# **Proposta arquitetura – Aplicação para registro e análise de horas trabalhadas**

Nesse documento será apresentado a proposta de arquitetura para desenvolvimento de uma aplicação que terá como responsabilidade o registro de ponto de entrada e saída dos funcionários cadastrados. Além de quando necessário a visualização dos dados relacionados a horas trabalhadas e intervalos realizados. Outra funcionalidade será a possibilidade de obter relatórios de meses passados com o espelho de ponto dos funcionários.

A partir da rápida descrição acima, identificamos três domínios:

- Registros de entrada/saída

- Visualizacao de dados

- Geracao de relatorios

**Proposta de arquitetura MVP**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Nessa primeira solução, iremos implementar um AWS Cognito para autenticação do usuário, onde apenas com essa autenticação será possível realizar requisições a aplicação.

Toda estrutura estará dentro de uma mesma VPC multi região. Com isso, quando o usuário realizar a requsicao http request, ela terá que passar pelo API gateway, que irá validar se o usuário foi autorizado no cógnito.

Caso sim, a requisição ser;a direcionada para uma aplicação python, um monolito que agrupa os três domínios descritos anteriormente. Para essa decisão foi considerado que existe apenas um desenvolvedor trabalhando, e precisa ser entregue o MVP em 5 dias. Além de atender os requisitos mínimos necessários, o ideal para um MVP.

Além disso, toda gestão de dados será feita dentro de um banco de dados SQL, onde teremos o seguinte schema:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Em relação a infraestrutura escolhida, para garantir desempenho, escalabilidade, disponibilidade da solução, iremos criar uma imagem Docker e deployar dentro de um AWS Fargate, onde automatiza algumas configurações, simplificando a implementação da imagem garantindo todos os pontos descritos anteriormente.

Em relação a segurança, como é possível acessar a api apenas pessoas aprovadas pelo cógnito, controlamos o acesso as informações. Além disso, via código validamos se a pessoa que está solicitando a informação é a pessoa a qual pertence os registros, ou é uma pessoa com um nível de autorização de acesso a essas informações independentes se é o dono dos dados ou não, por exemplo RH.

Ao usar o Dynamo DB como banco de dados SQL, garantimos a integridade dos dados, e como a princípio teremos apenas uma rota de alteração dos dados, é simples garantir o nível atômico das alterações.

Sendo um monolito, com apenas um BD, se torna simples a manutenção do mesmo.

Todos os dados utilizados são importantes e úteis para atender as funcionalidades da aplicação, em conformidade com a LGPD, onde também permitimos o acesso a todos os dados pertencentes aos usuários assim que requerido, e armazenamos apenas os necessários.

A resiliência é garantida por estarmos utilizando toda infraestrutura em multi zone e serveless, garantindo uma disponibilidade acima de 99% do tempo, de acordo com o descrito na documentação da AWS.

**Proposta de arquitetura fase 2**

Para a fase dois, pensando que é a fase subsequente, propomos o seguinte desenho de arquitetura:

Onde mantemos o monolito inicial, adicionando as rotas necessárias dentro dele. Mas adicionaremos novos componentes para atender algumas das demandas.

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

Primeiro, iremos relacionar a autenticação do usuário no notebook com a autenticação no cógnito. Com isso iremos obrigar o usuário estar conectado na internet para trabalhar (empresas como o Itaú e Mercado Livre já o fazem). Assim que for feito o login, também será autorizado no cógnito da AWS.

Com esse processo, será gerado um evento no SNS, o qual será responsável de enviar um e-mail e um sms ao empregado avisando sobre a batida do ponto.

Além dessa conexão, será necessário a criação de uma API utilizando o streamlit para gerar uma interface de gerenciamento.

A escolha desse tipo de solução é por ser uma solução python, mantendo uma única linguagem de solução para todo o processo, facilitando a manutenção. Além disso, podemos subir ele em um ECS, dentro da VPC, garantindo a segurança de acesso aos dados apenas para as pessoas que já foram autorizadas no cógnito, uma vez que a rota só é acessível via gateway.

A edição de novos tipos de relatórios e rotas para solicitação de ajustes será feita no código do monolito com falado anteriormente.

Ou seja, para sair do MVP para a fase dois, será necessário a execução de tarefas para adicionar os componentes e correlacioná-los, criação da API do streamlit e adição das novas rotas. Sem nenhuma necessidade de alteração de infra, separação em micro serviço ou qualquer outro trabalho/retrabalho sobre o MVP.

Em fases futuras talvez faca sentido, mas não agora.

O schema do banco de dados seguirá o mesmo.