RELATÓRIO DA POC (Proof of Concept)

Neste documento, dispomos de informações acerca da Prova de Conceito e todo seu processo de desenvolvimento, onde declaramos as etapas pelas quais passamos para desenvolver o front-end e back-end, o banco de dados do projeto, como foi a integração entre todas as partes, a integração com a plataforma AWS, as dificuldades enfrentadas e as tarefas das quais cada um se encarregou.

Banco de Dados - MySQL
Responsáveis: Beatriz H., Gabriel P., Camila F.
O primeiro passo da criação do banco foi a modelagem MER e DER, feita pelo grupo como uma forma de mapear as necessidades do banco. Após três dias de discussões, foi alcançado um modelo prévio, com 13 tabelas. Em seguida, a implementação deste no MySQL, acarretando em mudanças na regra de negócio de bancos pré-existentes. Com a implementação feita, toda a parte de banco foi considerada completa. Porém, novas revisões com histórias de usuário e casos de uso causaram a remoção de 2 tabelas, consideradas supérfluas ao banco, o que resultou em um modelo final de 11 tabelas. MER e DER foram ajustados, e banco corrigido nos conformes.

Back-end - Python, framework Django
Responsáveis: Lucas F., Camila F., Gabriel P.
O back-end da aplicação é o responsável pela implementação da regra de negócio. Em uma aplicação web ele não toca na parte visual da aplicação e sim, visa manter os links internos funcionando e suas devidas conexões estabelecidas. Para o nosso projeto, utilizamos a linguagem Python juntamente com o framework Django para integrar nosso sistema com o banco de dados MySQL (default para o sistema). Caso o cliente possua o dele, será feita uma cobrança extra para desenvolvimento de script de importação do mesmo. O projeto consiste na seguinte MTV (Model, Template e View): ● Templates: é o diretório responsável por armazenar as páginas HTML do projeto; ● Models.py: é o arquivo que define quais atributos o projeto irá possuir; ● Views.py: são as regras de negócio da aplicação; Também temos: ● Forms.py: é o arquivo onde é definido os campos dos formulários; ● Urls.py: são as rotas para acessar os recursos do projeto.

Front-end - CSS, framework Bootstrap
Responsáveis: Bruna R., Gabriel R.
Para o front-end, foram trabalhadas páginas HTML estilizadas com CSS através do framework Bootstrap. Foram criadas as páginas de cadastro dos usuários (da empresa do coworking e seu administrador, e do cliente final) e sua tela de login. Este ponto da aplicação faz interação direta com o usuário, tendo ligação de extrema importância com o back-end, pois recebe os dados que o usuário insere e conecta com a View, que faz o Post no banco de dados. Como o foco da POC é a demonstração da arquitetura, questões gráficas como paleta de cores e imagens foram ignoradas em favor de funcionalidade, e serão implementadas no MVP.

Host - AWS
Responsável: Fernando S.
Para a arquitetura do projeto, foi criado um repositório do github que é utilizado na etapa de source do serviço AWS CodePipeline criado para tratar do processo de integração contínua e que envia os arquivos para um bucket no AWS S3. Para a etapa de build foi necessário criar uma instância EC2 dedicada ao Jenkins e tentamos utilizar o serviço AWS codedeploy para esta etapa de deploy, na qual, iria mandar o arquivo da aplicação .zip para uma instância EC2 da aplicação se tivesse funcionado. E por fim, foi utilizado o serviço AWS RDS para gerenciar o banco de dados MySQL, na qual foi integrado com a instância EC2 utilizada para disponibilizar a aplicação web O seguinte erro surgiu durante a implantação do codedeploy: *The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems.* Entretanto, a instância da aplicação está vinculada ao grupo de implantação e a função de serviço foi configurada para gerar as permissões: **AmazonSSMFullAccess** **AWSCodeDeployRole** **AmazonSSMManagedInstanceCore**

AWSCodeDeployFullAccess
AmazonS3FullAccess
AmazonEC2RoleforAWSCodeDeploy
AmazonEC2RoleforSSM

O codedeploy-agent.sh foi instalado e está funcionando na instância EC2.

