Documentação de Desenvolvimento

BigCard Training

Sistema de Avaliação Técnica

Gerado em: 09/01/2026

Documentação de Desenvolvimento - BigCard Training

Arquitetura

Visão Geral

Sistema modular com perguntas configuráveis via arquivo de texto, eliminando a necessidade de alterar código para personalizar questionários. Inclui script .bat para facilitar execução no Windows.

Componentes Principais

1. QuestionLoader: Classe responsável por ler e interpretar o arquivo 'perguntas.txt'

2. BigCardHandler: Handler HTTP para servir o formulário e processar respostas

3. Gerador Dinâmico de HTML: Cria formulário baseado nas perguntas carregadas

4. executar.bat: Script Windows para inicialização simplificada (verifica Python e arquivos)

Servidor HTTP (Python)

- Framework: 'http.server' (biblioteca padrão Python)
- Porta: 3000
- Protocolo: HTTP
- Host: 0.0.0.0 (aceita conexões de toda rede local)

Formato do Arquivo perguntas.txt

Estrutura Geral

```
# Comentários começam com # # Linhas vazias são ignoradas NUMERO. Texto da
pergunta # Para múltipla escolha: NUMERO. Texto da pergunta [MULTIPLA_ESCOLHA] a)
Primeira alternativa b) Segunda alternativa c) Terceira alternativa
```

Regras de Formatação

1. Numeração: Cada pergunta deve começar com um número seguido de ponto e espaço

2. Questões Abertas: Apenas o texto da pergunta

3. Múltipla Escolha:

- Adicione '[MULTIPLA_ESCOLHA]' após a pergunta

- Liste alternativas nas linhas seguintes

- Formato: 'letra') texto da alternativa

- Letras válidas: a-j (até 10 alternativas)

4. Comentários: Linhas iniciadas com '#' são ignoradas

5. Linhas Vazias: Também são ignoradas

Exemplo Completo

```
# Seção 1 - Conceitos Básicos 1. O que é uma software house? 2. Explique o conceito de TEF house? # Seção 2 - Procedimentos 3. Qual procedimento correto? [MULTIPLA_ESCOLHA] a) Opção A b) Opção B c) Opção C 4. Descreva o processo de validação?
```

Fluxo de Dados

1. Inicialização do Servidor

```
[Servidor Inicia] ↓ [QuestionLoader lê perguntas.txt] ↓ [Valida formato e organiza dados] ↓ [Servidor aguarda conexões]
```

2. Processo de Carregamento de Perguntas

```
class QuestionLoader: def load_questions(self): # 1. Lê arquivo linha por linha # 2. Ignora comentários e linhas vazias # 3. Detecta início de questão (número + ponto) # 4. Identifica tipo (aberta ou múltipla escolha) # 5. Agrupa alternativas quando necessário # 6. Armazena em estrutura de dados
```

Estrutura de Dados Interna:

```
{ 'numero': 1, 'texto': 'Texto da pergunta', 'tipo': 'aberta' | 'multipla_escolha', 'alternativas': ['a) ...', 'b) ...'] # Apenas para múltipla escolha }
```

3. Geração Dinâmica do HTML

```
[Cliente acessa /] ↓ [Servidor chama get_formulario_html()] ↓ [Itera sobre perguntas carregadas] ↓ [Para cada questão:] - Se aberta: gera <textarea> - Se múltipla: gera <input type="radio"> para cada alternativa ↓ [Retorna HTML completo com JavaScript]
```

4. Submissão e Salvamento

```
[Usuário preenche formulário] ↓ [JavaScript valida respostas] ↓ [POST /enviar com JSON] ↓ [salvar_resposta() formata dados] ↓ [Busca texto da pergunta via QuestionLoader] ↓ [Para múltipla escolha: converte índice em texto] ↓ [Escreve em respostas.txt]
```

Endpoints

GET `/` ou `/formulario`

- Descrição: Retorna o HTML do formulário de avaliação
- Content-Type: 'text/html; charset=utf-8'
- Resposta: HTML completo gerado dinamicamente

POST `/enviar`

- Descrição: Recebe e salva as respostas do formulário
- Content-Type: 'application/json'
- Payload esperado:

```
{ "nome": "Nome do Funcionário", "respostas": [ "resposta texto 1", "2", // Índice da alternativa (múltipla escolha) "resposta texto 3" ] }
```

- Resposta sucesso: '{"success": true}'

Estrutura de Dados

Classe QuestionLoader

```
class QuestionLoader: def __init__(self, filename): self.filename = filename
self.questions = [] # Lista de dicionários self.load_questions() def
get_total_questions(self) -> int def get_question_text(self, index) -> str def
get_question_type(self, index) -> str def get_alternativas(self, index) -> list
```

Arquivo respostas.txt

```
================================================================================
==================== DATA/HORA: 2026-01-08 22:34:21 NOME: Nome do Participante
TOTAL DE RESPOSTAS: 9/9 ============================================================
========================================== 1. Texto da pergunta RESPOSTA:
Resposta do usuário 2. Pergunta múltipla escolha RESPOSTA: c) Alternativa
escolhida ...
```

Frontend Dinâmico

Geração de Questões

Questões Abertas:

```
<div class="question"> <div class="question-title">1. Texto da pergunta</div>
<textarea id="q0" class="form-input"></textarea> </div>
```

