1. 💡 Visão Geral do Ecossistema "SkillSync"

O "SkillSync" é um ecossistema de aplicações desenvolvido para a Global Solution, focado no tema **"O Futuro do Trabalho"**. A solução aborda diretamente a ascensão da "gig economy" (economia de projetos) e a necessidade de requalificação profissional (reskilling).

A arquitetura é composta por quatro componentes principais que se integram para criar uma plataforma coesa:

1. **API Principal (.NET):** O "coração" do sistema. É responsável por gerenciar usuários (login/cadastro), o CRUD de perfis (freelancers) e projetos (contratantes), e orquestrar a comunicação entre o banco de dados e os outros microserviços.
2. **Aplicativo Mobile (React Native):** A interface do usuário final. Consome *apenas* a API .NET para todas as suas operações.
3. **Portal Admin (Java WebApp):** Uma aplicação web *separada* onde administradores usam IA (Spring AI) para gerar "Dicas de Carreira" e salvá-las no banco de dados.
4. **Microserviço de IA (Python/FastAPI):** O "cérebro" de matchmaking e o foco *desta disciplina*. É uma API independente que recebe dados de projetos e perfis, e retorna uma análise de compatibilidade.

Este documento detalha o desenvolvimento do **Microserviço de IA (item 4)**.

---

2. 🎯 O Papel do Microserviço de IA (Foco desta Matéria)

O componente desenvolvido para esta disciplina é o **"SkillSync AI Matchmaking API"**.

Este serviço atua como o "cérebro" de *matchmaking* para a plataforma. O funcionamento é o seguinte:

1. A API recebe uma requisição (`POST /gerar-match`) contendo os dados de um **projeto** (enviado pela API .NET) e uma **lista de perfis** (também enviada pelo .NET).
2. Ela utiliza um modelo de IA Generativa (Google Gemini) para analisar e comparar o projeto com cada perfil da lista.
3. Ela então retorna um JSON estruturado contendo um *score* de compatibilidade (0-100) e uma justificativa de RH para cada *match*, ordenados do mais relevante para o menos relevante.

---

3. 🌍 Alinhamento com o Tema da Global Solution

O tema da Global Solution é **"O Futuro do Trabalho"**. Esta solução se alinha diretamente a este desafio de várias formas:

- **Foco na "Gig Economy":** O projeto aborda diretamente a ascensão do trabalho flexível, da "gig economy" e das plataformas de freelancers, que são pilares do futuro do trabalho.
- **IA como Parceira (Não Substituta):** A solução utiliza a "IA como parceira do ser humano", um dos temas sugeridos. A IA não substitui o recrutador; ela o *auxilia*, realizando a análise inicial de centenas de perfis e fornecendo uma lista filtrada e justificada, otimizando o processo de contratação.
- **Novas Formas de Inclusão Produtiva:** Ao usar tecnologia para conectar talentos (freelancers) a oportunidades (projetos) de forma eficiente, a plataforma promove novas formas de inclusão produtiva.

---

4. 📋 Cumprimento dos Requisitos da Disciplina

O projeto foi estruturado para cumprir 100% dos critérios de avaliação da matéria "Disruptive Architectures: IoT, IoB & Generative IA":

- **1. Aplicação de IA Generativa (Requisito Técnico):**
  - A solução é uma API REST desenvolvida em Python/FastAPI que implementa um modelo de IA Generativa de ponta, o **Google Gemini** (especificamente o `gemini-2.5-flash`), para realizar suas tarefas.
- **2. Geração de Texto e Prompt Engineering (Requisito Técnico):**
  - O núcleo do projeto é a **Engenharia de Prompt**. Foi criado um *prompt* robusto e detalhado, estruturado em quatro camadas (Contexto, Instruções de Análise, Sistema de Pontuação e Formato de Resposta).
  - Este *prompt* instrui a IA a atuar como um especialista em RH e a **gerar um texto** em um formato JSON estritamente controlado (`response_mime_type`), contendo o `id_perfil`, `score_compatibilidade` e `justificativa`.
- **3. Integração Efetiva com Outras Disciplinas (Critério de Avaliação):**
  - Este é um pilar central do projeto, desenhado para evitar a penalidade de -40 pontos por falta de integração.
  - A arquitetura (documentada no `README.md`) define que este microserviço de IA é "puro" (não se conecta ao banco de dados).
  - Ele é projetado para ser consumido pela **API .NET** (outra disciplina). A API .NET será responsável por buscar os dados no Oracle, montar o JSON de requisição, chamar este *endpoint* (`POST /gerar-match`) e, então, repassar a resposta da IA para o App Mobile.
- **4. Boas Práticas de Código e Documentação (Critério de Avaliação):**
  - **Código:** O `main.py` utiliza `Pydantic` (BaseModel) para validação rigorosa dos dados de entrada e saída, `python-dotenv` para gerenciamento seguro de chaves de API e `ThreadPoolExecutor` para processamento assíncrono de requisições.
  - **Documentação:** O `README.md` é completo, explicando a arquitetura, o *prompt*, as tecnologias, e como instalar e rodar o projeto.

    - **Deploy:** A API foi implantada na nuvem (Render.com) e está acessível publicamente, com um link para a documentação interativa (Swagger).
- **5. Apresentação e Demonstração Funcional:**
    - O projeto está 100% funcional. O script `test_api.py` foi criado para validar o funcionamento dos *endpoints* `/health` e `/gerar-match`, provando que a IA processa os dados e retorna os *matches* esperados.