Projeto IcarUSP: Evolução do Backend

Relatório Final - Sistema de Catalogação e Busca de Livros

Isabela Miki & Jonas Rodrigues

2 de julho de 2025

Visão Geral do Projeto

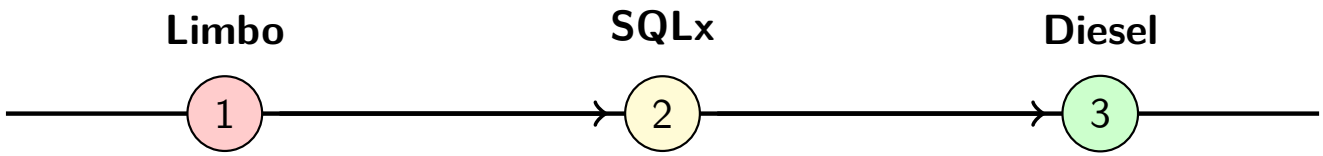
- **Objetivo:** Sistema simplificado de catalogação e busca de livros
- **Problema:** Interface complexa do Dedalus (sistema atual da USP)
- **Solução:** Sistema com busca por similaridade usando embeddings
- **Foco desta apresentação:** Evolução técnica do backend

Evolução do Banco de Dados

Motivos da mudança:

- Limbo: projeto experimental, documentação limitada
- PostgreSQL: solução consolidada, amplo suporte
- pgvector: extensão madura para busca vetorial

Evolução dos Frameworks ORM



SQLite nativo
Sem ORM

Async/await
Query builder

ORM completo
Type-safe

Necessidade de melhor integração com PostgreSQL

Busca por maior abstração e segurança de tipos

ORM

```rust
#[derive(Queryable, Selectable, Identifiable, PartialEq, Debug, Clone)]
#[diesel(table_name = books)]
#[diesel(check_for_backend(diesel::pg::Pg))]
pub struct Book {
    pub id: i32,
    pub title: String,
    pub publication_year: i32,
    pub abstract_text: String,
    pub embedding: Option<Vector>,
}

#[derive(Insertable, Debug)]
#[diesel(table_name = books)]
pub struct NewBook {
    pub title: String,
    pub publication_year: i32,
    pub abstract_text: String,
    pub embedding: Option<Vector>,
}
```

Arquitetura Final do Backend

- **Linguagem:** Rust (performance + segurança de memória)
- **Framework Web:** Actix-web (async, alta performance)
- **ORM:** Diesel (type-safe, compile-time checks)
- **Banco de Dados:** PostgreSQL + pgvector
- **Funcionalidades implementadas:**
  - CRUD para autores e livros
  - Sistema de relacionamentos (N:M)
  - Busca por similaridade com embeddings
  - API REST completa

Busca Vetorial - Implementação

- **Algoritmo:** N-gram hashing (3-gramas)
- **Dimensionalidade:** Vetores de 512 dimensões
- **Normalização:** *L2 norm* para consistência
- **Métrica de similaridade:** Distância cosseno
- **Vantagens:**
  - Busca semântica (não apenas palavras-chave)
  - Performance otimizada com pgvector
  - Implementação leve (sem dependências externas)

Cobertura de Testes - *cargollvm — cov*

Módulo	Regions	Cover	Functions
database/initialization.rs	35	77.14%	5
database/insertion.rs	275	90.18%	13
database/query.rs	328	63.41%	25
embedding/mod.rs	103	**99.03%**	11
lib.rs	67	74.63%	10
models/author.rs	46	71.74%	9
models/book.rs	99	**90.91%**	13
models/book_author.rs	29	75.86%	7
schema.rs	3	**100.00%**	3
TOTAL	**985**	**79.48%**	**96**

- **18 testes** executados com sucesso
- Módulo de embeddings com cobertura quase perfeita
- Foco em testes unitários e de integração

Endpoints da API REST

Método	Endpoint	Descrição
POST	/insert/author	Cadastrar novo autor
POST	/insert/book	Cadastrar novo livro
POST	/insert/book-author-link	Vincular livro a autores
POST	/search/authors	Buscar autores por nome
POST	/search/books	Buscar livros por título
POST	/search/author/books	Buscar livros por autor
POST	/search/book/embedding	Busca por similaridade

- API completa com CORS habilitado
- Tratamento robusto de erros
- Validação de entrada em todos os endpoints

Lições Aprendidas

- **Escolha de tecnologias:** Importância de avaliar maturidade e suporte
- **Iteração rápida:** Prototipagem permitiu identificar limitações cedo
- **Type safety:** Diesel preveniu muitos bugs em tempo de compilação
- **Testes:** Cobertura alta nos módulos críticos (embedding, models)
- **Documentação:** Tecnologias maduras facilitam desenvolvimento

Demonstração do Sistema

Funcionalidades a demonstrar:

- Cadastro de autores e livros
- Busca tradicional por título/autor
- Busca por similaridade semântica
- Visualização de relacionamentos

Muito Obrigado!

Perguntas?

Projeto IcarUSP
Sistema de Catalogação e Busca de Livros