Design e Modelagem

1. Visão Geral da Arquitetura

Padrão Arquitetural:

Layered Architecture (Arquitetura em Camadas)

Justificativa:

A arquitetura em camadas foi escolhida por permitir uma clara separação de responsabilidades entre as partes do sistema, facilitando o entendimento, a manutenção e a evolução da aplicação. Esse padrão organiza o sistema em camadas bem definidas, como apresentação, regras de negócio e acesso a dados. Além disso, a Arquitetura em Camadas está alinhada com a arquitetura adotada pelo framework Django, utilizado no desenvolvimento do projeto.

Linguagem & Framework:

O sistema será desenvolvido utilizando Python com o framework Django REST Framework, responsável pela organização das camadas da aplicação.

Banco de Dados:

Será utilizado o SQLite, um banco de dados relacional, adequado ao escopo do projeto e compatível com o Django.

Segurança:

O acesso ao sistema será restrito a usuários internos da empresa, por meio de autenticação via API. Não haverá integração com APIs externas neste projeto.

2. Diagramas

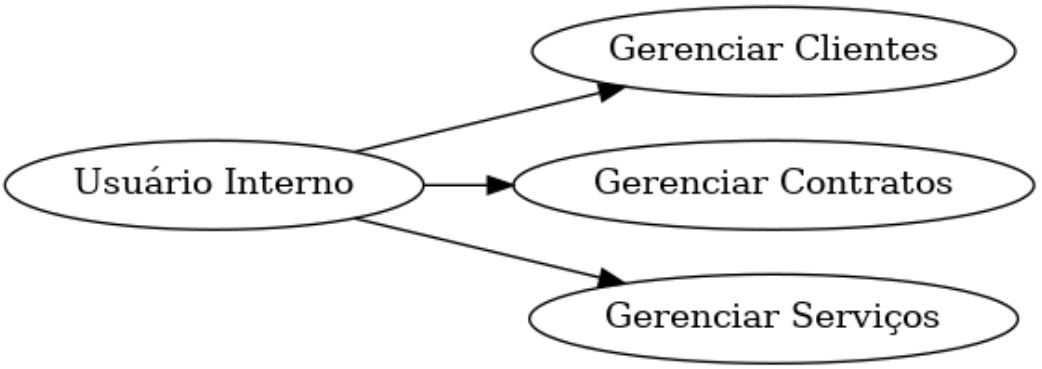
2.1 Protótipo

Foi elaborado um protótipo conceitual de um painel administrativo interno utilizando a ferramenta Canva, com o objetivo de representar visualmente o fluxo de uso do sistema. O protótipo possui caráter ilustrativo e não representa uma interface gráfica implementada, visto que o projeto tem foco na validação via API.

Link protótipo:https://www.canva.com/design/DAG8AQcLbn4/08oT_0pav4dO9SiHcJdV9Q/edit?utm_content=DAG8AQcLbn4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

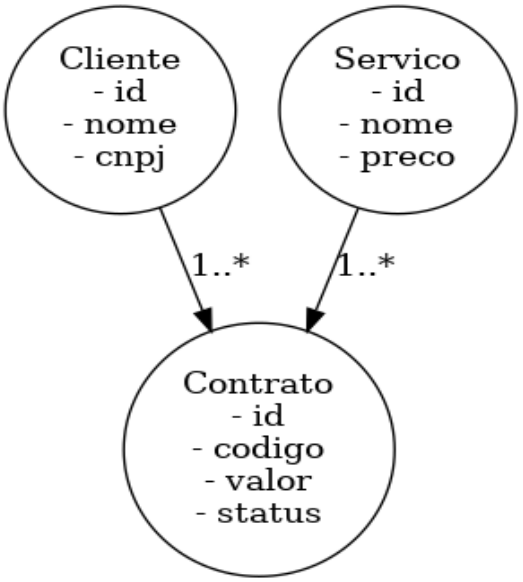
2.2 Diagrama de Casos de Uso

O Diagrama de Casos de Uso representa as principais funcionalidades disponíveis aos usuários internos da empresa, responsáveis pelo gerenciamento de clientes, contratos, serviços e colaboradores. Esse diagrama tem como objetivo demonstrar, de forma simplificada, as interações dos usuários com o sistema.



