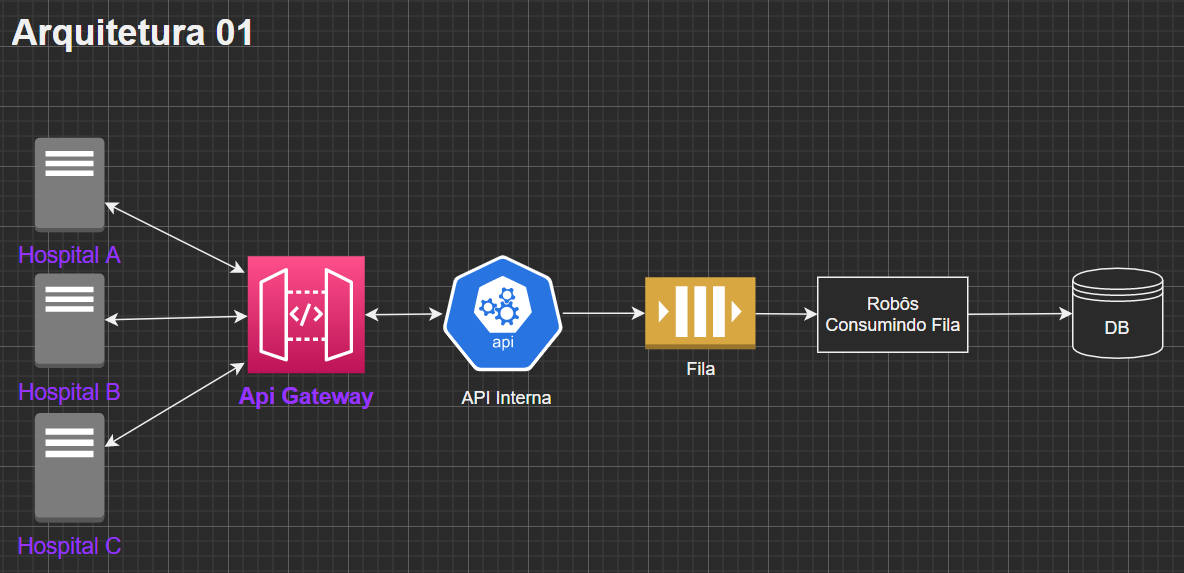
1. A integração entre os hospitais e o serviço deve ser feito de forma online, onde a cada novo registro ocorrido no hospital seja enviado uma requisição para o endpoint do nosso serviço, para que os dados de exames estejam o mais atualizado possível em nosso banco de dados (DB).

Defini a arquitetura para que seja utilizado um API Gateway servindo de camada de segurança e autenticação nesse processo, assim somente hospitais que possuem as devidas credenciais de acesso possam enviar requisições para o serviço.

Utilizaremos uma Rest API com Docker, sendo executado em múltiplas instancias de containers em um ECS (Elastic Container Service) na AWS, com devidos monitoramentos de saúde e ferramentas como o NewRelic para acompanhamento de logs, e políticas de auto escalonamento.

Quando a API recebe a requisição ela efetua a validação dos dados recebidos e em seguida a postagem desses dados em uma fila utilizando qualquer serviço de mensageria como RabbitMq ou AWS SQS, está fila será processada pelos robôs que fazem a classificação e inserção no banco de dados.

Quando o robô tiver sucesso na classificação e na inserção dos dados poderemos remover esse registro da fila, assim garantimos que mesmo que ocorram erros no processamento não ocorra perda de informações e seja feita nova tentativa de inserção no DB.



Importante notar que o aumento de usuários não deveria impactar diretamente o banco de dados, por isso algumas mudanças na arquitetura desse projeto se tornam necessárias, para lidar com as limitações de hardware dos servidores, devemos primeiro adicionar um Cloud CDN, para que o conteúdo estático do portal possua redundância em vários servidores no mundo e proximidade ao usuário final, assim garantindo baixa latência e responsividade na navegação.

Ao adicionar ao portal a compatibilidade com Progressive Web Apps (PWA), podemos utilizar do serviço de notificações (push notifications), e um cache local para navegação mesmo com servidores offline, elevando ao patamar máximo de performance e latência na navegação e experiencia do usuário.

O front-end se comunica com o backend que consiste em uma API que atende as requisições e dependendo do serviço segue um fluxo de operação especifico.

O atendimento da consulta de exames permite que o usuário no front-end efetua uma solicitação e a API recebe a solicitação e retorna à informação para o front-end que usuário deverá aguardar e será notificado quando o arquivo de exame estiver disponível para download, ao mesmo tempo ele efetua a postagem da solicitação na Fila Solicitar Resultado Exame.

A fila será processada por robôs que utilizam um numero limitado de conexões ao banco de dados utilizando uma arquitetura que tenha um pool de conexões, onde seja atendido a fila e solicitado uma conexão ao banco de dados (DB) do pool de conexões, após obter os dados no DB e gerado o arquivo e enviado para o Amazon Simple Storage Service(S3), será enviado uma notificação para o serviço de notificações (Podem ser utilizados OneSignal, AWS SNS ou mesmo um serviço sendo executado no front-end) para o recebimento dessa notificação.

Cada cliente, ou seja, o próprio navegador do usuário está executando service workers que recebem e processam essas notificações devido a implementação do PWA ele notifica mesmo se o usuário não estiver com a página aberta, funciona tanto para mobile e desktop sendo que o portal se comporta como um aplicativo.

