

Faculdade de Informática e Administração Paulista

FIAP Paulista

Henrique Martins Oliveira – RM563620

Henrique Texeira Cesar – RM563088

Symbio – Plataforma de Requalificação por IA

Global Solution – Java

São Paulo

2025

SUMÁRIO

OBJETIVO E ESCOPO DO PROJETO.....	2
BREVE DESCRIÇÃO DAS FUNCIONALIDADES.....	3
TABELAS ENDPOINTS	3
PROTÓTIPO – PRINTS DAS TELAS.....	4
MODELO DE ENTIDADE-RELACIONAMENTO	8
DIAGRAMA DE CLASSES.....	8

OBJETIVO E ESCOPO DO PROJETO

Objetivo: A SYMBIO é uma plataforma B2B de inteligência corporativa focada em Mobilidade Interna e Requalificação. O objetivo deste componente de backend, desenvolvido em Java com Quarkus, é servir como o núcleo de processamento lógico, orquestrando a comunicação entre o banco de dados Oracle, o frontend e a API de Inteligência Artificial.

Escopo: O escopo desta API é fornecer endpoints RESTful seguros e performáticos para:

1. Gerenciar o catálogo de Cargos, Vagas, Skills e Colaboradores.
2. Integrar-se a um modelo de IA para analisar e salvar o risco de automação de cada cargo.
3. Executar a lógica de "Match" (Análise de Lacunas), calculando a compatibilidade percentual entre um colaborador e uma vaga interna.

BREVE DESCRIÇÃO DAS FUNCIONALIDADES

A solução Java foi totalmente implementada e hospedada na nuvem (Render), com as seguintes funcionalidades principais em operação:

- **Cadastro Inteligente de Cargos:** O endpoint POST /cargos não apenas salva um novo cargo, mas primeiramente consulta a API de IA em Python (<https://symbio-api-ia.onrender.com/prever/risco>) para calcular o risco de automação (Baixo, Médio, Alto). Esse risco é então persistido no banco de dados Oracle, enriquecendo o dado.
- **Algoritmo de Match (A Alma do Projeto):** O endpoint GET /match/{idColab}/{idVaga} executa a lógica de negócio principal. Ele compara as skills do colaborador (com seus níveis de proficiência 1-5) contra os requisitos da vaga (com seus pesos de importância 1-10), aplicando um cálculo ponderado.
- **API RESTful Completa:** Fornece todos os endpoints CRUD (Create, Read, Update, Delete) necessários para o frontend gerenciar Colaboradores, Vagas e Skills, seguindo as melhores práticas REST.
- **Conectividade com Oracle:** A camada DAO utiliza JDBC puro para comunicação direta com o banco de dados Oracle da FIAP, garantindo total controle sobre as transações.