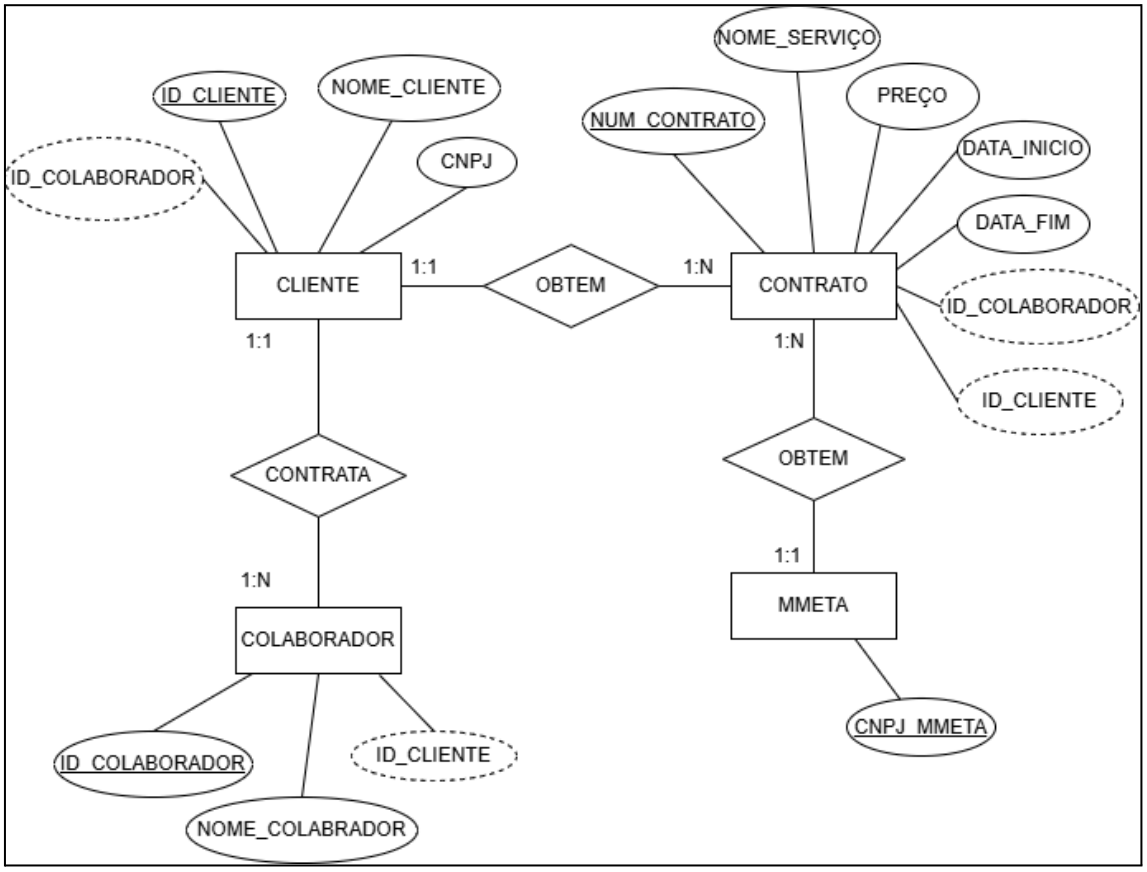
2.3 Diagrama de Classes

O Diagrama de Classes apresenta as principais classes do sistema, seus atributos e relacionamentos, refletindo a estrutura necessária para o funcionamento da aplicação. Ele serve como base para a implementação das entidades no código, representando a organização lógica do sistema.



2.4 Diagrama Entidade-Relacionamento (ER)

O Diagrama Entidade-Relacionamento representa a estrutura do banco de dados relacional utilizado no projeto, garantindo integridade e o correto relacionamento entre as entidades.



