



CÂMPUS BAURU

COMPUTAÇÃO

AFONSO¹, Cibele L. da S. H.; YOSHIDA², Diego K.; SANTOS³, Eduardo V.; DUARTE⁴, Flavio C.; ZANATA⁵, Gabriel J. S.; PEREIRA⁶, Leonardo L.; PINHEIRO⁷, Thomas H. S.; SANTOS⁸, Vinicius L.

**DESENVOLVIMENTO E PROJETO DE SISTEMAS:
sistema de gerenciamento de alunos e cursos**

Bauru, São Paulo

2025

¹ 0.24100445ead@alunos.unisagrado.edu.br

² diego.24100039ead@alunos.unisagrado.edu.br

³ eduardo.24100480ead@alunos.unisagrado.edu.br

⁴ flavio.24100562ead@alunos.unisagrado.edu.br

⁵ gabriel.23100426ead@alunos.unisagrado.edu.br

⁶ leonardo.24100327ead@alunos.unisagrado.edu.br

⁷ thomas.24100016ead@alunos.unisagrado.edu.br

⁸ vinicius.24100293ead@alunos.unisagrado.edu.br

1 INTRODUÇÃO

O projeto faz parte do módulo *Bootcamp: Desenvolvimento e Projeto de Sistemas*, e consiste no desenvolvimento de uma *Application Programming Interface* (API) simples, tendo como proposta temática o gerenciamento de alunos e cursos de uma instituição de ensino.

2 PROPOSTA

A proposta foi disponibilizada em documentação à parte, conforme anexo A.

2.1 ESCOPO

O projeto abrange somente o processamento e retorno de requisições em nível do servidor *web*, sem a necessidade da implementação de interfaces gráficas, podendo-se fazer uso de outras ferramentas para visualização dos pacotes de dados enviados e recebidos. O gerenciamento dos dados deve ocorrer por meio do uso de Sistemas de Gerenciamento de Banco de Dados (SGBD), conectando-o à API no servidor *web*.

3. REQUISITOS

A coleta de requisitos foi realizada a partir da análise do documento contendo as instruções do projeto (vide anexo A).

3.1 REQUISITOS FUNCIONAIS

Identificador	Requisito	Prioridade
RF001	Cadastrar novos alunos	Essencial
RF002	Adicionar alunos aos cursos	Essencial
RF003	Listar alunos já cadastrados	Essencial
RF004	Cadastrar novos cursos	Essencial
RF005	Listar cursos já cadastrados	Essencial
RF006	Listar alunos por curso (filtrar por curso específico)	Desejável
RF007	Listar cursos em que determinado aluno está cadastrado (filtrar por aluno específico)	Desejável

Tabela 1 – Organização dos requisitos funcionais

3.2 REQUISITOS NÃO-FUNCIONAIS

Identificador	Categoria	Descrição	Prioridade
RNF001	Projeto	A aplicação deverá utilizar uma das seguintes linguagens orientadas a objeto: <i>Python (Flask)</i> , <i>Java Spring</i> ou <i>NodeJS</i> .	Essencial
RNF002	Projeto	O banco de dados a ser utilizado deverá ser gratuito e do tipo relacional, dentre as opções: <i>SQLite</i> , <i>MySQL</i> ou <i>PostgreSQL</i> .	Essencial
RNF003	Projeto	As requisições e retorno de dados deverão ser realizadas através de ferramentas como <i>Postman</i> , <i>Insomnia</i> ou similares.	Essencial
RNF004	Previsibilidade	Os dados retornados e enviados deverão estar na estrutura esperada (corpo da requisição padronizado)	Essencial
RNF005	Confiabilidade	Os dados enviados devem ser validados antes de gravados	Essencial
RNF006	Confiabilidade	Não deve haver nenhuma falha no processamento das requisições (envio e recebimento) em todos os <i>endpoints</i>	Essencial
RNF007	Portabilidade	A API poderá ser consumida em qualquer navegador e sistema operacional	Importante
RNF008	Extensibilidade	A implementação de novas funcionalidades deverá ser facilitada, evitando dependências profundas e recursos obsoletos	Importante
RNF009	Desempenho	O processamento de requisições não deve demorar mais que 400 ms	Desejável

Tabela 2 – Organização dos requisitos não-funcionais

4 DIAGRAMAS UML (UNIFIED MODELING LANGUAGE)

A elaboração dos diagramas organiza as diferentes visões do projeto de maneira gráfica e intuitiva, de maneira a simplificar a implementação independente da tecnologia escolhida. Para este projeto, foram empregados os diagramas de: casos de uso e de classe.

4.1 DIAGRAMA DE CASOS DE USO

Esse diagrama recai sobre questões hipotéticas de implementação (como em uma possível expansão), visto que o escopo do projeto não abrange restrições de usuário, tanto de interface quanto de privilégios (na implementação real, qualquer usuário poderá realizar todos os tipos de requisição).