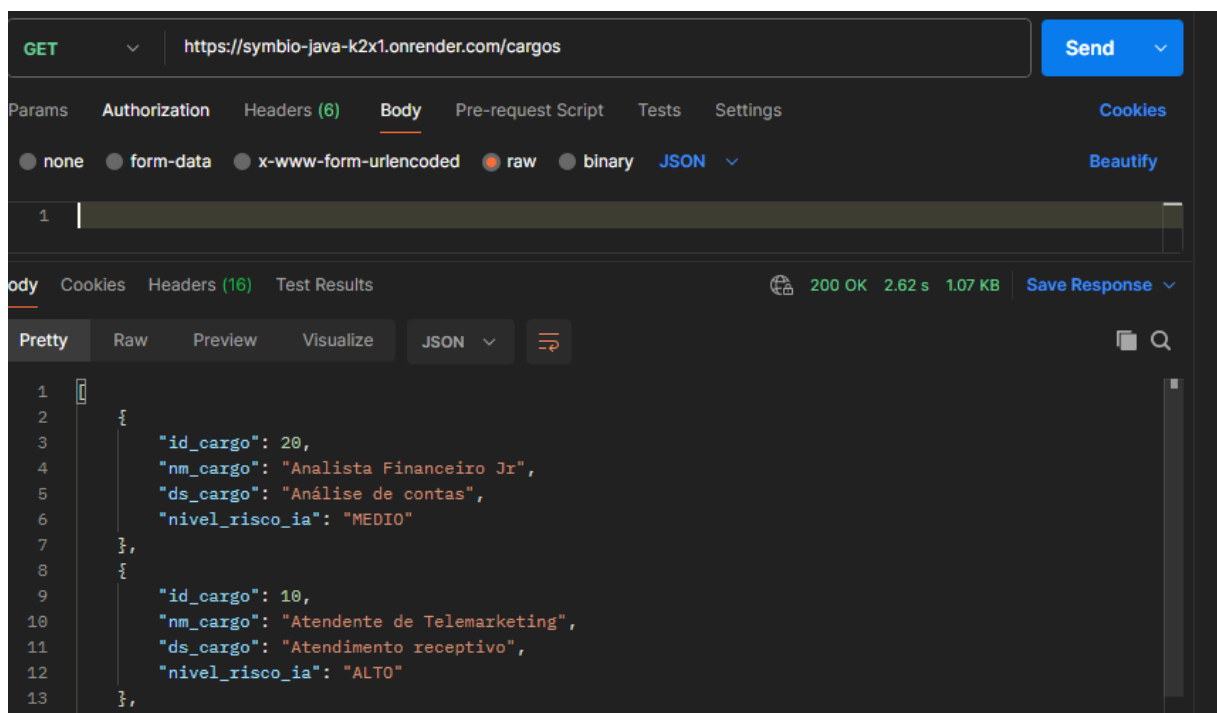
TABELAS ENDPOINTS

Verbo	URI (Caminho)	Descrição	Status (Sucesso)
GET	/hello	Endpoint de <i>Health Check</i> (teste).	200 OK
POST	/cargos	Cadastra novo cargo, consultando a IA.	201 Created
GET	/cargos	Lista todos os cargos.	200 OK
GET	/match/{idColab}/{idVaga}	Calcula o Match.	200 OK
GET	/colaboradores	Lista todos os colaboradores.	200 OK
GET	/colaboradores/{id}	Busca um colaborador por ID.	200 OK
POST	/colaboradores	Cadastra um novo colaborador.	201 Created

PUT	/colaboradores/{id}	Atualiza um colaborador.	200 OK
DELETE	/colaboradores/{id}	Deleta um colaborador.	204 No Content
GET	/vagas	Lista todas as vagas.	200 OK
POST	/vagas	Cadastra uma nova vaga.	201 Created
GET	/skills	Lista todas as skills.	200 OK
POST	/skills	Cadastra uma nova skill.	201 Created

PROTÓTIPO – PRINTS DAS TELAS

Listar Cargos - GET



GET <https://symbio-java-k2x1.onrender.com/cargos> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

