Manual Completo para Concurso da Caixa - Técnico Bancário Novo em Tecnologia da Informação

Introdução

Este manual é destinado a candidatos que estão se preparando para o concurso da Caixa Econômica Federal para o cargo de Técnico Bancário Novo na área de Tecnologia da Informação. Ele abrange todos os tópicos do edital, explicados de forma clara e objetiva, com exemplos práticos e códigos, para ajudar você a se preparar da melhor maneira possível.

Sumário

---

1. Engenharia de Software

1.1 Processos de Software

Processo Unificado (UP)

- •Conceitos Gerais: O Processo Unificado é um framework de desenvolvimento de software iterativo e incremental. Isso significa que o software é desenvolvido em pequenas partes, chamadas de iterações, e cada parte vai sendo melhorada ao longo do tempo.
- •Disciplinas: As disciplinas do UP incluem:
  - •Modelagem de Negócios: Entender o contexto e os processos de negócios.
  - •Requisitos: Coletar e definir o que o software deve fazer.
  - •Análise e Design: Criar a arquitetura e o design do software.
  - •Implementação: Codificar o software.
  - •Teste: Verificar se o software funciona corretamente.
  - •Implantação: Colocar o software em produção.
  - •Gestão de Configuração e Mudança: Controlar as mudanças no software.
  - •Gestão de Projeto: Planejar e acompanhar o projeto.
  - •Ambiente: Preparar o ambiente de desenvolvimento.

Fases do UP:

- •Concepção: Definir a visão do projeto e sua viabilidade.
- •Elaboração: Detalhar a arquitetura do sistema.
- •Construção: Desenvolver o produto.
- •Transição: Entregar o produto ao usuário final.

Exemplo Prático:
Imagine que você está desenvolvendo um sistema de gestão para uma biblioteca. Na fase de concepção, você define que o sistema deve permitir a gestão de livros e empréstimos. Na fase de elaboração, você cria a arquitetura do sistema, definindo os componentes necessários, como o banco de dados para armazenar as informações dos livros e dos usuários. Na fase de construção, você desenvolve o código para essas funcionalidades. Na fase de transição, você instala o sistema na biblioteca e treina os funcionários para usá-lo.

UX (User Experience)

- •Definição: User Experience (UX) é a experiência que um usuário tem ao interagir com um produto ou sistema. Um bom design de UX torna o sistema fácil de usar e agradável.
- •Princípios de UX:
  - •Utilidade: O sistema deve ser útil.
  - •Usabilidade: O sistema deve ser fácil de usar.
  - •Acessibilidade: O sistema deve ser acessível a todos os usuários, incluindo aqueles com deficiências.
  - •Estética: O sistema deve ser visualmente agradável.

•Design Centrado no Usuário: O design deve focar nas necessidades dos usuários.

---

2. Estrutura de Dados e Algoritmos

2.1 Tipos de Busca e Ordenação

Busca Sequencial

•Definição: Na busca sequencial, você percorre uma lista elemento por elemento até encontrar o valor desejado.

•Exemplo Prático:

```
def busca_sequencial(lista, valor):
    for i in range(len(lista)):
        if lista[i] == valor:
            return i
    return -1
lista = [1, 3, 5, 7, 9]
valor = 5
resultado = busca_sequencial(lista, valor)
print(f"Valor encontrado na posição: {resultado}")
```

Neste exemplo, a função `busca_sequencial` percorre a lista e retorna a posição do valor 5.

Busca Binária

•Definição: A busca binária funciona em listas ordenadas e divide repetidamente o intervalo de busca pela metade até encontrar o valor.

•Exemplo Prático:

```
def busca_binaria(lista, valor):
    esquerda, direita = 0, len(lista) - 1
    while esquerda <= direita:
        meio = (esquerda + direita) // 2
        if lista[meio] == valor:
            return meio
        elif lista[meio] < valor:
            esquerda = meio + 1
        else:
            direita = meio - 1
    return -1
lista = [1, 3, 5, 7, 9]
valor = 5
resultado = busca_binaria(lista, valor)
print(f"Valor encontrado na posição: {resultado}")
```

Aqui, a função `busca_binaria` retorna a posição do valor 5 na lista ordenada.