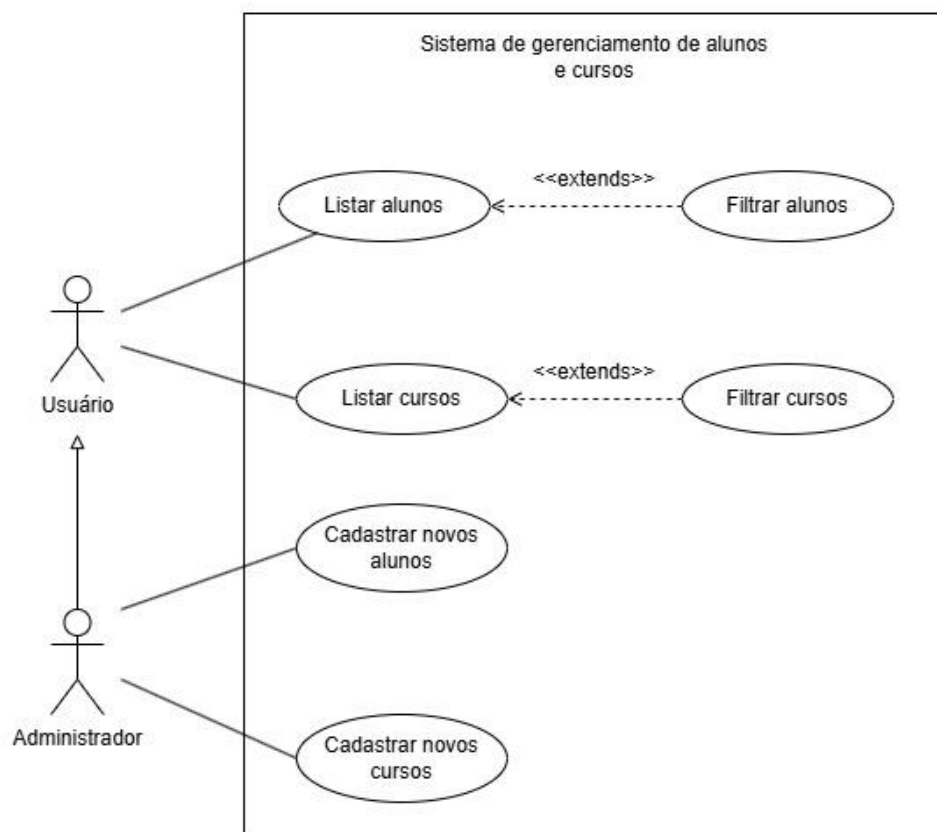


Figura 1 – Diagrama de casos de uso

4.2 DIAGRAMA DE CLASSE

[será criado posteriormente]

5 MODELAGEM DO BANCO DE DADOS

Foi possível elencar, a partir dos requisitos extraídos, os atributos e entidades necessárias para o funcionamento da API, conforme os diagramas a seguir:

