Projeto – Engenharia de Software + Inteligência Artificial

Lyrics Classifier

Jean Victor Yoshida Lima, João Pedro Cabrera Rodrigues Penna, João Vitor Gozzo Bruschi, Nícolas Justo Melão

Resumo e Objetivos

Construir um pipeline de ML para classificar letras de músicas em categorias (gênero, humor/valência, década, tema), com entrega de componente reutilizável (joblib/ONNX) e opção de serviço HTTP (FastAPI). O consumo deve ser possível localmente (artefatos) e remotamente (API), com reprodutibilidade e documentação de design.

Metodologia

- Dados: planilha XLSX (data/dataset_genero_musical.xlsx) com colunas musica e genero; ingestão validada. Recomendado versionamento (Git LFS/S3) e metadados (hash, data, origem).

- Pré-processamento: limpeza, normalização, remoção de stopwords, lematização opcional; idioma configurável (pt/en).

- Features: TF-IDF (obrigatório) com n-gramas (1–2) e filtros; Embeddings (opcional) via sentence-transformers.

- Modelos: Baseline Naive Bayes (Multinomial/Bernoulli); alternativas: Regressão Logística, Linear SVM (calibrada), Random Forest, XGBoost. Seleção pelo melhor F1-macro em hold-out (opcional k-fold).

- Avaliação: Accuracy, F1-macro, classificação por classe, matriz de confusão (PNG). Repetibilidade com random_state.

- Exportação: component.joblib contendo pipeline e metadados (config, classes, VERSION); metrics.json, config.json, VERSION. ONNX opcional quando suportado.

- Serviço: FastAPI com /predict, /metadata, /health.

Resultados

- Dataset demonstrativo (gênero musical) em PT-BR. Ao rodar scripts/train.py, o melhor modelo e métricas são gravados em reports/ e artifacts/

- Itens gerados:

  - reports/metrics_<modelo>_<ts>.json e reports/confusion_<modelo>_<ts>.png

  - artifacts/<modelo>_<ts>_<hash>/component.joblib e metadados

Análise de Erros

- Verificar classes com F1 baixo e exemplos mal classificados.

- Inspecionar vocabulário TF-IDF; avaliar stemming/lematização e n-gramas.

- Considerar re-balanceamento (class weights/oversampling) e calibração de probabilidades.

- Para embeddings, testar modelos específicos para PT-BR (e.g., paraphrase-multilingual-mpnet-base-v2).

Reprodutibilidade

- VERSION e config.json copiados no artefato; seeds fixos; scripts determinísticos.

- Recomenda-se registrar hash do dataset e commit do código na publicação do artefato.

Ética

- Revisar vieses no corpus (representatividade de gêneros, décadas, temas); evitar decisões automatizadas sem supervisão.

- Garantir que o consumo do modelo respeite direitos autorais de letras e privacidade.

Diagramas

Diagramas de engenharia em formato PNG (gerados a partir do código Mermaid):