2.2 Métodos de Ordenação

Bubble Sort

- Definição: O Bubble Sort ordena a lista, trocando elementos adjacentes que estão na ordem errada.
- Exemplo Prático:

```
def bubble_sort(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n-i-1):
            if lista[j] > lista[j+1]:
                lista[j], lista[j+1] = lista[j+1], lista[j]
lista = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(lista)
print("Lista ordenada:", lista)
```

Neste exemplo, `bubble_sort` ordena a lista em ordem crescente.

Selection Sort

- Definição: O Selection Sort encontra repetidamente o menor elemento e o move para a posição correta.
- Exemplo Prático:

```
def selection_sort(lista):
    n = len(lista)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if lista[j] < lista[min_idx]:
                min_idx = j
        lista[i], lista[min_idx] = lista[min_idx], lista[i]
lista = [64, 25, 12, 22, 11]
selection_sort(lista)
print("Lista ordenada:", lista)
```

Aqui, `selection_sort` ordena a lista.

Insertion Sort

- Definição: O Insertion Sort constrói uma lista ordenada, inserindo um elemento de cada vez na posição correta.
- Exemplo Prático:

```
def insertion_sort(lista):
    for i in range(1, len(lista)):
        chave = lista[i]
```

```
        j = i - 1
        while j >= 0 and chave < lista[j]:
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = chave
lista = [12, 11, 13, 5, 6]
insertion_sort(lista)
print("Lista ordenada:", lista)
```

Neste exemplo, `insertion_sort` organiza a lista.

2.3 Estruturas de Dados

Lista Encadeada

- Definição: Uma lista encadeada é uma estrutura de dados linear, onde cada elemento aponta para o próximo.
- Exemplo Prático:

```
class No:
    def __init__(self, dado=None):
        self.dado = dado
        self.proximo = None
class ListaEncadeada:
def init(self):
self.cabeca = None
<span class="token keyword">def</span> <span class="token function">inserir</span><span class="token punctuation">(</span>self<span class="token punctuation">,</span> novo_dado<span class="token punctuation">)</span><span class="token punctuation">:</span>
    novo_no <span class="token operator">=</span> No<span class="token punctuation">(</span>novo_dado<span class="token punctuation">)</span>
    novo_no<span class="token punctuation">.</span>proximo <span class="token operator">=</span> self<span class="token punctuation">.</span>cabeca
    self<span class="token punctuation">.</span>cabeca <span class="token operator">=</span> novo_no

<span class="token keyword">def</span> <span class="token function">exibir</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    no_atual <span class="token operator">=</span> self<span class="token punctuation">.</span>cabeca
    <span class="token keyword">while</span> no_atual<span class="token punctuation">:</span>
        <span class="token keyword">print</span><span class="token punctuation">(</span>no_atual<span class="token punctuation">.</span>dado<span class="token punctuation">,</span> end<span class="token operator">=</span><span class="token string">" -&gt; "</span><span class="token punctuation">)</span>
        no_atual <span class="token operator">=</span> no_atual<span class="token punctuation">.</span>proximo
    <span class="token keyword">print</span><span class="token punctuation">(</span><span class="token string">"None"</span><span class="token punctuation">)</span>
lista = ListaEncadeada() lista.inserir(3) lista.inserir(2) lista.inserir(1)
lista.exibir()
```

Este exemplo demonstra como criar e exibir uma lista encadeada simples.

