CovidShot
Documento de Definição dos Estilos Arquiteturais

1. Introdução

1.1. Finalidade

O presente documento tem como objetivo apresentar como o software CovidShot está arquitetado, desde seus conceitos básicos, seus estilos arquiteturais (Cliente-Servidor e Em Camadas) e como estes estruturam o sistema.

1.2. Escopo

CovidShot é um software cuja funcionalidade principal é o agendamento de vacinas, possibilitando o uso por diversos agentes governamentais e presente no mundo todo. O sistema auxilia no controle e gestão de horários, quantidades e fabricantes, além de mostrar dados em tempo real sobre os casos de Covid-19 e sobre as doses aplicadas no país de escolha (de acordo com a língua escolhida para o software).

2. Definições, acrônimos e abreviações

Azure: é uma plataforma destinada à execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem.

Cliente-servidor: segundo Sommerville (2016), é o estilo arquitetural cliente-servidor é aquele no qual um aplicativo é modelado como um conjunto de serviços fornecidos por servidores. Os clientes podem acessar esses serviços e apresentar os resultados aos usuários finais. Clientes e servidores são processos separados.

Repository Pattern: é um padrão de projeto que permite um encapsulamento da lógica de acesso a dados, impulsionando o uso da injeção de dependência (DI) e proporcionando uma visão mais orientada a objetos das interações com o banco de dados.

SPA Pattern: é um padrão arquitetural de um aplicativo da web ou site que interage com o usuário reescrevendo dinamicamente a página da web atual com novos dados do servidor. O objetivo são transições mais rápidas que façam o site parecer mais um aplicativo nativo.

Arquitetura em Camadas: Arquitetura em camadas visa a criação de aplicativos modulares, de forma que a camada mais alta se comunica com a camada mais baixa e assim por diante, fazendo com que uma camada seja dependente apenas da camada imediatamente abaixo**.**

3. Definição dos Estilos Arquiteturais

Para o projeto CovidShot, foi escolhido após a definição e análise dos requisitos funcionais, a utilização do estilo arquitetural em Camadas em conjunto com estilo Cliente-Servidor.

De maneira mais abrangente, o sistema será dividido entre Cliente, utilizando o padrão arquitetônico SPA, na qual será instanciado no próprio navegador do usuário, subindo para deploy utilizando o sistema externo Vercel. Já o Servidor, será uma API REST dividida em quatro camadas, hospedada em um APP Service do Azure.

3.1. Camada de Aplicação

A camada de aplicação está conectada diretamente ao cliente, sendo a porta de entrada do Servidor, nela estão contidas as controllers de acesso à API, os modelos que serão utilizados na camada de apresentação (ViewModels), as conversões entre os os modelos utilizando AutoMapper e os serviços de domínio, que serão responsáveis por gerenciar o processamento da camada de Serviços além de realizar tarefas de responsabilidade do cliente.

3.2. Camada de Serviços

A camada de serviços contém todas as validações do sistema, conexões com serviços externos (Covid 19 API), conexões de mensageria utilizando Azure Functions para o envio de emails e traduções simultâneas de todo o sistema, além dos serviços que executam toda a regra de negócio da aplicação.

