Recommender & Semantic Agent Server

This repository contains the backend services for a data-driven educational recommender system grounded in explicit domain structure and a theory of systematic errors.

The system infers student learning gaps by analyzing distributions of structured errors observed in student interactions with BNCC-aligned mathematics challenges. Rather than learning curricular structure from data, it operates under a priori pedagogical hierarchies defined by the BNCC and uses empirical error frequencies to distinguish between systematic and random errors.

At the core of the project lies an adaptation of the theory of systematic versus random errors, originally developed in experimental physics, applied here to the epistemology of learning. Within this framework, persistent error patterns are interpreted as indicators of incomplete or fragile conceptual understanding, while sporadic errors are treated as noise or absence of knowledge.

The system is composed of two main services:

- **Semantic Agent**: responsible for generating and maintaining structured error tables
  per BNCC skill, respecting prerequisite hierarchies.
- **Recommender**: responsible for diagnosing systematic learning gaps from student
  histories and generating targeted pedagogical recommendations.

This README documents the project from a **local development perspective only**.
It intentionally avoids prescribing decisions related to deployment, orchestration,
security, or scalability. These choices are left to Engineering and DevOps, based on
the explicit contracts and behaviors described here.

Execução Desenvolvimento

1.1 Pré-requisitos

- Python ≥ 3.10
- MySQL rodando localmente
- Redis rodando localmente
- Variáveis `.env` **não são usadas** (export inline)
- Portas padrão:

- Recommender: `8000`
- Semantic Agent: `8001`

1.2 Redis

Commando de teste:

```
redis-cli -h 127.0.0.1 -p 6379 ping
```

Resposta esperada:

```
PONG
```

---

2. Subir o Recommender (Flask – porta 8000)

2.1 Garantir que a porta está livre

```
lsof -ti :8000 | xargs kill -9
```

2.2 Subir o serviço

```
DB_HOST="127.0.0.1" \

DB_PORT="3306" \

DB_USER="root" \

DB_PASS="XXXXXXX" \

DB_PASSWORD="XXXXXXX" \

DB_NAME="JN_DB" \

DB_NAME_1="SGP_DB" \
REDIS_HOST="127.0.0.1" \

REDIS_PORT="6379" \

REDIS_DB="0" \

PORT=8000 \

python app.py
```

Saída esperada (aproximada):

```
Running on http://127.0.0.1:8000
```

2.3 Health-check do Recommender

Health simples (se existir rota):

```
curl -i http://127.0.0.1:8000/health
```

Se **não houver rota explícita**, validar com endpoint real:

```
curl -i http://127.0.0.1:8000/get_recommendations
```

Resposta esperada:

- 405 Method Not Allowed (ok)
- ou 400 com JSON de erro (ok)

3. Subir o Agente Semântico (porta 8001)

3.1 Garantir que a porta está livre

```
lsof -ti :8001 | xargs kill -9
```

3.2 Subir o serviço

```
OPENAI_API_KEY="sk-XXXXXXXXXXXXXXXXXXXXX" \

CURATED_BUNDLES_JSON="/Users/andresardao/curated_bundles.json" \

DB_HOST="127.0.0.1" \

DB_PORT="3306" \

DB_USER="root" \

DB_PASS="XXXXXXXXXX" \

DB_PASSWORD="XXXXXXXX" \

DB_NAME="JN_DB" \

DB_NAME_1="SGP_DB" \

REDIS_HOST="127.0.0.1" \

REDIS_PORT="6379" \

REDIS_DB="0" \

PORT=8001 \

python error_table_app.py
```

Saída esperada:

```
Running on http://127.0.0.1:8001
```

3.3 Health-check do Agente Semântico

```
curl -i http://127.0.0.1:8001/
```

Ou, se existir rota dedicada:

```
curl -i http://127.0.0.1:8001/health
```

---

4. Rodar o Recommender (teste funcional)

4.1 Requisição padrão

```
curl -s -X POST http://127.0.0.1:8000/get_recommendations \
  -H "Content-Type: application/json" \
  -d '{"data_challenge_id":476522}' | jq
```

4.2 IDs (data_challenge_id) úteis para testes manuais:

EF05MA01 → 513200
EF05MA05 → 476522
EF05MA03A → 458626
Português → 447508
EF05MA15 → 494637
EF05MA23 → 480097
EF05MA03 → 458691

5. Rodar o Agente Semântico (geração de tabelas)

5.1 Disparo manual

```
curl -s -X POST http://127.0.0.1:8001/generate_error_tables \
  -H "Content-Type: application/json" \
  -d '{"hab":"EF05MA06","yr":5,"sub":2}' | jq
curl -s -X POST http://127.0.0.1:8001/generate_error_tables \
  -H "Content-Type: application/json" \
  -d '{"hab":"EF05MA05","yr":5,"sub":2}' | jq
```

Resposta esperada (exemplo):

```
{
  "hab": "EF05MA06",
  "tables": [
    "classificacao_erros_EF05MA06.csv"
  ],
  "notify": {
    "attempted": true,
    "ok": true
  }
}
```

5.2 Verificação de efeitos colaterais

- Arquivo CSV gerado em disco
- Redis atualizado (se aplicável)
- Recommender apto a consumir nova tabela

Opcional:

```
ls | grep classificacao_erros
```

---

6. Debug rápido (quando algo dá errado)

6.1 Portas

```
lsof -i :8000
lsof -i :8001
```

6.2 Redis

```
redis-cli monitor
```

6.3 MySQL (queries ativas)

```
SHOW PROCESSLIST;
```

---

7. Shutdown limpo (manual)

```
lsof -ti :8000 | xargs kill -9
lsof -ti :8001 | xargs kill -9
```

---

8. Ordem recomendada de execução

1. MySQL
2. Redis
3. Recommender (`8000`)
4. Semantic Agent (`8001`)
5. Testes manuais (curl)