Pilha (Stack)

- Definição: Uma pilha é uma estrutura LIFO (Last In, First Out), onde o último elemento inserido é o primeiro a ser removido.
- Exemplo Prático:

```
class Pilha:
    def __init__(self):
        self.itens = []
<span class="token keyword">def</span> <span class="token function">esta_vazia</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">return</span> <span class="token builtin">len</span><span class="token punctuation">(</span>self<span class="token punctuation">.</span>itens<span class="token punctuation">)</span> <span class="token operator">==</span> <span class="token number">0</span>

<span class="token keyword">def</span> <span class="token function">push</span><span class="token punctuation">(</span>self<span class="token punctuation">,</span> item<span class="token punctuation">)</span><span class="token punctuation">:</span>
    self<span class="token punctuation">.</span>itens<span class="token punctuation">.</span>append<span class="token punctuation">(</span>item<span class="token punctuation">)</span>

<span class="token keyword">def</span> <span class="token function">pop</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">if</span> <span class="token operator">not</span> self<span class="token punctuation">.</span>esta_vazia<span class="token punctuation">(</span><span class="token punctuation">)</span><span class="token punctuation">:</span>
        <span class="token keyword">return</span> self<span class="token punctuation">.</span>itens<span class="token punctuation">.</span>pop<span class="token punctuation">(</span><span class="token punctuation">)</span>

<span class="token keyword">def</span> <span class="token function">topo</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">if</span> <span class="token operator">not</span> self<span class="token punctuation">.</span>esta_vazia<span class="token punctuation">(</span><span class="token punctuation">)</span><span class="token punctuation">:</span>
        <span class="token keyword">return</span> self<span class="token punctuation">.</span>itens<span class="token punctuation">[</span><span class="token operator">-</span><span class="token number">1</span><span class="token punctuation">]</span>
pilha = Pilha() pilha.push(1) pilha

.push(2) pilha.push(3) print("Topo da pilha:", pilha.topo()) print("Removido:", pilha.pop()) print("Topo da pilha:", pilha.topo())
```

Neste exemplo, `Pilha` demonstra operações básicas de uma pilha.

Fila (Queue)

•Definição: Uma fila é uma estrutura FIFO (First In, First Out), onde o primeiro elemento inserido é o primeiro a ser removido.

•Exemplo Prático:

```
class Fila:
    def __init__(self):
        self.itens = []
<span class="token keyword">def</span> <span class="token function">esta_vazia</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">return</span> <span class="token builtin">len</span><span class="token punctuation">(</span>self<span class="token punctuation">.</span>itens<span class="token punctuation">)</span> <span class="token operator">==</span> <span class="token number">0</span>

<span class="token keyword">def</span> <span class="token function">enqueue</span><span class="token punctuation">(</span>self<span class="token punctuation">,</span> item<span class="token punctuation">)</span><span class="token punctuation">:</span>
    self<span class="token punctuation">.</span>itens<span class="token punctuation">.</span>append<span class="token punctuation">(</span>item<span class="token punctuation">)</span>

<span class="token keyword">def</span> <span class="token function">dequeue</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">if</span> <span class="token operator">not</span> self<span class="token punctuation">.</span>esta_vazia<span class="token punctuation">(</span><span class="token punctuation">)</span><span class="token punctuation">:</span>
        <span class="token keyword">return</span> self<span class="token punctuation">.</span>itens<span class="token punctuation">.</span>pop<span class="token punctuation">(</span><span class="token number">0</span><span class="token punctuation">)</span>

<span class="token keyword">def</span> <span class="token function">frente</span><span class="token punctuation">(</span>self<span class="token punctuation">)</span><span class="token punctuation">:</span>
    <span class="token keyword">if</span> <span class="token operator">not</span> self<span class="token punctuation">.</span>esta_vazia<span class="token punctuation">(</span><span class="token punctuation">)</span><span class="token punctuation">:</span>
        <span class="token keyword">return</span> self<span class="token punctuation">.</span>itens<span class="token punctuation">[</span><span class="token number">0</span><span class="token punctuation">]</span>
fila = Fila() fila.enqueue(1) fila.enqueue(2) fila.enqueue(3) print("Frente da fila:", fila.frente()) print("Removido:", fila.dequeue()) print("Frente da fila:", fila.frente())
```

Aqui, `Fila` demonstra as operações básicas de uma fila.

Árvore Binária

•Definição: Uma árvore binária é uma estrutura hierárquica onde cada nó tem, no máximo, dois filhos.

