Projeto: Chatbot Gamificado para Ensino de Pontos Históricos do Recife

1. Definição do Projeto

Sistema educacional gamificado que utiliza Deep Learning para ensinar e avaliar conhecimentos sobre pontos históricos do Recife através de três modos de interação:

- **Modo Foto**: Comparação visual de fotos usando Deep Learning
- **Modo Descrição**: Avaliação de descrições usando NLP
- **Modo Mistério**: Desafio de identificação histórica

---

2. Problemática Escolhida

Problema Principal

Como tornar o aprendizado sobre patrimônio histórico do Recife mais interativo, envolvente e eficaz, especialmente para estudantes e turistas que precisam aprender sobre os locais históricos da cidade?

Questões Específicas

1. **Engajamento Educacional**: Estudantes têm dificuldade em memorizar informações históricas tradicionais

2. **Avaliação Objetiva**: Como avaliar o conhecimento adquirido de forma precisa e justa?

3. **Interatividade**: Como tornar o aprendizado mais dinâmico e menos monotônico?

4. **Tecnologia Educacional**: Integrar IA/Deep Learning com métodos de ensino tradicionais

Solução Proposta

Sistema chatbot gamificado que combina:

- Análise visual de imagens com CNN

- Processamento de linguagem natural (NLP) para avaliação de descrições
- Gamificação (pontos → XP, níveis, conquistas)
- Interface interativa e intuitiva

---

3. Datasets Utilizados

3.1 Dataset de Imagens Históricas

- **Fonte**: `data/recife_historic/`
- **Estrutura**: 12 classes (uma pasta por local histórico)
- **Estado atual no filesystem**: 6 imagens presentes na pasta `casa_da_cultura/`. As demais pastas estão criadas mas vazias.
- **Locais**: Casa da Cultura, Forte das Cinco Pontas, Igreja Madre de Deus, Igreja Nossa Senhora do Carmo, Igreja Santo Antônio, Igreja São Pedro dos Clérigos, Marco Zero, Mercado São José, Palácio da Justiça, Rua Aurora, Rua do Bom Jesus, Teatro Santa Isabel

- **Registros**: 23 entradas

3.2 Dataset de Descrições

- **Fonte**: `data/photo_descriptions.json`
- **Formato**: JSON com descrições oficiais e keywords
- **Campos**:
- `id`: Identificador único
- `image_path`: Caminho da imagem
- `name`: Nome do local
- `official_description`: Descrição oficial detalhada
- `keywords`: Lista de palavras-chave relevantes
- `difficulty`: Nível de dificuldade (Fácil, Médio, Difícil)
- `points`: Pontos associados ao desafio (usados como XP base)

3.3 Recursos de NLP

- **Stopwords**: 207 palavras em português (NLTK)
- **Lemmatization**: WordNet (português)
- **Tokenization**: Regex-based (mais robusta que NLTK para português)

---

4. Pipeline/ETL

4.1 Fluxo de Processamento de Imagens

```
Usuário envia foto
    ↓
Decodificação Base64 → PIL Image
    ↓
Transforms (resize, normalize)
    ↓
Modelo CNN (ImprovedCNN)
    ↓
    ├─→ Predição de Classe (local histórico)
    └─→ Confiança (%)
```

4.2 Fluxo de Processamento de Texto (Modo Descrição)

```
Usuário escreve descrição
    ↓
Limpeza de texto (remove acentos, lowercase)
    ↓
Tokenization (regex: \b\w+\b)
    ↓
Remove stopwords
    ↓
Lemmatization (WordNet)
```

```
    ↓
TF-IDF Vectorization
    ↓
Cosine Similarity
    ↓
Score combinado (60% similaridade + 40% keywords)
```

4.3 Fluxo de Comparação Visual (Modo Foto)

```
Foto do Desafio + Foto do Usuário
    ↓
    ├→ Classifica Foto do Desafio → Classe A
    └→ Classifica Foto do Usuário → Classe B
    ↓
Compara Classes:
    ├→ A == B: Alta similaridade
    └→ A != B: Baixa similaridade
```

4.4 Pipeline de Gamificação (Unificado em XP)

```
Ação do Usuário
    ↓
Calcula Pontos (baseado em score)
    ↓
Adiciona XP
    ↓
Verifica Level Up
    ↓
Atualiza Estatísticas
```

---

5. Modelos Escolhidos

5.1 Modelo de Visão (CNN)

Arquitetura: `ImprovedCNN` (do zero) e opção `ResNet18` (Transfer Learning)

```
Features (CNN):
  - Conv2d(3→64, kernel=7, stride=2)
  - MaxPool2d(kernel=3, stride=2)
  - Conv2d(64→128, kernel=3)
  - MaxPool2d(kernel=2)
  - Conv2d(128→256, kernel=3)
  - Conv2d(256→256, kernel=3)
  - MaxPool2d(kernel=2)
  - Conv2d(256→512, kernel=3)
  - Conv2d(512→512, kernel=3)
  - AdaptiveAvgPool2d(4x4)

Classifier:
  - Flatten
  - Linear(8192→1024) + ReLU + Dropout(0.5)
  - Linear(1024→512) + ReLU + Dropout(0.25)
  - Linear(512→256) + ReLU + Dropout(0.15)
  - Linear(256→num_classes)
```