Os arquivos serão disponibilizados no S3 localizados no Bucket Exames, ao clique do usuário no html da notificação vai ser efetuado download diretamente do S3 além disso teremos uma página para consulta de todos os resultados de exames e links para o download, onde serão armazenados em cache (Redis) para que o usuário possa consultar e efetuar o download a qualquer momento na interface do portal, sem precisar de abrir conexões com o BD.

O serviço de agendamento de exames vai ter uma fila de processamento utilizando o mesmo tipo de robôs com limitações de um pool de conexões ao banco de dados, e após o usuário na interface, obter à disponibilidade de agendamentos que está disponibilizado no cache Dados Agendamento.

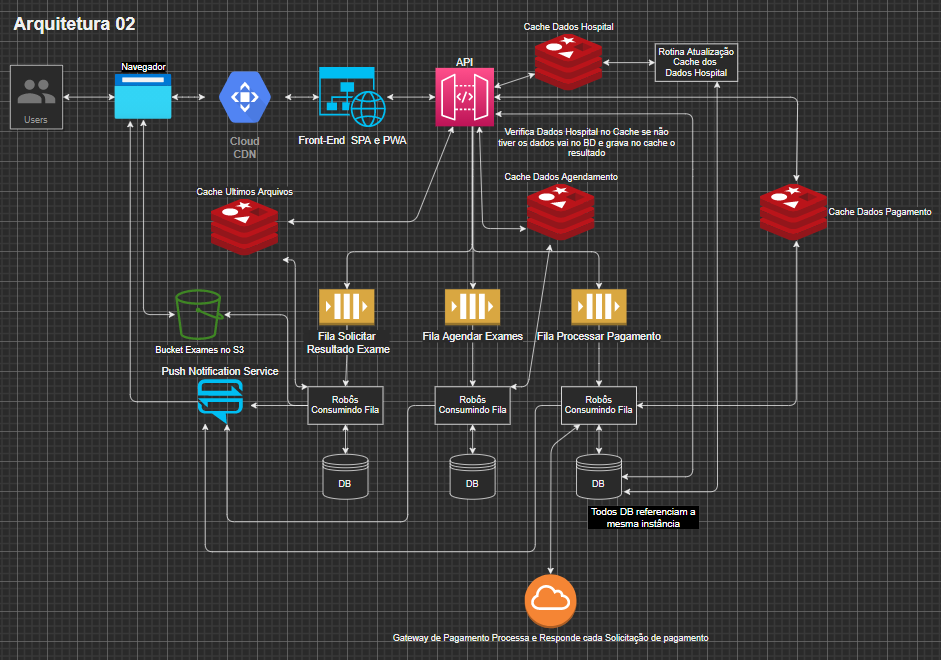
Ao solicitar um agendamento o usuário será informado que receberá uma notificação, sobre o agendamento e assim que for notificado informa se foi agendado com sucesso e o robô atualiza o cache de disponibilidade.

As solicitações de processamento de pagamento, teremos uma fila e robô de processamento com a arquitetura de um pool de conexões, e um serviço externo de processamento de pagamentos, assim que obtivermos sucesso no pagamento ou erro será enviado notificação para o usuário e no caso de erro será solicitado novamente ao usuário o os dados de pagamento e o motivo da recusa.

Os dados serão armazenados pelos robôs sem sobrecarregar a quantidade de conexões ao DB, e será atualizado o cache Dados Pagamento onde será utilizado para que o front-end exiba o histórico de cada tentativa e status do pagamento.

A exibição dos dados hospital será primeiro consultado o cache de Dados Hospital que é atualizado por uma rotina diária executada em horário definido como sendo o de menor utilização do portal, em caráter de exceção se algum hospital ainda não esteja no cache deverá ser feito a consulta diretamente no DB e inserido imediatamente no cache, para que os demais usuários não precisem efetuar solicitações ao DB para obter essas informações.

Todos os serviços e robôs podemos utilizar a mesma infra de servidores sem alterações de hardware, executando containers desses ou mesmo utilizando serviços em nuvem como o exemplo do S3.



Devemos nos preocupar, em manter o processo de entrega contínua de software onde um build de uma aplicação esteja pronto para entrega em ambiente produtivo, o time de DevOps configure todas as etapas necessárias para publicação em produção, ou seja, apenas uma vez seja compilado os binários da aplicação e todas as etapas sejam executadas e aprovadas.

E possua etapas de verificação de vulnerabilidades de softwares por exemplo a utilização de ferramentas como o SonarQube, testes unitários e automatizados através de ferramentas como Cypress ou bibliotecas de testes, e tenha ambientes de desenvolvimento, homologação e produção.

Necessitamos que o time ou profissional de Quality Assurance (QA) efetue a aprovação de etapas após a publicação em homologação, e que ocorra aprovação do usuário ou responsável pelo negócio depois da aprovação de QA.

Importante que tenha uma última aprovação para subida em ambiente produtivo e o envio automatizado de e-mails informando sobre a manutenção e relatório de mudanças da versão.

Para garantirmos menor impacto da subida de versões devemos nos preocupar em trabalhar com a conteinerização de aplicações para garantir que a subida seja efetuada através de imagens onde a cada alteração e subida de containers sejam verificados a saúde e disponibilidade dessa nova instancia e assim a cada alteração em ambiente produtivo seja evitado problemas de indisponibilidade, e preocupação também com a forma de implementação de funcionalidades e melhorias onde cada alteração seja compatível com a versão anterior sendo apenas incrementos de software, onde não ocorra quebra de contrato entre os serviços.