Ao excluir o grupo de implantação e criar outro, surge o seguinte erro:

"Houve um problema ao estabelecer a configuração do AWS Systems Manager: User: arn:aws:sts::998708728874:assumed-role/vocstartsoft/user1485930=fernando.sousa@aluno.ifsp.edu.br is not authorized to perform: ssm:CreateAssociation on resource: arn:aws:ssm:us-east-1::document/AWS-ConfigureAWSPackage with an explicit deny esta é uma etapa essencial para instalar o agente CodeDeploy em suas instâncias. Vá para o AWS System Manager para definir a configuração."

Mas, ao tentar utilizar o serviço AWS Systems Manager, surge o erro:

User: arn:aws:sts::998708728874:assumed-role/vocstartsoft/user1485930=fernando.sousa@aluno.ifsp.edu.br is not authorized to perform: cloudformation:ListStacks on resource: arn:aws:cloudformation:ap-southeast-2:998708728874:stack// with an explicit deny

Integração dos sistemas

Responsáveis: Gabriel P., Bruna R., Lucas F., Beatriz H.

A integração entre front, back e banco foi uma das últimas etapas do desenvolvimento. Primeiro, foi feita a conexão do banco de dados com o back, garantindo que qualquer comando enviado fosse recebido pela base de dados. Tal conexão foi feita no arquivo settings.py, pasta NewGen, e completa com sucesso.

Em seguida, a integração de front com back, que envolveu mover os arquivos de HTML e configurações para os templates, na pasta NewGenApp, também feita com sucesso.

Por fim, foi feito o teste de inserção, que envolvia preencher o formulário e verificar se os dados inseridos apareciam na base de dados. Inicialmente, não houve tal transferência, e investigações posteriores identificaram que a tag de finalização do form estava em posição incorreta, que não permitia a validação do POST pelo Django.

Após sua correção, a integração funcionou corretamente, com todas as informações sendo inseridas no banco de dados.

Dificuldades e Mudanças de regras

Umas das maiores dificuldades que enfrentamos durante o desenvolvimento da POC foi com as conexões relacionadas à AWS. Como especificado anteriormente, o maior impasse ocorreu na etapa de utilização do serviço da própria plataforma de nuvem, denominada AWS CodeDeploy, fundamental para a disponibilidade de nossa aplicação em um servidor de nuvem.

No desenvolvimento do banco de dados, tivemos um problema ao excluir uma tabela que verificamos não ser importante, pois utilizamos variáveis da mesma como chaves estrangeiras em outras tabelas, e ao tentar excluir a tabela, não era possível por conta dos índices correlacionados às chaves estrangeiras.

Outra dificuldade foi a integração front-back, para recebimento de dados do front e armazenamento destes no banco. Após a depuração de código e alterações do back-end, foi descoberta a falha no front-end.

Por questões do AWS, não foi possível verificar a questão do domínio, e por questões pessoais da equipe, foi decidido não fazer o uso de um domínio pago.

E por conta desses atrasos, não houve tempo para implementação da internacionalização do projeto.

Entre as mudanças de regras definidas, a questão de banco de dados foi re-estudada pela equipe; em casos de clientes com bancos de dados pré-existentes, a possibilidade de tal banco possuir um formato similar ao nosso seria baixa. Portanto, torna-se viável manter o banco para todas as iterações como MySQL, e em casos de bancos pré-existentes ao cliente, será dada uma opção de importar os dados, sob o pagamento de uma taxa, ou ignorar este banco. Assim, a aplicação não necessita de ORMs, facilitando sua manutenção.

Também está sendo estudada a mudança de AWS para Heroku, por conta dos problemas encontrados na implementação com o AWS, e também das apresentações de outras equipes, que tiveram mais facilidade com a outra plataforma de nuvem.