Entidades

CLIENTE

- ID (PK)
- Nome
- CNPJ

SERVIÇO

- ID (PK)
- Nome
- Sigla
- Preco_Base

CONTRATO

- ID (PK)
- Codigo
- Valor_Negociado
- Data_de_Inicio
- Data_de_Termino
- Status
- Cliente_ID (FK)
- Servico_ID (FK)

COLABORADOR

- ID (PK)
- Nome
- Cliente_ID (FK)

Relacionamentos

- Cliente **1:N** Contrato
- Serviço **1:N** Contrato
- Cliente **1:N** Colaborador

3. Design da API (Contrato de Interface)

Padrões Globais

- **Formato de dados:** JSON (application/json)
- **Base da API:** /api/v1
- **Autenticação:** obrigatória para todos os endpoints
- **Banco de dados:** SQLite
- **Tratamento de Erros:** Todas as respostas de erro da API seguirão um padrão único em formato JSON, com o objetivo de facilitar a identificação e o tratamento de falhas por parte do consumidor da API. As mensagens de erro conterão informações claras sobre o tipo de problema ocorrido.

```
{
 "timestamp": "2025-01-01T10:00:00",
 "status": 400,
 "error": "Bad Request",
 "message": "Erro de validação nos dados informados"
 }
```

Código	Descrição	Quando ocorre
400	Bad Request	Dados inválidos ou incompletos
401	Unauthorized	Requisição sem autenticação
403	Forbidden	Usuário sem permissão
404	Not Found	Recurso não encontrado
409	Conflict	Violação de regra de negócio
500	Internal Server Error	Erro interno da aplicação

3.1 Recursos da API

Recurso: Clientes (/clients)

Método	Rota	Aplicação	Autenticação
GET	/api/clients	Lista clientes cadastrados	Sim
POST	/api/clients	Cadastra novo cliente	Sim
PUT	/api/clients/{id}	Atualiza dados do cliente	Sim
DELETE	/api/clients/{id}	Exclui cliente	Sim

Recurso: Serviços (/services)

Método	Rota	Aplicação	Autenticação
GET	/api/v1/services	Lista serviços cadastrados	Sim
POST	/api/v1/services	Cadastra novo serviço	Sim
PUT	/api/v1/services/{id}	Atualiza dados do serviço	Sim
DELETE	/api/v1/services/{id}	Exclui serviço	Sim

Recurso: Contratos (/contracts)

Método	Rota	Descrição	Autenticação
GET	/api/v1/contracts	Lista contratos cadastrados	Sim
GET	/api/v1/contracts/{id}	Consulta contrato por ID	Sim
POST	/api/v1/contracts	Cria novo contrato	Sim
PUT	/api/v1/contracts/{id}	Atualiza dados do contrato	Sim

3.2 Detalhe de Endpoint

Criar Contrato – POST /api/v1/contracts

Request Body (Entrada)

```
{
  "cliente_id": 1,
  "servico_id": 2,
  "valor_negociado": 1500.00,
 "data_inicio": "2025-01-01",
 "data_fim": "2025-12-31",
 "observacoes": "Contrato anual"
}
```

Response (Sucesso – 201 Created)

```
{
  "id": 10,
  "codigo": "CTR-2025-001",
  "status": "VIGENTE"
}
```

Códigos de Status Possíveis

Código	Descrição
201	Created – Contrato criado com sucesso
400	Bad Request – Erro de validação
401	Unauthorized – Usuário não autenticado
404	Not Found – Cliente ou serviço inexistente
409	Conflict – Violação de regra de negócio
500	Internal Server Error – Erro interno

