**RELATÓRIO FASE 2**

**Setup de Infraestrutura e Ambientes**

1. **Objetivo da Fase**

O objetivo desta fase é preparar o ambiente de desenvolvimento e produção para garantir que a infraestrutura esteja pronta para suportar o desenvolvimento contínuo da aplicação. Vamos configurar a infraestrutura necessária, incluindo a conteinerização com Docker, pipelines de CI/CD, e o provisionamento do banco de dados SQL Server.

1. **Atividades Principais**
   1. **Configuração de Docker e Docker Compose**

* Objetivo: Conteinerizar a aplicação e os serviços dependentes (back-end, front-end e banco de dados) para garantir portabilidade e consistência no ambiente de desenvolvimento e produção.
* Passos:
  + Definir o Dockerfile para o back-end (ASP.NET Core) e front-end (Angular ou React):
    - O Dockerfile deve conter as instruções para a criação da imagem da aplicação.
    - O container do SQL Server será configurado também, usando a imagem oficial da Microsoft.
  + Criar um arquivo docker-compose.yml para orquestrar os containers (back-end, front-end e banco de dados):
    - O docker-compose permitirá levantar o ambiente completo com um único comando.
    - Verifique a persistência de dados no SQL Server para garantir que os dados não sejam perdidos ao reiniciar os containers.
* Entregáveis:
  + Dockerfile para o back-end e front-end.
  + Arquivo docker-compose.yml funcional para levantar os serviços locais.
  1. **Configuração de CI/CD**
* Objetivo: Implementar uma pipeline de integração e entrega contínua (CI/CD) para automação do build, testes, e deploy da aplicação.
* Passos:
  + Configurar Azure DevOps ou GitHub Actions:
    - Criação de pipelines automáticas para o projeto.
  + Definir stages para:
    - Build: Construção da aplicação em ASP.NET Core e front-end.
    - Testes: Execução de testes unitários e de integração.
    - Deploy: Deploy automático no ambiente de desenvolvimento e, posteriormente, no ambiente de produção.
  + Definir Gatilhos Automáticos:
    - A pipeline deve ser acionada automaticamente para cada push/merge no branch principal.
    - Configurar integração com Docker para criar imagens e fazer o deploy no ambiente.
  + Monitoramento e Logs:
    - Configurar logs de erros e monitoramento básico no ambiente de produção.
* Entregáveis:
* Pipelines CI/CD configuradas para build, testes e deploy contínuo.
* Deploy automático no ambiente de desenvolvimento com logs e monitoramento básico.
  1. **Provisionamento do Banco de Dados SQL Server**
* Objetivo: Configurar e provisionar o banco de dados SQL Server que será utilizado pela aplicação para armazenar todas as informações.
* Passos:
  + Provisionar um container Docker com SQL Server no ambiente de desenvolvimento:
    - Configurar as permissões e o schema inicial.
    - Criar tabelas básicas como: Usuários, Despesas, Aprovações, Logs.
  + Configuração do Banco no Azure (ou VPS da Hostinger como backup):
    - Caso o ambiente seja no Azure, configurar o Azure SQL Database.
    - Criar backups automáticos e configurar replicação, se necessário.
  + Script de Seed:
    - Criar scripts de seed para popular o banco com dados iniciais (ex: tipos de categorias de despesas, moedas padrão).
* Entregáveis:
  + Banco de dados SQL Server funcional no ambiente de desenvolvimento.
  + Scripts para criação do schema e seed de dados.
  + Backup e configuração do banco no ambiente de produção (Azure ou VPS).
  1. **Configuração de Infraestrutura no Azure**
* Objetivo: Configurar o ambiente de produção e desenvolvimento no Azure para garantir que a aplicação esteja pronta para deploy e possa escalar conforme necessário.
* Passos:
  + Criar e configurar o Azure App Service:
    - Configurar para hospedar a API e o front-end da aplicação.
    - Definir limites de escalabilidade para lidar com o tráfego crescente.
  + Configurar o Azure Blob Storage:
    - Será usado para armazenar os recibos de despesas que forem submetidos.
    - Configurar a persistência e acessos seguros aos arquivos.
  + Configurar Monitoramento e Alertas:
    - Utilizar Azure Monitor para configurar logs de erros, rastreamento de desempenho, e alertas automáticos em caso de falhas.
* Entregáveis:
  + Azure App Service configurado e pronto para deploy.
  + Azure Blob Storage configurado para armazenar arquivos de recibos.
  + Monitoramento e alertas básicos configurados no Azure Monitor.

1. **Gestão de Riscos**
   1. **Falhas no Docker Compose**

* Risco: Problemas de configuração no Docker Compose podem atrasar o desenvolvimento local.
* Mitigação: Garantir que todas as dependências estejam documentadas e testadas antecipadamente. Fazer testes locais antes de integração com o CI/CD.
  1. **Problemas com CI/CD**
* Risco: O CI/CD pode apresentar falhas ou ser complexo para configurar corretamente, resultando em deploys incorretos.
* Mitigação: Configurar uma pipeline básica de CI/CD e evoluir gradualmente, realizando testes automatizados em cada passo.
  1. **Custos no Azure**
* Risco: O uso de serviços no Azure pode resultar em custos elevados, especialmente em ambientes de produção.
* Mitigação: Monitorar os custos de uso no Azure e, se necessário, migrar para a Hostinger com configuração manual de containers.

1. **Conclusão**

Ao final da Fase 2, a infraestrutura da aplicação estará configurada e pronta para o desenvolvimento contínuo. Isso inclui a containerização com Docker, pipelines CI/CD funcionando corretamente, e o banco de dados SQL Server configurado e provisionado. Com isso, será possível iniciar o desenvolvimento das funcionalidades da aplicação nas próximas fases.