3.3. Camada de Domínio

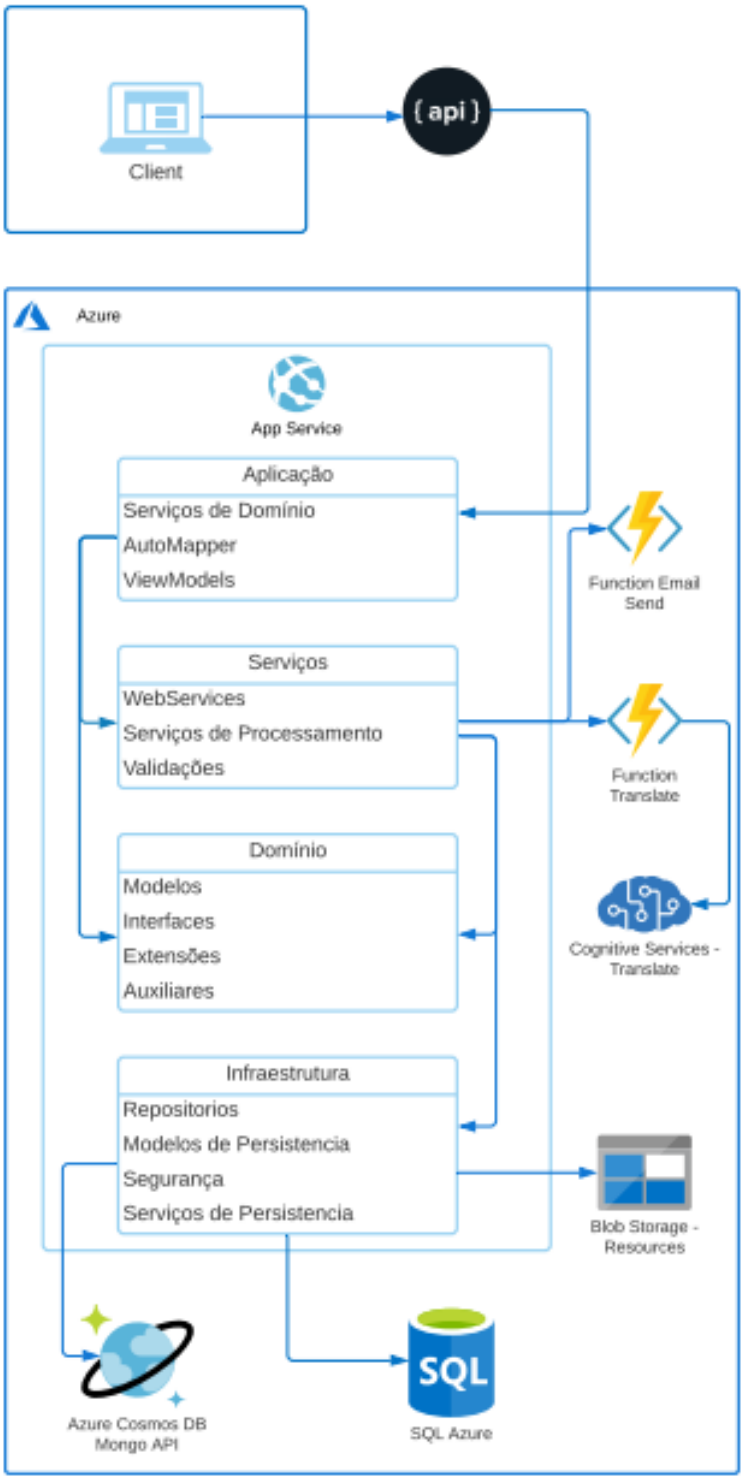
Na camada de domínio, estão localizados os modelos (entidades) que serão compartilhadas por toda a API, todas as interfaces utilizadas para a injeção de

dependência, extensões e serviços auxiliares compartilhados entre as outras camadas.

3.4. Camada de Infraestrutura

Por fim, a camada de infraestrutura contém os repositórios de acesso à persistência dos dados, separados por contextos no domínio, os modelos de persistência, sejam eles ligados aos bancos de dados NoSQL (Cosmos DB from Mongo API) ou relacionais (SQL do Azure), tratamento de segurança de acesso e serviços de persistência ligados as traduções armazenadas no Blob Storage.

3.5. Diagrama dos Estilos Arquiteturais do Sistema



4. Comportamento do Servidor dentro do Azure

Como dito anteriormente, a API de Servidor será hospedada dentro de um APP Service do Azure, em conjunto com outros recursos de sua nuvem pública para seu funcionamento escalável e manutenível.

4.1. Serviços do Azure Utilizados

4.1.1. APP Service

Será utilizado para hospedagem da API na nuvem, facilitando sua manutenção, acesso e escalabilidade.

4.1.2. Azure Cosmos DB From Mongo API

Será utilizado a instância do Cosmos DB ligada a API do MongoDB, para o armazenamento de dados NoSQL (dados das vacinas, informações e instruções de vacinações, localização de postos de vacinação, entre outros).

4.1.3. SQL Azure

Será utilizado a instância do SQL Azure (SQL Server) para o armazenamento de dados relacionais (dados do usuário, permissões e validações).

4.1.4. Azure Functions

Será utilizado para o processamento de envio assíncrono de emails aos usuários cadastrados, assim como a chamada e retorno das traduções do sistema utilizando Azure Cognitive Services.

4.1.5. Azure Cognitive Services

Será utilizado para a tradução de todos os textos, botões e informações presentes no sistema para mais de 80 idiomas utilizando tecnologia de inteligência artificial.

4.1.6. Azure Blob Storage

Será utilizado para o armazenamento dos arquivos de traduções em texto, para economizar custo e aumentar o desempenho, caso as traduções já tenham sido concluídas anteriormente.