Diagrama de Classes

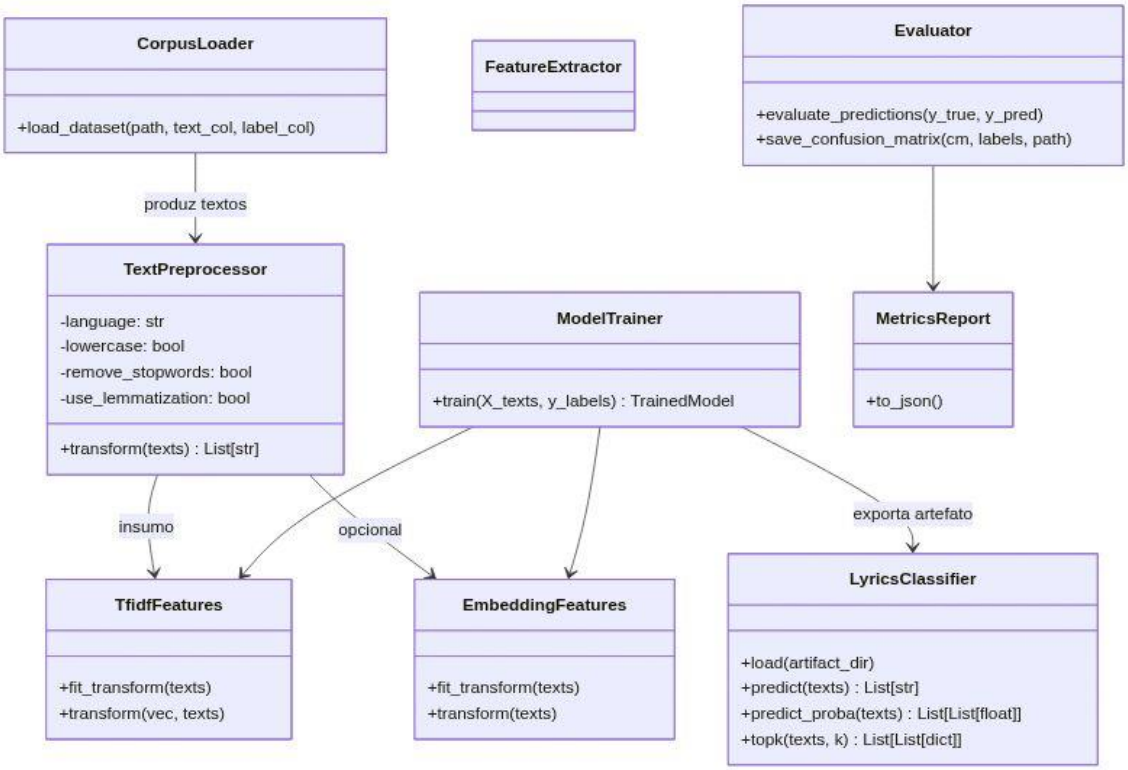


Diagrama de Componentes