•Exemplo Prático:

```
class No:
    def __init__(self, chave):
        self.esquerda = None
        self.direita = None
```

```
        self.chave = chave
def inserir(raiz, chave):
if raiz is None:
return No(chave)
else:
if chave < raiz.chave:
raiz.esquerda = inserir(raiz.esquerda, chave)
else:
raiz.direita = inserir(raiz.direita, chave)
return raiz
def em_ordem(raiz):
if raiz:
em_ordem(raiz.esquerda)
print(raiz.chave, end=" -> ")
em_ordem(raiz.direita)
raiz = No(50)
raiz = inserir(raiz, 30)
raiz = inserir(raiz, 20)
raiz = inserir(raiz, 40)
raiz = inserir(raiz, 70)
raiz = inserir(raiz, 60)
raiz = inserir(raiz, 80)
print("Em ordem traversal:")
em_ordem(raiz)
```

Neste exemplo, uma árvore binária é criada e percorrida em ordem.

---

3. Linguagens de Programação

3.1 Tipos de Linguagens

Linguagens Orientadas a Objeto

- Java: Uma linguagem de programação popular para desenvolvimento web e mobile.
- Exemplo Prático:

```
public class ExemploJava {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Neste exemplo, um simples programa Java imprime “Hello, World!”.

C#: Utilizada principalmente na plataforma .NET.

- Exemplo Prático:

```
using System;
class ExemploCSharp {
static void Main() {
Console.WriteLine("Hello, World!");
}
}
```

Aqui, um programa C# que imprime “Hello, World!”.

Linguagens Procedurais

- C: Uma linguagem eficiente para programação de sistemas.
- Exemplo Prático:

```
#include <stdio.h>
int main() {
printf("Hello, World!\n");
return 0;
}
```

Este exemplo em C imprime “Hello, World!”.

3.2 Ferramentas e Bibliotecas

Java SE, JEE, Microprofile

- Java SE: Plataforma Java Standard Edition para aplicações desktop.
- Java EE: Plataforma Java Enterprise Edition para aplicações web.
- Microprofile: APIs para microserviços em Java.

Python Bibliotecas

- Pandas: Biblioteca para manipulação de dados.
- Exemplo Prático:

```
import pandas as pd
data = {
'Nome': ['Ana', 'Bruno', 'Carla'],
'Idade': [23, 35, 42]
}
df = pd.DataFrame(data)
print(df)
```

Este exemplo cria um DataFrame com Pandas.

- NumPy: Biblioteca para computação científica.
- Exemplo Prático:

```
import numpy as np
array = np.array([1, 2, 3, 4, 5])
print(array)
```

Aqui, um array NumPy é criado e impresso.

- SciPy: Biblioteca para matemática e ciência.
- Exemplo Prático:

```
from scipy import stats
data = [1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 10]
mean = stats.tmean(data)
print("Mean:", mean)
```

Neste exemplo, a média dos dados é calculada usando SciPy.

- Matplotlib: Biblioteca de plotagem 2D.
- Exemplo Prático:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Exemplo de Gráfico')
plt.show()
```

Este exemplo cria um gráfico simples com Matplotlib.

- TensorFlow, PyTorch: Bibliotecas para aprendizado de máquina.
- Exemplo Prático com TensorFlow:

```
import tensorflow as tf
x = tf.constant([1, 2, 3])
y = tf.constant([4, 5, 6])
result = tf.add(x, y)
print(result)
```

Aqui, TensorFlow é usado para somar dois arrays.

- Scikit-learn: Biblioteca para aprendizado de máquina e mineração de dados.
- Exemplo Prático:

```
from sklearn.linear_model import LinearRegression
import numpy as np
x = np.array([[1], [2], [3], [4], [5]])
y = np.array([1, 3, 5, 7, 9])
model = LinearRegression().fit(x, y)
print("Coeficiente:", model.coef_)
print("Intercepto:", model.intercept_)
```

Este exemplo ajusta um modelo de regressão linear usando Scikit-learn.