Questões de Múltipla Escolha:

```
<div class="question"> <div class="question-title">3. Texto da pergunta</div>
<div class="radio-option"> <input type="radio" name="q2" value="0" id="q2a">
<label for="q2a">a) Alternativa</label> </div> <!-- ... mais alternativas -->
</div>
```

JavaScript Adaptativo

```
const TOTAL_QUESTIONS = N; // Carregado dinamicamente const QUESTION_TYPES =
['aberta', 'multipla_escolha', ...]; // Validação dinâmica baseada no tipo for
(let i = 0; i < TOTAL_QUESTIONS; i++) { if (QUESTION_TYPES[i] ===
'multipla_escolha') { // Valida seleção de radio } else { // Valida texto mínimo
} }
```

Funções Principais

`QuestionLoader.load_questions()`

- Lê arquivo 'perguntas.txt'
- Parseia formato customizado
- Valida estrutura
- Armazena em 'self.questions'

`QuestionLoader.get_question_text(index)`

- Retorna texto formatado: "N. Pergunta"
- Índice baseado em 0

`QuestionLoader.get_question_type(index)`

- Retorna: "aberta" ou "multipla_escolha"

`QuestionLoader.get_alternativas(index)`

- Retorna lista de strings: '['a) ...', 'b) ...']'
- Vazia se não for múltipla escolha

`salvar_resposta(data)`

- Formata timestamp
- Itera sobre respostas
- Busca texto da pergunta via QuestionLoader
- Para múltipla escolha: converte índice em texto completo da alternativa
- Escreve em modo append no arquivo

`get_formulario_html()`

- Gera HTML dinâmico
- Loop sobre 'question_loader.get_total_questions()'
- Cria campos baseado no tipo
- Injeta constantes JavaScript (total, tipos)
- Retorna string HTML completa

Configurações

Arquivo de Perguntas

```
QUESTIONS_FILE = "perguntas.txt" # Linha 16
```

Porta do Servidor

```
PORT = 3000 # Linha 14
```

Arquivo de Respostas

```
DATA_FILE = "respostas.txt" # Linha 15
```

Melhorias Futuras

1. Validação de Formato: Checar sintaxe do perguntas.txt na inicialização
2. Hot Reload: Recarregar perguntas sem reiniciar servidor
3. Múltiplos Questionários: Suporte a vários arquivos de perguntas
4. Banco de Dados: Migrar de .txt para SQLite
5. Dashboard Web: Interface para visualizar respostas
6. Exportação: CSV/Excel das respostas
7. Gabarito: Sistema de correção automática
8. Tipos Adicionais: Caixas de seleção, escala Likert, etc

Tratamento de Erros

Arquivo perguntas.txt Ausente

```
if not os.path.exists(self.filename): raise FileNotFoundError(f"Arquivo
{self.filename} não encontrado!")
```

- Servidor não inicia
- Mensagem clara no terminal

Formato Inválido

- Sistema ignora linhas malformadas silenciosamente
- Logs podem ser adicionados para debug

Validações Cliente

- Nome obrigatório
- Respostas abertas: mínimo 10 caracteres
- Múltipla escolha: seleção obrigatória
- Scroll automático para campo com erro

Encoding

- Python: UTF-8 em todas operações
- HTML: '<meta charset="UTF-8">'
- HTTP: 'charset=utf-8'
- Arquivo: 'encoding='utf-8'' explícito

Garante suporte completo a acentuação e caracteres especiais.

Debug

Verificar Perguntas Carregadas

```
# Adicione no código para debug: for i, q in
enumerate(question_loader.questions): print(f"{i}: {q}")
```

Logs do Servidor

- Suprimidos por padrão (linha 163: 'log_message' vazio)
- Mostra apenas: "■ Nova resposta salva: [Nome]"

Testar Carregamento

```
python3 -c "from server import QuestionLoader; ql =
QuestionLoader('perguntas.txt'); print(f'Total: {ql.get_total_questions()}')"
```

Script executar.bat (Windows)

Funcionalidade

O arquivo 'executar.bat' é um atalho para usuários Windows que:

1. Verifica se Python está instalado no sistema

2. Verifica se 'server.py' existe no diretório

3. Verifica se 'perguntas.txt' existe no diretório

4. Inicia o servidor automaticamente

5. Exibe mensagens de erro claras se algo estiver errado

Estrutura do Script

```
@echo off chcp 65001 >nul # Define codificação UTF-8 title BigCard Training #
Define título da janela # Verificações python --version >nul 2>&1 # Testa se
Python existe if not exist "server.py" # Verifica arquivo if not exist
"perguntas.txt" # Verifica arquivo # Execução python server.py # Inicia servidor
```

Uso

Duplo clique no arquivo 'executar.bat' ou execute via terminal:

```
executar.bat
```

Vantagens

- ■ Não precisa abrir terminal manualmente
- ■ Validações automáticas de pré-requisitos
- ■ Mensagens de erro amigáveis para usuários não-técnicos
- ■ Experiência "plug and play"
- ■ Ideal para distribuição interna

Mensagens de Erro

Python não encontrado:

```
[ERRO] Python nao encontrado! Por favor, instale o Python 3 em:
https://www.python.org/downloads/
```

Arquivo ausente:

```
[ERRO] Arquivo server.py nao encontrado! Certifique-se de que este arquivo .bat
esta na mesma pasta que server.py
```