`selecao/prova-pratica/avaliacao_mobile.md`

Avaliação Prática – Desenvolvedor Mobile

Tempo

- 3h: desenvolvimento da solução + tutorial
- 1h30: gravação da apresentação ou apresentação ao vivo

Objetivo

Criar um aplicativo mobile simples que classifique mensagens textuais.

Cenário

Um aplicativo mobile é necessário para testar a classificação de mensagens digitadas pelo usuário.

Requisitos mínimos

- App com uma tela
- Campo de texto para entrada
- Botão para classificar
- Exibição da categoria resultante
- App deve compilar e executar, consumindo um serviço de classificação (vide anexo)

Tecnologias

Livre escolha (Flutter, React Native, Android, iOS).

Entregas

- Código-fonte do aplicativo
- README.md em formato de tutorial técnico
- Instruções de execução

Apresentação

Explicar arquitetura, fluxo de dados, decisões técnicas, limitações e evoluções.

Anexo: Exemplo de script de classificação simples

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel, Field
from typing import Literal, Optional
import time

app = FastAPI(title="Mini Text Service", version="1.0.0")


class ClassifyRequest(BaseModel):
    text: str = Field(..., min_length=1, max_length=2000)
    strategy: Optional[Literal["rules"]] = "rules"


class ClassifyResponse(BaseModel):
    category: Literal["pergunta", "relato", "reclamacao"]
    confidence: float
    strategy: str
    elapsed_ms: int


@app.get("/health")
def health():
    return {"status": "ok"}


@app.get("/info")
def info():
    return {
        "service": "mini-text-service",
        "version": "1.0.0",
        "endpoints": ["/health", "/info", "/classify", "/echo"],
    }


@app.post("/echo")
def echo(payload: dict):
    # Útil para testar requests/response e debug de rede
    return {"received": payload}


def classify_rules(text: str) -> tuple[str, float]:
    t = text.strip().lower()

    # Heurísticas simples, suficientes para a avaliação (não é um modelo)
    if "?" in t or t.startswith(("como ", "por que ", "pq ", "qual ", "quais ")):
        return "pergunta", 0.85
```

```
    if any(k in t for k in ["não funciona", "erro", "ruim", "problema", "insatisfeito",
"reclama"]):
        return "reclamacao", 0.75

    return "relato", 0.60


@app.post("/classify", response_model=ClassifyResponse)
def classify(req: ClassifyRequest):
    start = time.time()

    text = (req.text or "").strip()
    if not text:
        raise HTTPException(status_code=400, detail="text must be non-empty")

    if req.strategy != "rules":
        raise HTTPException(status_code=400, detail="unsupported strategy")

    category, confidence = classify_rules(text)
    elapsed_ms = int((time.time() - start) * 1000)

    return ClassifyResponse(
        category=category,
        confidence=confidence,
        strategy=req.strategy,
        elapsed_ms=elapsed_ms,
    )
```

Conteúdo do arquivo requirements.txt para instalar as dependências:

```
fastapi>=0.110,<0.116
uvicorn[standard]>=0.30,<0.34
pydantic>=2.9,<3.0
```

Como instalar as dependências:

```
pip install -r requirements.txt
```

Como executar o serviço localmente:

```
# Considerando o nome do script como app.py
uvicorn app:app --host
```