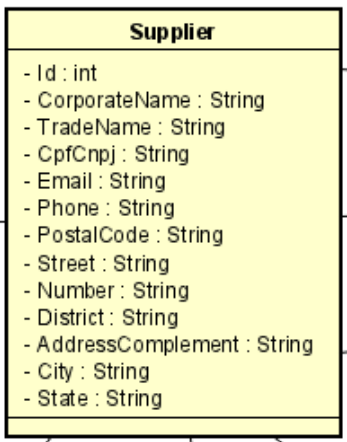


|  |  |  |
| --- | --- | --- |
| **Sprint Documentation** | | |
| **Sprint #02 Desenvolver API Carrier** | **Start Date:**01/10/2024 | **Final Date:** 14/10/2024 | |
| **Team Members** | | | |
| Maysa Lopes de Sousa | | | |
| Izabelly Cristina Silva Brito | | | |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint Backlog** | | | |
| Task# | Description | Start Date | Final date |
| 037 | Sprint 02 - Desenvolver a API para Supplier | 01/10/2024 | 07/10/2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task Description** | | | | | |
| Task # | Description | Assigned To | Status | Estimated Hours | Logged Hours |
| 037 | Desenvolvido a API Supplier “Fornecedor” com (Estruturação do Banco de Dados, Operações Crud e Validação dos Dados.) | Maysa Lopes de Sousa e Izabelly Cristina Silva Brito | Concluído | 45h | 30h |
|  |  |  |  |  |  |

|  |
| --- |
| **Sprint Results** |



Esse diagrama de classes foi fundamental para guiar o desenvolvimento da nossa API. Ele ofereceu uma visão clara de como a estrutura dos dados deveria ser organizada no banco de dados, o que facilitou bastante a implementação das operações de CRUD.

Com essa representação visual, ficou mais simples compreender como cada informação da transportadora seria armazenada e recuperada. Por exemplo, campos como "CorporateName", "TradeName" e "CpfCNPJ" precisavam ser do tipo string para comportar dados textuais, enquanto o "Id", sendo um número inteiro, funcionaria como chave primária.

Além disso, o diagrama contribuiu para manter o código organizado e consistente. A partir dessa estrutura, conseguimos planejar cada parte da API de forma clara, evitando confusões e garantindo que todas as funcionalidades fossem bem implementadas. Ele também facilitou os testes, pois sabíamos exatamente quais dados estávamos manipulando e como deveriam ser validados. Em suma, o diagrama foi um recurso essencial para assegurar o alinhamento entre o código e o banco de dados.

**Telas Produzidas**

O que implementamos na API

Durante esta sprint, trabalhamos no desenvolvimento de uma API para gerenciar o cadastro de fornecedores (Supplier). O objetivo principal foi implementar funcionalidades que permitam cadastrar, atualizar, excluir e listar fornecedores, utilizando os métodos HTTP (GET, GET por ID, POST, PUT, DELETE).

A API foi desenvolvida com foco em garantir que o sistema de cadastro de fornecedores seja eficiente, com validação adequada dos dados e uma estrutura flexível para futuras modificações.

**1. GET /suppliers**

Esse endpoint retorna todos os fornecedores cadastrados no sistema. Configuramos a resposta para que todos os dados dos fornecedores sejam exibidos de forma clara, facilitando a consulta e a visualização das informações.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**2. GET /suppliers/{id}**

Aqui, implementamos uma busca específica por ID, que retorna os dados de um fornecedor específico. Esse recurso é importante quando é necessário obter informações detalhadas sobre um fornecedor sem a necessidade de percorrer toda a lista.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**3. POST /suppliers**

Esse endpoint foi criado para permitir o cadastro de um novo fornecedor no sistema. Todos os campos obrigatórios são validados antes de permitir que o cadastro seja concluído, garantindo que nenhuma informação essencial fique de fora.

