**Análise das abordagens arquiteturais**

A arquitetura foi pensada para que os dados fossem administrados por uma API de um terceiro. Pois trata-se de um sistema onde exige-se integrações com aparelhos de geo-localização, aonde estou utilizando um player de mercado, para este fim.

A solução se complementa por outros componentes. Um serviço de geração e envio de relatórios. Onde os mesmos são consolidados em um banco de dados e posteriormente enviados em um determinado horário durante a madrugada para os e-mails dos responsáveis via “schedule”.

Além disto, este serviço será responsável por manter um histórico de relatórios consolidados, onde facilitará a busca e para regerar os mesmos.

E o Painel Admin, onde tem-se as regras de negócio do sistema dividida em camadas de aplicação.

Esta abordagem deve-se ao grande volume de tempo e processamento que a API do terceiro demanda, quando trata-se de grandes volumes de dados.

**Prova de conceito**

Foram implementadas algumas provas de conceito para validar as decisões tecnológicas tomadas.

**Design Pattern**

Linguagens de desenvolvimento

Persistencia de dados

Autenticação e autorização

**Design Pattern**

O Design em camadas permitiu uma boa organização do código, e o reaproveitamento de pacotes e componentes.

A distribuição em com Interfaces e Injeção de dependências, permite um bom nível de desacoplamento do código em geral, onde futuras evoluções e troca de componentes ficam mais fáceis de serem acopladas e desacopladas.

**Linguagens de Desenvolvimento**

E a abordagem com Asp Net core 6 mostrou-se bem performática, e fexível com eventuais atualizações e acoplamento de outras tecnologias.

Para o front-end manteve-se dentro das tecnologias do Dot Net, utilizando o Razor Page, utilizando-se de HTML5, CSS3 e Javascript. Com o back-end em C#.

**Persistencia de dados**

Para a persistência de dados foi utilizado o consumo de uma API de terceiro, realizando requisições HTTP, com protocolo REST.

E para o serviço de controle de redundância e consolidação de relatórios, foi-se utilizado o Dapper, para persistência de dados no Banco de dados relacional de controle interno. Dado a performance e o maior controle das queries.

**Demonstração da Plataforma**

As provas de conceito foram implementadas em um Paas (

Platform as a Service ) Heroku, onde diminui-se a com a complexidade de infra estrutura, ganha-se tempo de builds e deploys e ainda mantem a disponibilidade e escalabilidade do sistema. Com pipelines simples e robustas.

Segue um exemplo de uso da plataforma.

A autenticação é realizada via login e senha, onde o usuário mantem-se logado via OAuth0 e JWT para a tokenização.

Pode se navegar por todo o sistema de acordo com o nível de acesso do usuário.

No próprio dashboard tem informações rápidas e relevantes para o sistema, e filtros dinâmicos.

Podemos também gerenciar todas as etapas de localização dos veículos, cadastros e geração de relatórios, conforme demonstrarei a seguir.