---

4. Desenvolvimento de Software para a Web

4.1 Tecnologias Web

Sistemas Distribuídos e Microsserviços

- •Definição: Sistemas compostos por múltiplos serviços independentes que se comunicam por meio de APIs.
- •Vantagens: Escalabilidade, flexibilidade e resiliência.
- •Exemplo Prático:

```
from flask import Flask, jsonify
app = Flask(name)
@app.route('/api/dados', methods=['GET'])
def get_dados():
dados = {'chave': 'valor'}
return jsonify(dados)
if name == 'main':
app.run(debug=True)
```

Neste exemplo, um simples serviço web é criado usando Flask, um microframework para Python.

Padrões REST, SOAP

- •REST (Representational State Transfer): Usa HTTP e CRUD (Create, Read, Update, Delete).
- •SOAP (Simple Object Access Protocol): Usa XML para troca de informações estruturadas.
- •Exemplo Prático REST:

```
from flask import Flask, jsonify, request
app = Flask(name)
dados = {}
@app.route('/api/dados/<chave>', methods=['GET'])
def get_dado(chave):
return jsonify({chave: dados.get(chave, 'Não encontrado')})
@app.route('/api/dados/<chave>', methods=['POST'])
def add_dado(chave):
valor = request.json['valor']
dados[chave] = valor
return jsonify({chave: valor})
```

```
if name == 'main':
app.run(debug=True)
```

Aqui, um serviço REST simples é criado com Flask.

HTML, XML, JSON

- •HTML (HyperText Markup Language): Linguagem de marcação usada para criar páginas web.
- •Exemplo Prático:

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo HTML</title>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

Este exemplo cria uma página HTML básica.

- •XML (eXtensible Markup Language): Linguagem de marcação usada para transportar e armazenar dados.
- •Exemplo Prático:

```
<livro>
    <titulo>Exemplo de XML</titulo>
    <autor>Autor Exemplo</autor>
</livro>
```

Este exemplo cria um documento XML simples.

- •JSON (JavaScript Object Notation): Formato leve de troca de dados.
- •Exemplo Prático:

```
{
    "nome": "Ana",
    "idade": 23
}
```

Aqui, um documento JSON simples é mostrado.

---

5. Teste de Software (Qualidade)

5.1 Controle de Qualidade

Técnicas de Teste

•**Teste
de Unidade**: Verifica a menor parte testável do software.

•Exemplo Prático:

```
import unittest
def soma(a, b):
return a + b
class TestSoma(unittest.TestCase):
def test_soma(self):
self.assertEqual(soma(1, 2), 3)
if name == 'main':
unittest.main()
```

Neste exemplo, um teste de unidade para a função `soma` é criado usando `unittest`.
Teste de Integração: Verifica a interação entre módulos.

•Exemplo Prático:

```
import unittest
class BancoDeDados:
def conectar(self):
return "Conectado ao banco de dados"
class TestBancoDeDados(unittest.TestCase):
def test_conexao(self):
bd = BancoDeDados()
self.assertEqual(bd.conectar(), "Conectado ao banco de dados")
if name == 'main':
unittest.main()
```

Aqui, um teste de integração é realizado para verificar a conexão ao banco de dados.

Teste de Sistema: Verifica o sistema completo.

•Exemplo Prático:

```
import unittest
def sistema():
return "Sistema funcionando corretamente"
class TestSistema(unittest.TestCase):
def test_sistema(self):
self.assertEqual(sistema(), "Sistema funcionando corretamente")
if name == 'main':
unittest.main()
```

Neste exemplo, um teste de sistema simples é criado.

Teste de Aceitação: Verifica se o sistema atende aos requisitos.

•Exemplo Prático:

```
import unittest
def aceitar_requisitos():
return "Requisitos aceitos"
class TestAceitacao(unittest.TestCase):
def test_aceitacao(self):
self.assertEqual(aceitar_requisitos(), "Requisitos aceitos")
if name == 'main':
unittest.main()
```

Aqui, um teste de aceitação é realizado para verificar os requisitos do sistema.

Teste de Regressão: Garante que novas mudanças não introduzam erros.