Interface gráfica do usuário

Descrição gerada automaticamente

4. PUT /suppliers/{id}

No endpoint de atualização, desenvolvemos a lógica para modificar as informações de um fornecedor. Qualquer dado relacionado ao fornecedor pode ser atualizado, e o sistema retorna uma mensagem confirmando que as alterações foram realizadas com sucesso.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

5. DELETE /suppliers/{id}

Por fim, implementamos a funcionalidade que permite excluir um fornecedor do sistema, com base no ID fornecido. Isso garante que fornecedores que não estão mais em operação ou que não são mais relevantes para o sistema possam ser removidos.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Estrutura do Cadastro de Transportadoras**

Cada transportadora deve ser registrada com as seguintes informações no banco de dados, com todos os tipos e validações necessários:

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descrição |
| **ID** | **Int** | Identifica a chave primária. |
| **CorporateName** | **String** | Razão social da empresa. (Varchar(100)) |
| **TradeName** | **String** | Nome comercial. (Varchar(100)) |
| **CpfCnpj** | **String** | Número de inscrição na Receita Federal (CPF/CNPJ). (Varchar(14)) |
| **Email** | **String** | Endereço de e-mail. (Varchar(50)) |
| **Phone** | **String** | Número de telefone. (Varchar(11)) |
| **PostalCode** | **String** | CEP. (Varchar(8)) |
| **Street** | **String** | Nome da rua. (Varchar(100)) |
| **Number** | **String** | Número do endereço. (Varchar(5)) |
| **District** | **String** | Bairro. (Varchar(50)) |
| **AddressComplement** | **String** | Complemento do endereço. (Varchar(100)) |
| **City** | **String** | Cidade. (Varchar(100)) |
| **State** | **String** | Estado. (Varchar(2)) |

**Funcionalidades Não Implementadas**

Autenticação e Autorização de Usuários

O foco principal desta sprint foi a construção da estrutura essencial da API e a implementação das operações CRUD (Create, Read, Update, Delete). A inclusão de autenticação e autorização adicionaria uma camada extra de complexidade, exigindo o gerenciamento de permissões e perfis de acesso. Decidimos concentrar os esforços na conclusão das funcionalidades centrais da API, deixando a implementação de segurança para fases futuras, uma vez que o objetivo era garantir o funcionamento básico antes de avançar para recursos mais sofisticados.

Validação Avançada de Dados (CPF/CNPJ, E-mail, Telefone)

Optamos por realizar uma validação básica dos dados fornecidos, como limitar os tamanhos dos campos (varchar), para garantir um nível mínimo de integridade. No entanto, validações mais rigorosas, como a verificação dos formatos corretos de CPF, CNPJ, e-mail e telefone, ainda não foram implementadas. Essas verificações avançadas exigiriam o uso de expressões regulares e regras específicas, o que demandaria mais tempo e testes. Decidimos que, para esta sprint, uma validação básica seria suficiente para atender aos requisitos iniciais, com a intenção de aprimorar essa parte nas próximas fases.

**Aprendizados**

Durante esta sprint, conseguimos concluir a API dentro do prazo estipulado, garantindo que todas as funcionalidades essenciais fossem entregues. Além de desenvolver os endpoints, realizamos validações básicas no banco de dados, assegurando o correto funcionamento da API. Todos os testes foram bem-sucedidos, e a API está pronta para ser integrada com outros módulos do sistema.

Essa sprint nos proporcionou importantes aprendizados. A capacidade de definir prioridades foi fundamental para que nos concentrássemos nas funcionalidades mais críticas, garantindo a entrega de uma API funcional dentro do prazo. A comunicação constante e a colaboração eficiente da equipe foram determinantes para a rápida solução de problemas. Os testes unitários ressaltaram a importância de verificar cada funcionalidade, mostrando que devemos investir mais em testes automatizados nas próximas fases. Manter o código organizado e bem documentado também foi crucial, facilitando o trabalho em equipe e garantindo clareza para futuras manutenções. Esses aprendizados serão aplicados em sprints futuras, contribuindo para a melhoria contínua da qualidade e do desempenho do projeto.