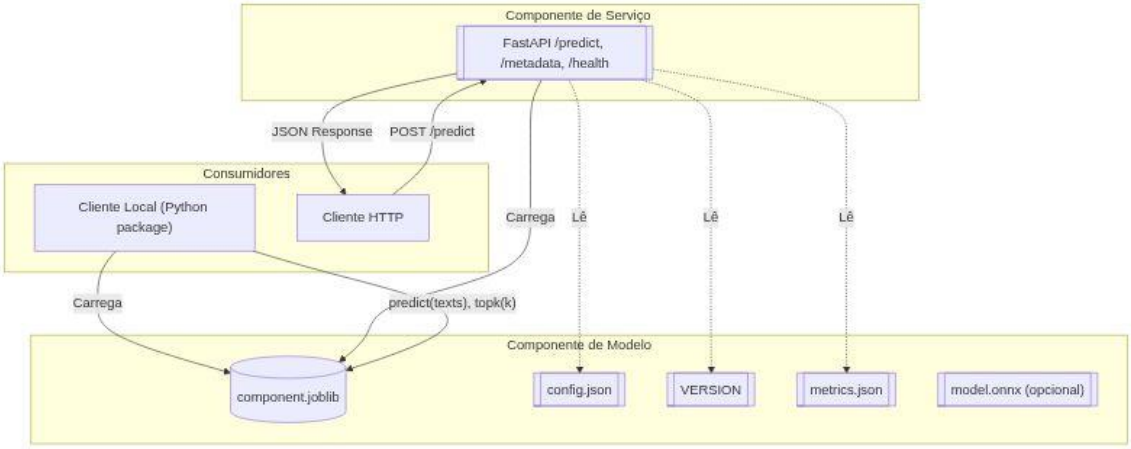
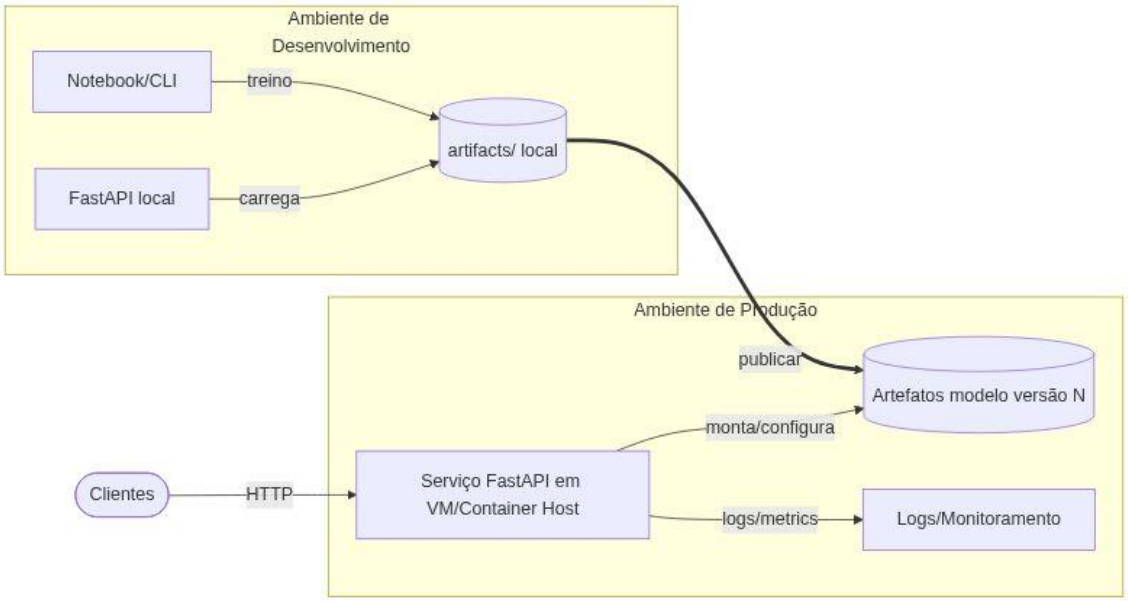
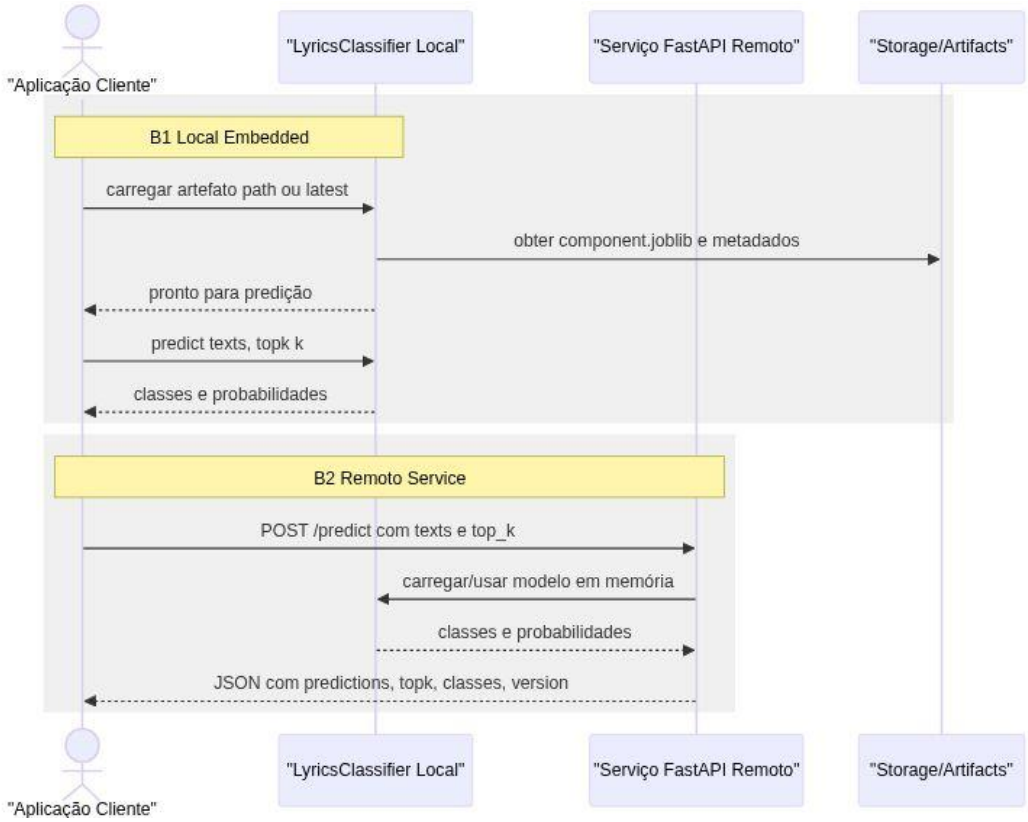