•Exemplo Prático:

```
import unittest
def funcao_antiga():
return "Funciona corretamente"
class TestRegressao(unittest.TestCase):
def test_funcao_antiga(self):
self.assertEqual(funcao_antiga(), "Funciona corretamente")
if name == 'main':
unittest.main()
```

Neste exemplo, um teste de regressão é criado para verificar uma função existente.

5.2 Automação de Testes

Selenium: Ferrenter code hereamenta para automação de navegadores web.

•Exemplo Prático:

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("http://www.google.com")
print(driver.title)
driver.quit()
```

Aqui, Selenium é usado para abrir o Google e imprimir o título da página.

Jenkins: Ferramenta de integração contínua.

•Configuração Básica:

•Instalação: Baixe e instale Jenkins em seu servidor.

•Pipeline: Crie um pipeline para executar testes automatizados.

JUnit: Framework para testes de unidade em Java.

•Exemplo Prático:

```
import static org.junit.Assert.assertEquals;
```

```
import org.junit.Test;
public class ExemploJUnit {
@Test
public void testSoma() {
assertEquals(3, 1 + 2);
}
}
```

Este exemplo mostra um teste de unidade em Java usando JUnit.

---

6. Bancos de Dados

6.1 Modelagem de Dados

Modelo Entidade-Relacionamento (ER)

- Definição: Representação gráfica de entidades e seus relacionamentos.
- Exemplo Prático:

  Imagine um modelo ER para uma biblioteca:
- Entidades: Livro, Autor, Empréstimo.
- Relacionamentos: Livro-Escrito por-Autor, Empréstimo-Contém-Livro.

6.2 SQL (Structured Query Language)

Comandos SQL

- CREATE TABLE: Cria uma nova tabela.
- Exemplo Prático:

```
CREATE TABLE Alunos (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100),
    Idade INT
);
```

Este comando cria uma tabela `Alunos` com colunas `ID`, `Nome` e `Idade`.

- INSERT INTO: Insere novos registros.
- Exemplo Prático:

```
INSERT INTO Alunos (ID, Nome, Idade) VALUES (1, 'Ana', 23);
```

Este comando insere um novo registro na tabela `Alunos`.

- SELECT: Consulta dados.
- Exemplo Prático:

```
SELECT * FROM Alunos;
```

Este comando seleciona todos os registros da tabela `Alunos`.

- UPDATE: Atualiza registros existentes.

•Exemplo Prático:

```
UPDATE Alunos SET Idade = 24 WHERE ID = 1;
```

Este comando atualiza a idade do aluno com `ID` 1 para 24.

•DELETE: Remove registros.

•Exemplo Prático:

```
DELETE FROM Alunos WHERE ID = 1;
```

Este comando remove o registro do aluno com `ID` 1.

6.3 NoSQL

MongoDB

•Definição: Um banco de dados NoSQL orientado a documentos.

•Exemplo Prático:

```
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['biblioteca']
colecao = db['livros']
livro = {"titulo": "Exemplo de MongoDB", "autor": "Autor Exemplo"}
colecao.insert_one(livro)
```

Neste exemplo, um documento é inserido em uma coleção MongoDB.

Redis

•Definição: Um banco de dados em memória, chave-valor.

•Exemplo Prático:

```
import redis
r = redis.Redis(host='localhost', port=6379, db=0)
r.set('chave', 'valor')
print(r.get('chave'))
```

Aqui, um valor é armazenado e recuperado usando Redis.

Cassandra

•Definição: Um banco de dados distribuído, orientado a colunas.

•Exemplo Prático:

