários que têm acesso ao sistema para realizar cadastros de clientes, atualizações de informações e consultas em geral. Eles podem ser responsáveis por atendimentos telefônicos, presenciais ou online, e precisam ter acesso rápido às informações dos clientes para prestar um atendimento eficiente.

Clientes: são os usuários que acessam a aplicação via internet e realizam o cadastro.

Contexto de Tecnologia

Atualmente a empresa não possui qualquer sistema, mas ela tem o interesse em criar esse Sistema de Cadastro de Clientes e utilizar o provedor de nuvem Microsoft Azure nessa modernização.

A empresa também deseja trabalhar com uma esteira de *DevSecOps* integrado ao provedor de nuvem a fim de evitar problemas de *deploy*.

Detalhamento dos Componentes

Aplicação Web (*Frontend*)

Considere um aplicativo web composto de uma interface web (SPA) desenvolvida em Angular. O objetivo é onde hospedar esse SPA de maneira que ele atenda todos os requisitos não funcionais (descritos no final do documento).

Endpoints de API (*Backend*)

Considere APIs desenvolvidas em Java que devem se integrar com o *Frontend*. Assim como na Aplicação Web, o objetivo é onde colocar essas APIs de maneira que ele atenda todos os requisitos não funcionais (descritos no final do documento).

Sistema de Autenticação e Autorização

Para operações no Sistema de Cadastro de Clientes, o usuário deverá realizar o processo de autenticação e estar devidamente autorizado para uso das funcionalidades. Esse processo é de responsabilidade de um *Identity Provider* (IDP) e está fora do escopo do Sistema de Cadastro de Clientes. O IDP está integrado com um *API Gateway*.

Banco de Dados

O modelo de banco de dados que deverá ser utilizado nessa solução devesuportar todas as propriedades de transações ACID.

Notificações

O Departamento de Marketing sinalizou a necessidade de receber uma notificação do Sistema de Cadastro de Clientes quando um novo registro for adicionado. Outro ponto, é que se faz necessário também notificar o sistema interno de LGPD que se encontra *on-premise*, informado que um novo registro foi efetuado e que o cliente

deu o consentimento para que suas informações pessoais sejam coletadas e utilizadas pela empresa.

Logs

Para esse sistema, há dois tipos de logs que deverão ser registrados.

1. **Logs de Negócio**: são usados para registrar as informações relacionadas ao processo de negócio que ocorreu no sistema. Esses logs geralmente incluem informações sobre o que aconteceu no sistema, por exemplo: uma transação bem-sucedida, um usuário registrado com sucesso, entre outros eventos relevantes para o negócio.

2. **Logs Técnicos**: registram informações relacionadas ao funcionamento interno do sistema, como erros de programação, tempo de resposta do servidor, uso de recursos, entre outros aspectos técnicos. Eles serão usados principalmente para monitorar o sistema, diagnosticar problemas e otimizar o desempenho. Os logs técnicos deverão ser direcionados para um sistema integrado de observabilidade (métricas, logs, *tracing*, visualização e alertas).

Aspectos Não Funcionais

Segurança: o sistema deve ser seguro e protegido contra ataques e invasões, garantindo a privacidade e confidencialidade dos dados dos clientes;

Escalabilidade: o sistema deve ser capaz de lidar com grandes quantidades de dados e usuários simultaneamente, sem afetar seu desempenho ou disponibilidade;

Confiabilidade: o sistema deve ser confiável e estar sempre disponível, sem falhas ou interrupções inesperadas;

Desempenho: o sistema deve ter um bom desempenho, garantindo respostas rápidas e eficientes, mesmo em momentos de pico de uso;

Manutenibilidade: o sistema deve ser fácil de manter e atualizar, permitindo que os desenvolvedores realizem alterações e correções sem afetar seu funcionamento e sem introduzir novos problemas.

Resiliência: o sistema deve ser tolerante a falhas o tornando resiliente em eventuais problemas;

Entregáveis

Como resultado desse trabalho esperamos receber:

1. Diagrama de Contexto (sugestão C4 Model)
2. Diagrama de Container/Solução (sugestão C4 Model)
3. Diagrama de Infraestrutura (Modelo livre)