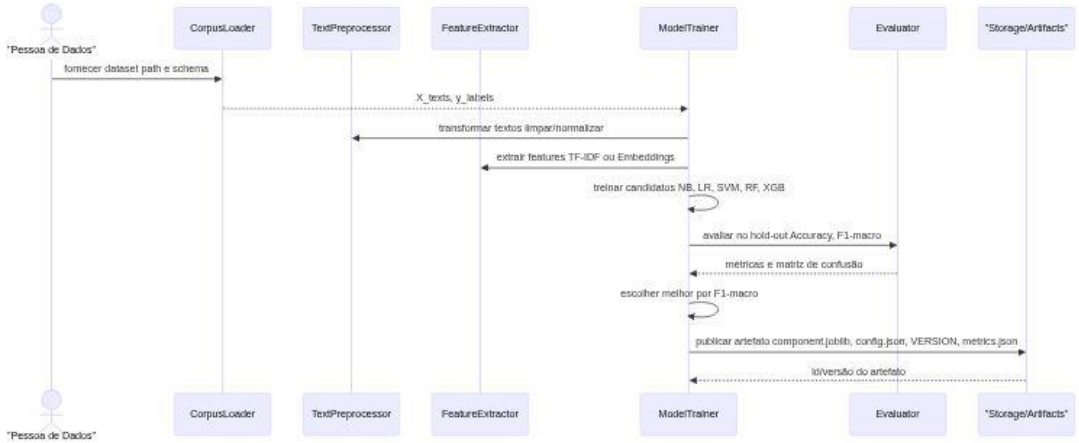
Diagrama de Implantação



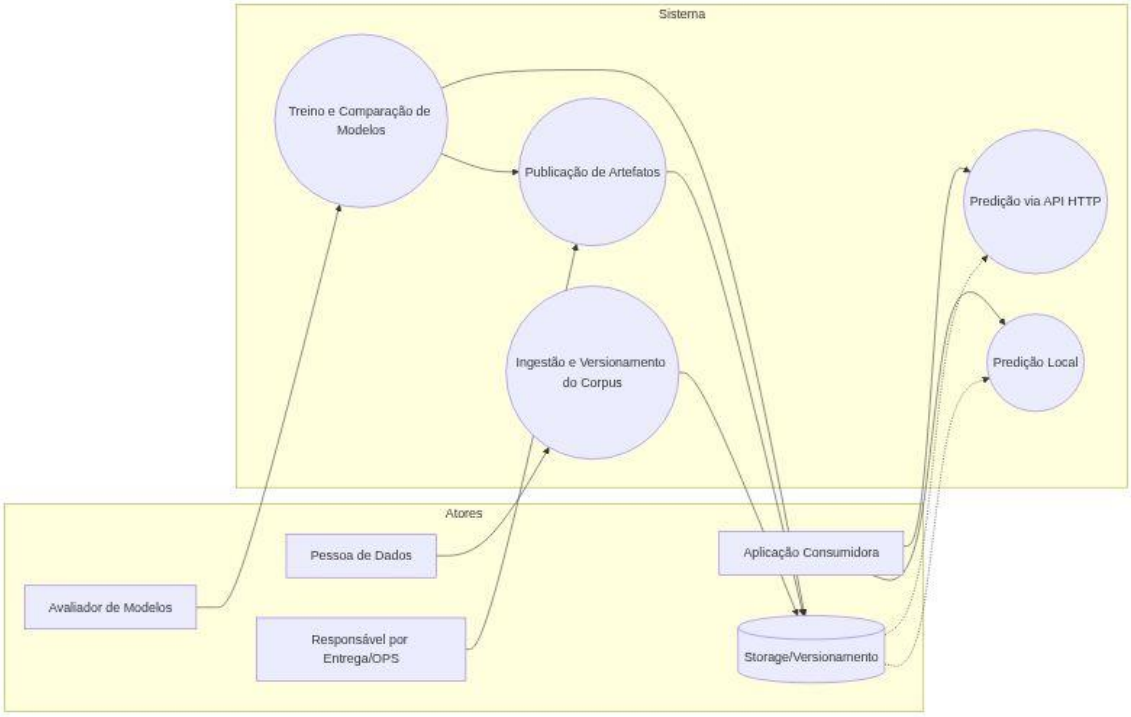
Sequência – Interferência/Consumo



Sequência – Treino/Validação/Publicação



Casos de Uso



Contrato da API

- POST `/predict`

  - Request: `{ "texts": [string], "top_k": int }`

  - Response: `{ "predictions": [string], "topk": [[{label, prob}]], "classes": [string], "version": string }`

- GET `/metadata` → `{version, model_name, classes, artifact_dir}`

- GET `/health` → `{status}`

Como Reproduzir

1) Instalar dependências e baixar NLTK (ver README).

2) Treinar: `python scripts/train.py --dataset data/dataset_genero_musical.xlsx --language pt`

3) Predizer local: `python scripts/predict.py --texts "exemplo de letra"`

4) Subir API: `uvicorn service.app:app --port 8000`; testar `/predict`.