1

body Cookies Headers (16) Test Results 200 OK 2.62 s 1.07 KB Save Response

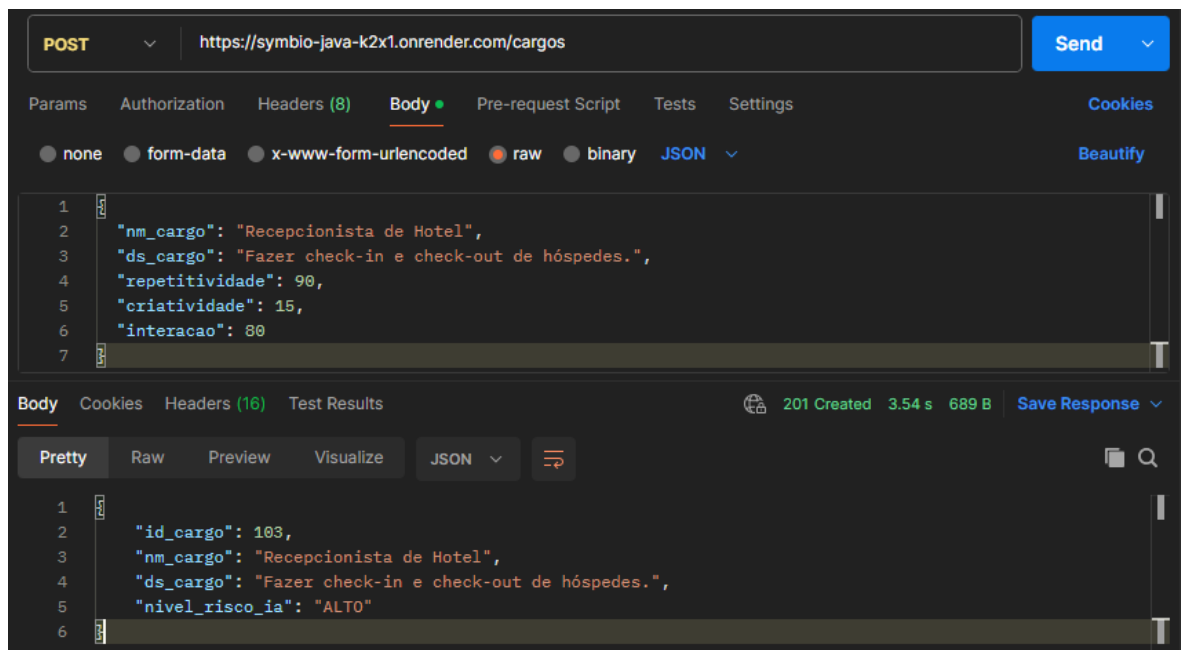
Pretty Raw Preview Visualize JSON

```

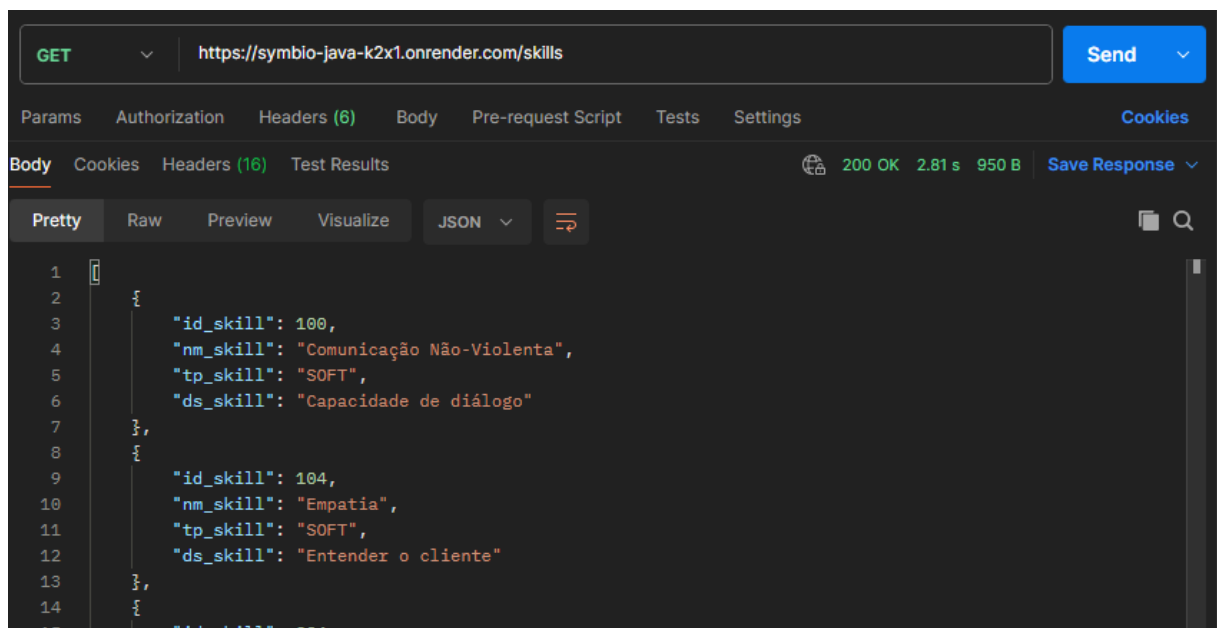
1  {
2    {
3      "id_cargo": 20,
4      "nm_cargo": "Analista Financeiro Jr",
5      "ds_cargo": "Análise de contas",
6      "nivel_risco_ia": "MEDIO"
7    },
8    {
9      "id_cargo": 10,
10     "nm_cargo": "Atendente de Telemarketing",
11     "ds_cargo": "Atendimento receptivo",
12     "nivel_risco_ia": "ALTO"
13   },
14 }

```

Cadastrar Cargo - POST



Listar Skills - GET



Cadastrar Skill – POST

POST <https://symbio-java-k2x1.onrender.com/skills> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON Beautify

```
1
2  "nm_skill": "Liderança de Equipe",
3  "tp_skill": "SOFT",
4  "ds_skill": "Capacidade de liderar times multidisciplinares"
5
```

Body Cookies Headers (16) Test Results 201 Created 2.62 s 688 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2  "id_skill": 300,
3  "nm_skill": "Liderança de Equipe",
4  "tp_skill": "SOFT",
5  "ds_skill": "Capacidade de liderar times multidisciplinares"
6
```

Buscar por ID

GET <https://symbio-java-k2x1.onrender.com/colaboradores/1>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON

```
1
2  "nm_skill": "Liderança de Equipe",
3  "tp_skill": "SOFT",
4  "ds_skill": "Capacidade de liderar times multidisciplinares"
5
```

Body Cookies Headers (16) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1
2  "id_colaborador": 1,
3  "nm_colaborador": "João Silva",
4  "ds_email": "joao.silva@symbio.com",
5  "dt_admissao": "2020-01-15",
6  "salario": 2500.0,
7  "id_cargo": 10
8
```

Atualizar Colaboradores – PUT

The screenshot displays a REST client interface with a PUT request to the URL `https://symbio-java-k2x1.onrender.com/colaboradores/1`. The request body is a JSON object with the following fields: `nm_colaborador`, `ds_email`, `dt_admissao`, `salario`, and `id_cargo`. Below the request, the response body is shown in a 'Pretty' JSON format, which includes an additional `id_colaborador` field.

Request:

```
PUT https://symbio-java-k2x1.onrender.com/colaboradores/1

{
  "nm_colaborador": "João Silva (Salário Atualizado)",
  "ds_email": "joao.silva@symbio.com",
  "dt_admissao": "2020-01-15",
  "salario": 2800.00,
  "id_cargo": 10
}
```

Response:

```
{
  "id_colaborador": 1,
  "nm_colaborador": "João Silva (Salário Atualizado)",
  "ds_email": "joao.silva@symbio.com",
  "dt_admissao": "2020-01-15",
  "salario": 2800.0,
  "id_cargo": 10
}
```

DELETE

MODELO DE ENTIDADE-RELACIONAMENTO

T_SYM_CARGO	
PK	<u>CONSTRAINT CK_SYM_RISCO_IA_CHECK</u> (nivel_risco, ia)
PK	<u>CONSTRAINT UK_SYM_NM_CARGO_UNIQUE</u> (nm_cargo)
PK	<u>CONSTRAINT PK_SYM_CARGO</u> (id_cargo)
– Atributos id_cargo NUMBER(5) GENERATED BY DEFAULT AS IDENTITY nm_cargo VARCHAR(50) NOT NULL ds_cargo VARCHAR(100) nivel_risco_ia VARCHAR(10) NOT NULL – Restrições	

T_SYM_SKILL	
PK	<u>CONSTRAINT CK_SYM_TP_SKILL_CHECK</u> (tp_servico, skill)
PK	<u>CONSTRAINT UK_SYM_NM_SKILL_UNIQUE</u> (nm_skill)
PK	<u>CONSTRAINT PK_SYM_SKILL</u> (id_skill)
– Atributos id_skill NUMBER(5) GENERATED BY DEFAULT AS IDENTITY nm_skill VARCHAR(50) NOT NULL tp_servico VARCHAR(10) NOT NULL ds_skill VARCHAR(100) – Restrições	

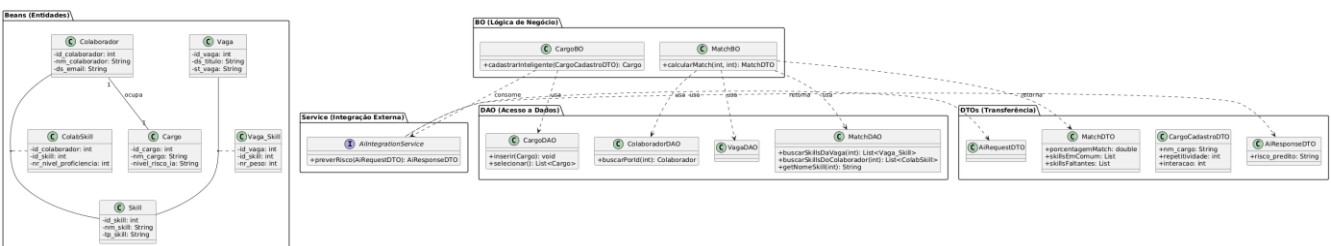
T_SYM_COLABORADOR	
PK	<u>CONSTRAINT FK_SYM_COLAB_CARGO_FOREIGN</u> (id_cargo, id_colaborador)
PK	<u>CONSTRAINT UK_SYM_EMAIL_COLAB_UNIQUE</u> (email)
PK	<u>CONSTRAINT PK_SYM_COLABORADOR</u> (id_colaborador)
– Atributos id_colaborador NUMBER(10) GENERATED BY DEFAULT AS IDENTITY nm_colaborador VARCHAR(100) NOT NULL ds_email VARCHAR(100) NOT NULL dt_admissao DATE NOT NULL salario NUMBER(10,2) NOT NULL id_cargo NUMBER(5) NOT NULL – Restrições	