Total de Parâmetros: 13,704,972 (ImprovedCNN)

Classes: 12 locais históricos

Input Size: 224x224 RGB

Optimizer: AdamW (lr=0.001, weight_decay=0.01)

Loss: CrossEntropyLoss (label_smoothing=0.1)

Scheduler: ReduceLROnPlateau

5.2 Modelo de NLP

Arquitetura: TF-IDF + Cosine Similarity

```
Vectorizer: TfidfVectorizer(max_features=1000)

Similarity: cosine_similarity(matrix1, matrix2)

Score Final: (similarity * 0.6) + (keyword_score * 0.4)
```

Preprocessamento:

- Tokenization: `re.findall(r'\b\w+\b', text)`
- Stopword Removal: NLTK Portuguese (207 words)
- Lemmatization: WordNet Portuguese

---

6. Endpoints da API

6.1 `POST /api/compare_visual_similarity`

- Entrada: `user_image (base64)`, `target_location`, `player_id`

- Saída: `similarity_score`, `points_earned`, `success`
- Lógica de pontos: baseado na similaridade (≥0.8 = 100% dos pontos; ≥0.6 = 70%; ≥0.4 = 50%; <0.4 = 30%)
- Efeito: soma `points_earned` ao **XP** do jogador, atualiza tentativas, acertos (se ≥ 0.6), streak e nível

6.2 `POST /api/photo_game/submit_description`

- Entrada: `description`, `photo_id`, `player_id`
- Saída: `final_score`, `keyword_score`, `points_earned`, `is_correct`, `total_xp`, `success`
- Lógica de pontos: baseada no score final (threshold 0.4), com multiplicadores conforme desempenho
- Efeito: soma `points_earned` ao **XP**, atualiza tentativas, acertos, streak e nível

6.3 `GET /api/player_stats/:player_id`

- Retorna: `name`, `level`, `experience (XP)`, `streak`, `total_correct`, `total_attempts`, `accuracy`, `achievements`

---

7. Interface (Frontend)

7.1 `templates/chatbot.html`

- Exibe: **Nível**, **XP**, **Precisão**, **Sequência**
- Atualiza XP após cada jogada (Foto/Descrição)
- Mostra `points_earned` retornado pelo backend

7.2 `templates/index.html`

- Dashboard de estatísticas com: **Nível**, **Experiência (XP)**, **Sequência**, **Precisão**

7.3 Observações sobre o Modelo

- O backend carrega automaticamente o arquivo `models/improved_recife_historic_model.pth`.
- O checkpoint salva a arquitetura usada (`ImprovedCNN` ou `ResNet18`).
- O carregamento detecta automaticamente a arquitetura correta pelo metadado ou pelas chaves do `state_dict`.

5.3 Treinamento (atualizado)

- Split estratificado (train/val) e métricas de validação a cada época.
- Early stopping por estagnação na validação.
- Balanceamento no treino com `WeightedRandomSampler`.
- Hiperparâmetros ajustáveis (LR menor no backbone e maior na cabeça para Transfer Learning).

---

8. Gamificação (Regras)

- Sistema unificado: **apenas XP**
- Fórmula de nível: `level = int((XP/100) ** 0.5) + 1`
- Conquistas concedem **XP** (moedas removidas)
- Streak incrementa em acertos; zera em erro

---

9. Execução

1. Inicie o servidor: `python main.py`

2. Acesse `http://localhost:5000/chatbot`

3. Crie/obtenha um `player_id` (rotas de criação já existentes na UI)

4. Jogue nos modos Foto/Descrição — o XP será somado automaticamente

---

10. Estrutura de Arquivos

```
appDeepLearning/
├── data/
│   ├── recife_historic/        # Imagens de locais históricos
│   └── photo_descriptions.json # Descrições oficiais
├── training/
│   ├── improved_recife_trainer.py  # Treinador principal
│   ├── image_trainer.py
│   └── recife_historic_trainer.py
├── models/
│   └── improved_recife_historic_model.pth  # Modelo treinado
├── game/
│   ├── photo_description_game.py  # Jogo de descrições
│   └── gamification.py             # Sistema de gamificação (XP)
├── templates/
│   ├── chatbot.html               # Interface do chatbot
│   └── index.html                 # Interface original
├── main.py                        # Backend Flask
└── requirements.txt               # Dependências
```

---

11. Observações Finais

- O sistema foi atualizado para remover **moedas** e manter somente **XP**
- As rotas agora retornam e aplicam `points_earned` diretamente ao XP
- Documentação e UI atualizadas para refletir essa simplificação