```
from cassandra.cluster import Cluster
cluster = Cluster(['127.0.0.1'])
session = cluster.connect()
session.execute("CREATE KEYSPACE IF NOT EXISTS exemplo WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}")
session.execute("CREATE TABLE IF NOT EXISTS exemplo.livros (id UUID PRIMARY KEY, titulo
text, autor text)")
```

Este exemplo cria um keyspace e uma tabela em Cassandra.

---

7. Agilidade

7.1 Métodos Ágeis

Scrum

- •Definição: Um framework ágil para gerenciamento de projetos.
- •Papéis: Product Owner, Scrum Master, Time de Desenvolvimento.
- •Eventos: Sprint, Planejamento da Sprint, Daily Scrum, Revisão da Sprint, Retrospectiva da Sprint.
- •Artefatos: Product Backlog, Sprint Backlog, Incremento.

Exemplo Prático:
Imagine um projeto de desenvolvimento de um site. O Product Owner prioriza as funcionalidades no Product Backlog. Durante o Planejamento da Sprint, o time escolhe as tarefas para a Sprint. Diariamente, o time se reúne para a Daily Scrum, discutindo o progresso e os impedimentos. No final da Sprint, o time apresenta o Incremento na Revisão da Sprint e discute melhorias na Retrospectiva.

Kanban

- •Definição: Método ágil para gerenciar o trabalho com foco na visualização do fluxo de trabalho.
- •Componentes: Quadro Kanban, Cartões Kanban, Colunas (A Fazer, Em Progresso, Concluído).

Exemplo Prático:
Imagine um quadro Kanban para o mesmo projeto de desenvolvimento de um site. As tarefas são movidas entre as colunas, do “A Fazer” para “Em Progresso” e finalmente para “Concluído”, visualizando o fluxo de trabalho e identificando gargalos.

---

8. Organização e Arquitetura de Computadores

8.1 Arquitetura de Computadores

Conceitos Básicos

- •CPU (Unidade Central de Processamento): O cérebro do computador, responsável por executar instruções.

- Memória RAM (Random Access Memory): Memória de acesso rápido usada para armazenar dados temporários.
- Memória ROM (Read-Only Memory): Memória de acesso somente leitura, usada para armazenar firmware.
- Armazenamento: Dispositivos como HDDs e SSDs usados para armazenar dados permanentemente.

Componentes de um Computador

- Processador: Realiza operações aritméticas e lógicas.
- Placa-mãe: Conecta todos os componentes do computador.
- Memória: Armazena dados e instruções temporariamente.
- Dispositivos de Entrada/Saída: Incluem teclado, mouse, monitor, impressora.

Exemplo Prático:
Imagine montar um computador. Você precisa de um processador para executar as instruções, memória RAM para armazenar dados temporários, uma placa-mãe para conectar todos os componentes e dispositivos de entrada/saída para interagir com o computador.

**8.

2 Sistemas de Numeração**

Binário, Decimal, Hexadecimal

- Binário: Base 2, usa 0 e 1.
- Decimal: Base 10, usa 0-9.
- Hexadecimal: Base 16, usa 0-9 e A-F.

Conversões

- Binário para Decimal:
  - Exemplo: `1010` (binário) = `10` (decimal)
- Decimal para Binário:
  - Exemplo: `10` (decimal) = `1010` (binário)
- Decimal para Hexadecimal:
  - Exemplo: `255` (decimal) = `FF` (hexadecimal)
- Hexadecimal para Decimal:
  - Exemplo: `FF` (hexadecimal) = `255` (decimal)

Exemplo Prático:

```
def binario_para_decimal(binario):
    return int(binario, 2)
def decimal_para_binario(decimal):
```

```
    return bin(decimal)[2:]
print(binario_para_decimal('1010'))  # Saída: 10
print(decimal_para_binario(10))      # Saída: 1010
```

Neste exemplo, funções Python convertem entre binário e decimal.

8.3 Processadores e Memória

Ciclo de Instrução

- Fase de Busca: A CPU busca a próxima instrução da memória.
- Fase de Decodificação: A CPU decodifica a instrução.
- Fase de Execução: A CPU executa a instrução.

Exemplo Prático:
Imagine a CPU executando um programa. Ela busca a instrução `ADD A, B` da memória, decodifica para entender que precisa somar os valores nos registradores `A` e `B`, e executa a soma, armazenando o resultado em `A`.

Conclusão

Este guia fornece uma visão abrangente dos tópicos abordados na Prova de Avaliação de Conhecimentos para o Técnico Superior Profissional de Informática no concurso público em Portugal. Com exemplos práticos, esperamos que ajude no estudo e preparação para a prova. Boa sorte!