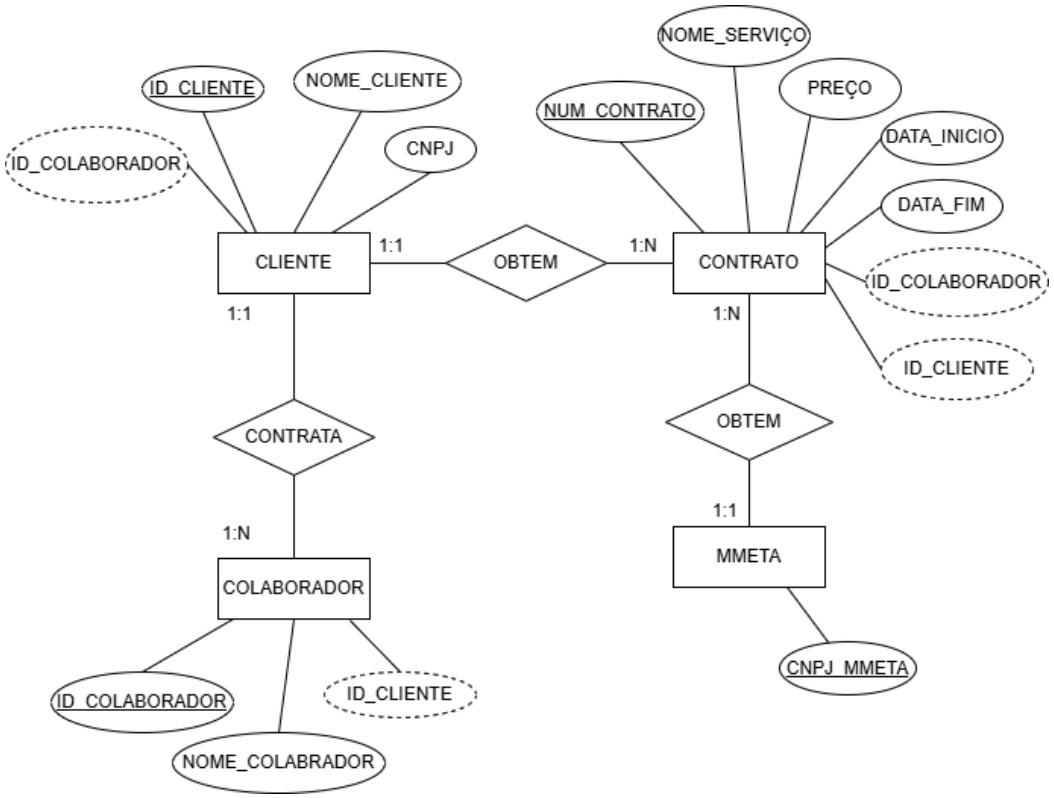
4. Modelagem de Dados (Schema)

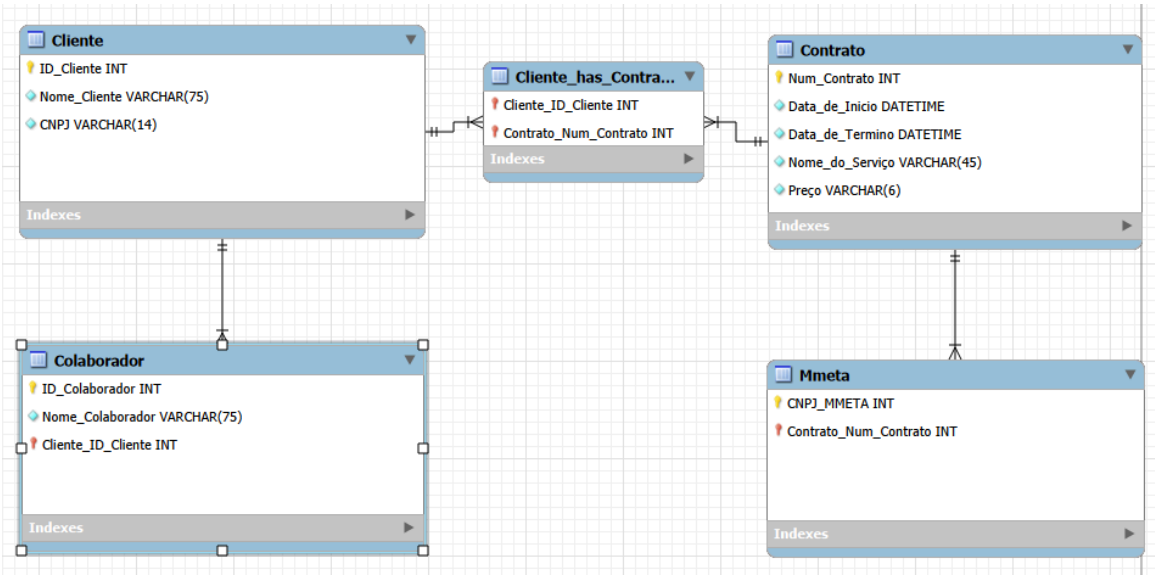
4.1 Visão Geral da Modelagem

A modelagem de dados foi definida com base nos requisitos do sistema de gestão de contratos, visando garantir a integridade, organização e consistência das informações armazenadas. O modelo adotado é relacional, compatível com o banco de dados SQLite, e contempla as principais entidades necessárias para o funcionamento da aplicação.

