**Resumo da Apresentação: API e Semantic Kernel Básico**

A apresentação tem como objetivo principal detalhar a implementação e uso das **APIs do Azure OpenAI** e introduzir o **Semantic Kernel** como um SDK (Software Development Kit) ou middleware para orquestrar o uso de modelos de IA, incluindo a noção de Agentes de IA.

**1. Uso da API do Azure OpenAI:**

* A API suporta diversas funcionalidades dos modelos: **Chat** (interações conversacionais), **Completar** (geração de texto baseada em prompt), **Imagens** (geração com DALL-E) e **Áudio** (Text-to-Speech).
* Demonstra como instanciar o cliente da API (exemplo em Python) utilizando credenciais essenciais como azure\_endpoint, api\_key e api\_version, idealmente carregadas de variáveis de ambiente seguras.
* Apresenta exemplos de chamadas específicas para:
  + **Completar:** Usando o endpoint completions com parâmetros como prompt, max\_tokens, temperature.
  + **Chat:** Utilizando client.chat.completions.create com model, messages e temperature.
  + **Imagem:** Empregando client.images.generate com model, prompt, n (número de imagens) e size.
* Reitera as **variáveis/parâmetros comuns** que controlam a geração: Model ID, Temperature, Max Tokens, Top P, e Penalidades (Presence/Frequency).
* Menciona a importância do **armazenamento seguro** de credenciais (Env Segura) e do monitoramento através do **Azure Monitor** para acompanhar o uso e performance da API.

**2. Introdução ao Semantic Kernel (SK):**

* O Semantic Kernel é apresentado como um **Middleware de IA** ou SDK que facilita a integração de modelos de IA (como os do Azure OpenAI, Hugging Face, etc.) em aplicações.
* Seu propósito é **abstrair a complexidade** das chamadas diretas à API, orquestrar fluxos, adicionar funcionalidades extras, automatizar processos e habilitar a criação de **Agentes de IA** mais sofisticados.
* A **arquitetura básica** do SK inclui:
  + **Kernel:** O núcleo orquestrador que gerencia a execução, seleciona serviços de IA, lida com templates de prompt, invoca funções, gerencia a memória e retorna resultados.
  + **Skills / Functions / Plugins:** Representam as capacidades que o Kernel pode invocar. Podem ser funções de código nativo (C#, Python) ou funções semânticas (prompts parametrizados).
  + **Memory:** Permite ao Kernel acessar dados para fornecer contexto às chamadas de IA, frequentemente utilizando **Vector Stores** para busca rápida de informações relevantes.
  + **Hooks / Filtros:** Mecanismos para interceptar e modificar o fluxo de execução (ex: validação, logging).
* Diagramas ilustram como o Kernel interage com a aplicação, modelos de IA, plugins e memória.
* A apresentação finaliza indicando uma **demonstração prática** ("Demo do SK") para explorar esses conceitos.

Em suma, a apresentação fornece uma visão geral de como interagir programaticamente com o Azure OpenAI via API e introduz o Semantic Kernel como uma ferramenta poderosa para construir aplicações de IA mais complexas e orquestradas, abstraindo chamadas diretas e adicionando capacidades como planejamento e memória.