5.1 MODELO CONCEITUAL

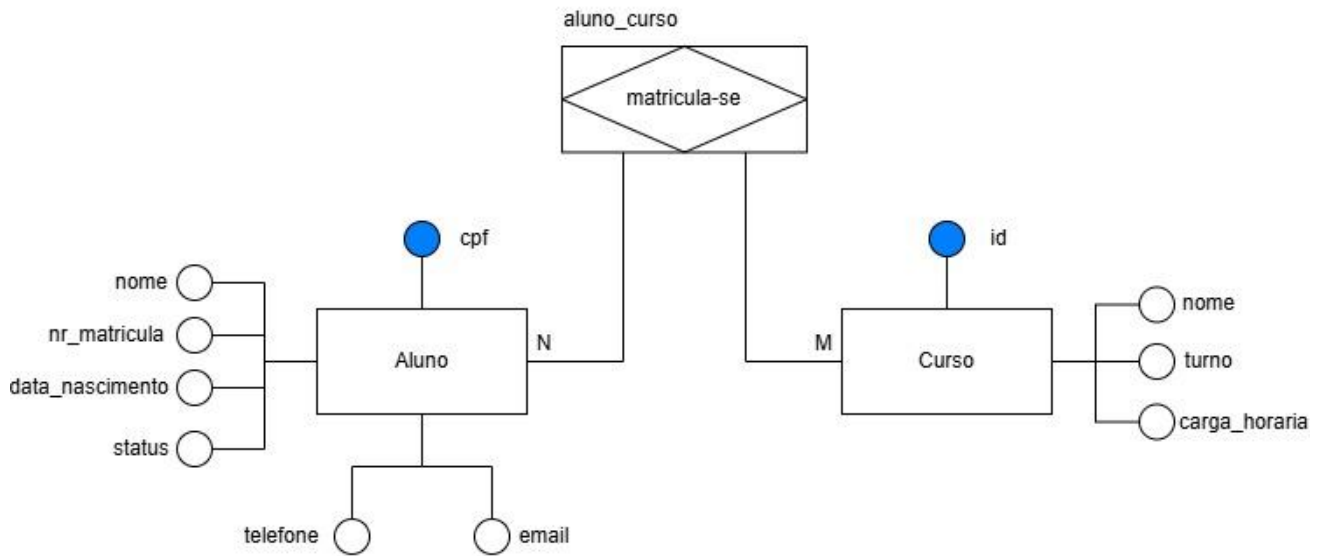


Figura 2 – Diagrama conceitual, notação Peter Chen

5.2 MODELO LÓGICO

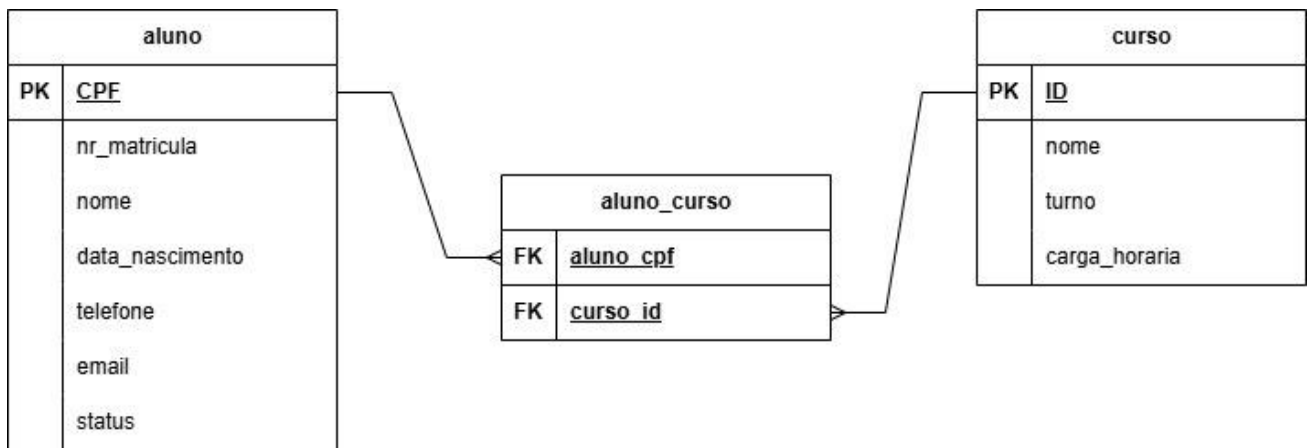


Figura 3 – Diagrama lógico, com tabela de junção

5.3 MODELO FÍSICO

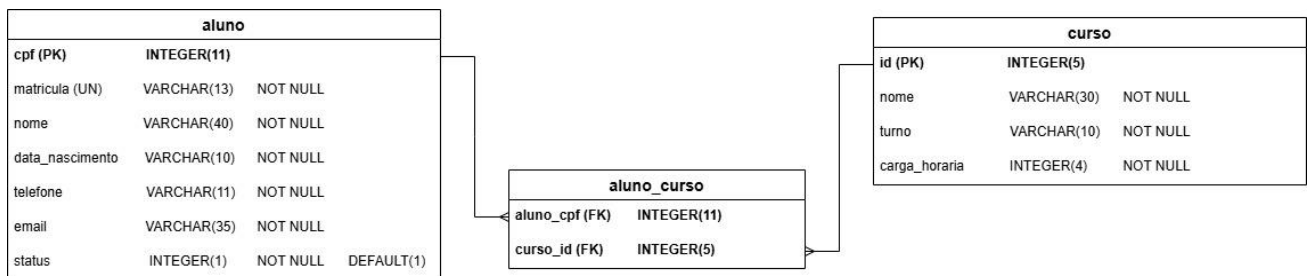


Figura 4 – Diagrama físico, para o SGBD SQLite

ANEXO A – Proposta do projeto disponibilizada pelo professor responsável

Projeto: API Simples para Gerenciamento de Bootcamp

1. Introdução

Este projeto tem como objetivo criar uma API simples para gerenciamento. A API permitirá o cadastro e consulta de alunos e cursos.

2. Objetivos

- Criar uma API básica para gerenciar alunos e cursos.
- Permitir cadastro e listagem de alunos e cursos.

3. Tecnologias Utilizadas

Sugestão de tecnologias:

- **Linguagem:** Python (Flask); Java; Java Spring ou NodeJS.
- **Banco de Dados:** SQLite (simples e fácil de usar); MySQL; PostgreSQL

Observação: não é permitido o uso de linguagens e/ou tecnologias que não foram citadas no documento. Seu uso acarretará nota zero.

4. Estrutura da API

4.1 Endpoints

Alunos

- POST /alunos - Cadastrar um novo aluno.
- GET /alunos - Listar todos os alunos.

Cursos

- POST /cursos - Criar um novo curso.
- GET /cursos - Listar todos os cursos.

Primeira entrega (31/03/25): projeto no GitHub em seu estado atual.

Segunda entrega (24/04/25): projeto no GitHub (completo) e apresentação do projeto (obrigatória).

Datas e horários de apresentação:

26/04 pela manhã

****A data de apresentação está sujeita a alterações com aviso prévio.**