4.2 Diagrama Entidade-Relacionamento (ER)

O Diagrama Entidade-Relacionamento representa a estrutura do banco de dados do sistema, evidenciando as entidades Cliente, Serviço, Contrato e Colaborador, bem como seus respectivos relacionamentos. Esse diagrama serve como base para a criação do banco de dados e para a implementação das regras de negócio da aplicação.





4.3 Dicionário de Dados

Tabela: Cliente

Campo	Tipo	Restrições	Descrição
ID	INTEGER	PK	Identificador único do cliente
Nome	TEXT	NOT NULL	Nome da empresa
CNPJ	TEXT	UNIQUE	CNPJ da empresa

Tabela: Serviço

Campo	Tipo	Restrições	Descrição
ID	INTEGER	PK	Identificador único do serviço
Nome	TEXT	NOT NULL	Nome do serviço
Sigla	TEXT	UNIQUE	Sigla do serviço
Preco_Base	DECIMAL	NOT NULL	Preço base do serviço

Tabela: Contrato

Campo	Tipo	Restrições	Descrição
ID	INTEGER	PK	Identificador único do contrato
Codigo	TEXT	UNIQUE	Código do contrato
Valor_Negociado	DECIMAL	NOT NULL	Valor negociado
Data_de_Inicio	DATE	NOT NULL	Data de início do contrato
Data_de_Termino	DATE	NOT NULL	Data de término do contrato
Status	TEXT	NOT NULL	Status do contrato
Cliente_ID	INTEGER	FK	Referência ao cliente
Servico_ID	INTEGER	FK	Referência ao serviço

Tabela: Colaborador

Campo	Tipo	Restrições	Descrição
ID	INTEGER	PK	Identificador único do colaborador
Nome	TEXT	NOT NULL	Nome do colaborador
Cliente_ID	INTEGER	FK	Cliente ao qual o colaborador está vinculado

4.4 Considerações sobre a Modelagem

A modelagem proposta atende às necessidades do sistema, garantindo relacionamentos claros entre as entidades e evitando redundância de dados. O uso de chaves primárias e estrangeiras assegura a integridade referencial, enquanto o modelo relacional facilita consultas e manutenções futuras.

5. Plano de Testes

5.1 Objetivo

Verificar se as principais funcionalidades da gestão de contratos estão funcionando corretamente, garantindo a integridade dos dados e o correto tratamento das regras de negócio.

5.2 Escopo de Testes

- Cadastro de clientes
- Cadastro de Serviços
- Criação e atualização de contratos
- Validação de dados de entrada
- Tratamento de erros de API

5.3 Tipo de Testes

- Testes Funcionais:
  Verificar se os endpoints da API respondem corretamente às requisições.
- Testes de Validação:
  Garantir que os dados obrigatórios sejam informados e respeitem as regras definidas.
- Testes de Erro:
  Verificar se a API retorna mensagens de erro padronizadas em situações inválidas.

5.4 Ambiente de Testes

Item	Descrição
Linguagem	Python
Framework	Django
Banco de Dados	SQLite
Ferramenta de Teste	Postman

5.5 Casos de Teste

Caso de Teste 01 – Cadastro de Cliente

Campo	Descrição
ID	CT-01
Funcionalidade	Cadastro de cliente
Endpoint	POST /api/v1/clients
Entrada	Nome e CNPJ válidos
Resultado Esperado	Cliente cadastrado com sucesso (201 Created)

Caso de Teste 02 – Erro de Validação

Campo	Descrição
ID	CT-02
Funcionalidade	Cadastro de cliente
Endpoint	POST /api/v1/clients
Entrada	CNPJ inválido
Resultado Esperado	Erro 400 – Bad Request

Caso de Teste 03 – Criação de Contrato

Campo	Descrição
ID	CT-03
Funcionalidade	Criação de contrato
Endpoint	POST /api/v1/contracts
Entrada	Dados válidos
Resultado Esperado	Contrato criado com sucesso (201 Created)

5.6 Critério de sucesso

Os testes serão considerados bem-sucedidos quando pelo menos 70% das funcionalidades críticas estiverem funcionando conforme o esperado.