T_SYM_VAGA	
PK	<u>CONSTRAINT CK_SYM_STATUS_VAGA_CHECK</u> (status_vaga)
PK	<u>CONSTRAINT PK_SYM_VAGA</u> (id_vaga)
id_vaga NUMBER(5) GENERATED BY DEFAULT AS IDENTITY ds_titulo VARCHAR2(100) NOT NULL dt_abertura DATE NOT NULL st_vaga VARCHAR2(10) NOT NULL	

T_SYM_COLAB_SKILL	
PK	<u>CONSTRAINT CK_SYM_NIVEL_SKILL_CHECK</u> (nr_nivel, id_colaborador, id_skill)
PK	<u>CONSTRAINT FK_SYM_CS_SKILL_FOREIGN_KEY</u> (id_colaborador, id_skill)
PK	<u>CONSTRAINT FK_SYM_CS_COLAB_FOREIGN_KEY</u> (id_colaborador, nr_nivel)
PK	<u>CONSTRAINT PK_SYM_COLAB_SKILL</u> (id_colaborador, id_skill, nr_nivel)
id_colaborador NUMBER(10) NOT NULL id_skill NUMBER(5) NOT NULL nr_nivel_proficiencia NUMBER(1) NOT NULL	

T_SYM_VAGA_SKILL	
PK	<u>CONSTRAINT CK_SYM_PESO_SKILL_CHECK</u> (nr_nivel, id_vaga, id_skill)
PK	<u>CONSTRAINT FK_SYM_VS_SKILL_FOREIGN_KEY</u> (id_vaga, id_skill)
PK	<u>CONSTRAINT FK_SYM_VS_VAGA_FOREIGN_KEY</u> (id_vaga, nr_peso)
PK	<u>CONSTRAINT PK_SYM_VAGA_SKILL</u> (id_vaga, id_skill, nr_peso)
id_vaga NUMBER(5) NOT NULL id_skill NUMBER(5) NOT NULL nr_peso NUMBER(2) NOT NULL	

DIAGRAMA DE CLASSES



COMO UTILIZAR

API Java (Quarkus): <https://symbio-java-k2x1.onrender.com>

API de IA (Python): <https://symbio-api-ia.onrender.com>

Vídeo de Demonstração: <https://www.youtube.com/watch?v=dFaqvPYaIJc>

Vídeo Pitch: https://youtu.be/mT2-jy_XQQ0

GITHUB: <https://github.com/Symbio-Global-Solution/symbio-java.git>

COLD START

Ambas as APIs "dormem" após 15 minutos sem uso.

A API Java (Passo 2) precisa chamar a API de IA (Passo 1). Se a API de IA estiver "dormindo", a API Java (que acorda mais rápido) não conseguirá obter o risco de IA a tempo e irá salvar o valor de fallback "ANALISE_PENDENTE" no banco de dados.

Para demonstrar o fluxo completo (onde a IA retorna "ALTO", "MEDIO" ou "BAIXO"), **você DEVE "aquecer" a API de IA (Python) PRIMEIRO.**

PASSO A PASSO PARA TESTAR

Use o Postman ou o cliente HTTP do IntelliJ.

Passo 1: "Acordar" a API de IA (Python)

Primeiro, envie uma requisição direta para a API de IA para forçá-la a "acordar".

Método: POST

URL: <https://symbio-api-ia.onrender.com/prever/risco>

Body (raw, JSON):

```
{ "features": [90, 10, 30] }
```

Ação: Clique em "Send" e **aguarde**. Esta primeira chamada pode demorar de **30 a 60 segundos**. Você saberá que funcionou quando receber a resposta {"risco_predito": "ALTO"}.

Passo 2: "Acordar" a API Java (Quarkus)

Agora que a IA está acordada, acorde o Java.

Método: GET

URL: <https://symbio-java-k2x1.onrender.com/hello>

Ação: Clique em "Send". Esta chamada também pode demorar alguns segundos.

Resultado: Você receberá o JSON {"status": "online", ...}.

Pronto! Ambas as APIs estão "quentes" e prontas para se comunicar.