Briefing da API

1- Visão Geral do Projeto

Nome do Projeto: API para dashboard de indicadores

Objetivo: Criar um método de requisição de dados para mostrar informações como, transações, usuários e cidades impactadas sobre empresas e lojas cadastradas no aplicativo Helpnei

2- Requisitos funcionais e não funcionais do Projeto

O projeto deve extrair e exibir os seguintes dados:

- Número total de lojas criadas (RF-02)
- Quantidade de usuários impactados (RF-03)
- Volume de transações realizadas (RF-04)
- Distribuição geográfica das lojas e usuários (RF-05)
- Métricas de engajamento e uso da plataforma (RF-06)

3- Requisitos Técnicos

Tecnologias:

- **Backend**: Node.js(ou outra stack definida)
- **Banco de Dados**: MySQL ou PostgreSQL
- **Padrão de Respostas**: JSON

4- Critérios para o sucesso

- A API deve ser segura, garantindo autenticação de usuários quando necessário.
- O tempo de resposta dos endpoints deve ser menor que 500ms.
- Deve ser possível filtrar os dados por período e localização.
- A API precisa estar documentada.
- O sistema deve suportar múltiplas requisições simultâneas sem queda de performance.

5- Restrições e Premissas

- O sistema deve seguir boas práticas RESTful para garantir integração fácil com o front-end.
- A API deve ser escalável para futuras melhorias e integrações.

Definition of Ready(DoR)

1. Artefatos Necessários

- **Briefing Estruturado** – O contexto e os objetivos da API devem estar documentados.
- **User Story detalhada** – A funcionalidade deve estar descrita no formato correto: **Como [usuário], quero [ação] para que [benefício]."**
- **Critérios de Aceitação** – Definição clara do que deve ser entregue para que a User Story seja aceita.
- **Wireframes e Protótipos** – Se a funcionalidade envolver interação visual, os layouts devem estar definidos.
- **Modelos de Dados e Esquema do Banco** – As tabelas, relacionamentos e estrutura do banco de dados devem estar definidos e aprovados.
- **Regras de Negócio Documentadas** – Todas as regras de operação da funcionalidade devem estar descritas.

2. Regras Técnicas Definidas

- **Endpoint bem especificado**:
  - **URL RESTful** definida (ex: GET /usuarios/{id}).
  - **Método HTTP correto** (GET, POST, PUT, DELETE).
  - **Formato da requisição e resposta JSON** especificado.
- **Tratamento de Erros Definido** – Respostas esperadas para erros (400 Bad Request, 404 Not Found, etc.).
- **Controle de Versionamento** – Definir versões da API (exemplo: v1/usuarios).
- **Performance e Eficiência** – Definir limites de requisição e evitar sobrecarga de dados.

3. Testes e Validação

- **Cenários de Teste Criados** – Casos normais e de exceção para validar a funcionalidade.
- **Dados para Testes Disponíveis** – Exemplo: usuários de teste cadastrados no banco de dados.
- **Plano de Testes de Integração** – Garantir que os endpoints funcionam bem com outros sistemas.

4. Aprovação e Alinhamento

- A equipe de desenvolvimento e o **Product Owner** revisaram e concordam que a funcionalidade pode ser desenvolvida.
- O item foi incluído no **Sprint Backlog** e tem **estimativa de esforço definida**.
- O desenvolvedor responsável compreendeu o escopo da tarefa e tirou dúvidas antes de iniciar.