

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 negócios 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.

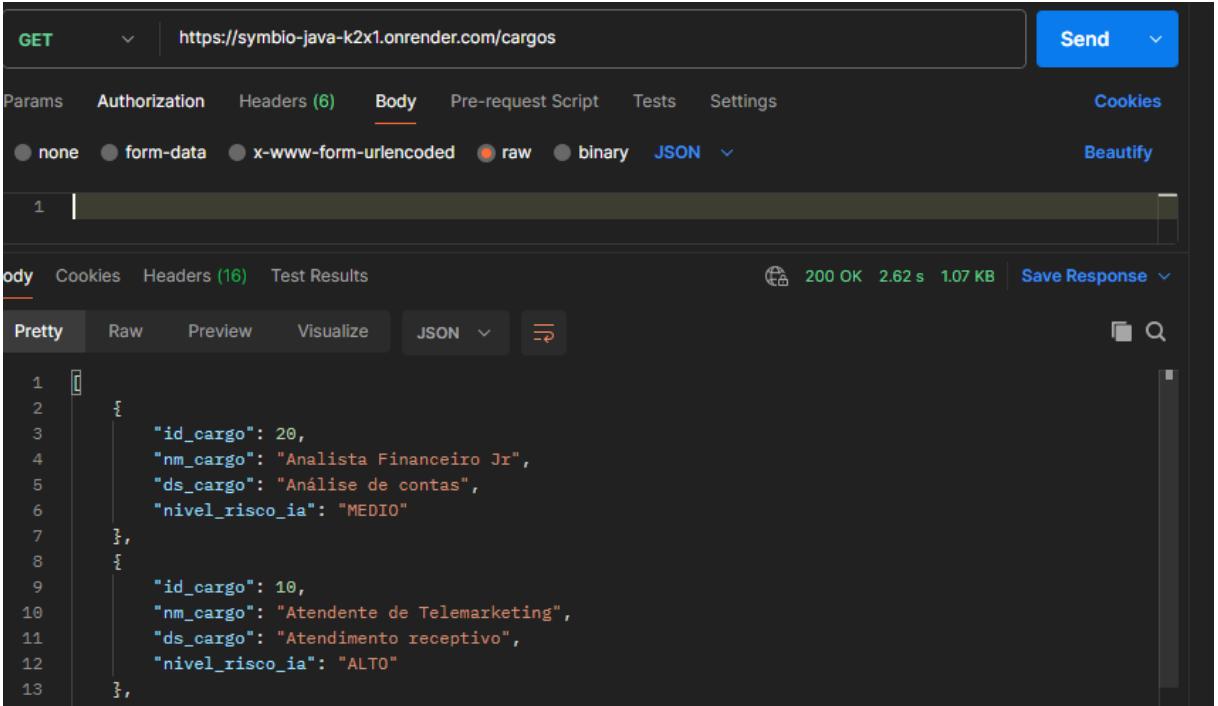
TABELAS ENDPOINTS

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

Body x-www-form-urlencoded raw binary JSON

```

1 [
  {
    "id_cargo": 20,
    "nm_cargo": "Analista Financeiro Jr",
    "ds_cargo": "Análise de contas",
    "nivel_risco_ia": "MÉDIO"
  },
  {
    "id_cargo": 10,
    "nm_cargo": "Atendente de Telemarketing",
    "ds_cargo": "Atendimento receptivo",
    "nivel_risco_ia": "ALTO"
  }
]
  
```

200 OK 2.62 s 1.07 KB Save Response

Pretty Raw Preview Visualize JSON

Cadastrar Cargo - POST

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

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

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

```
1 {  
2   "nm_cargo": "Recepção de Hotel",  
3   "ds_cargo": "Fazer check-in e check-out de hóspedes.",  
4   "repetitividade": 90,  
5   "criatividade": 15,  
6   "interacao": 80  
7 }
```

Body Cookies Headers (16) Test Results 201 Created 3.54 s 689 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id_cargo": 103,  
3   "nm_cargo": "Recepção de Hotel",  
4   "ds_cargo": "Fazer check-in e check-out de hóspedes.",  
5   "nivel_risco_ia": "ALTO"  
6 }
```

Listar Skills - GET

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

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

Body Cookies Headers (16) Test Results 200 OK 2.81 s 950 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [ {  
2   "id_skill": 100,  
3   "nm_skill": "Comunicação Não-Violenta",  
4   "tp_skill": "SOFT",  
5   "ds_skill": "Capacidade de diálogo"  
6 },  
7 {  
8   "id_skill": 104,  
9   "nm_skill": "Empatia",  
10  "tp_skill": "SOFT",  
11  "ds_skill": "Entender o cliente"  
12 },  
13 {  
14   "id_skill": 104  
15 }
```

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

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

Pretty Raw Preview Visualize JSON

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

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

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

Pretty Raw Preview Visualize JSON

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

Atualizar Colaboradores – PUT

The screenshot shows the Postman application interface for making a PUT request to update a collaborator. The URL is set to `https://symbio-java-k2x1.onrender.com/colaboradores/1`. The request method is selected as `PUT`. The `Body` tab is active, showing the JSON payload for the update:

```
1 {  
2     "nm_colaborador": "João Silva (Salário Atualizado)",  
3     "ds_email": "joao.silva@symbio.com",  
4     "dt_admissao": "2020-01-15",  
5     "salario": 2800.00,  
6     "id_cargo": 10  
7 }
```

Below the body, the `Test Results` tab is visible. The `Body` tab is also present at the bottom of the interface.

At the top of the main area, there are tabs for `Params`, `Authorization`, `Headers (8)`, `Body`, `Pre-request Script`, `Tests`, and `Se`. Below these tabs, there are radio buttons for `none`, `form-data`, `x-www-form-urlencoded`, `raw` (which is selected), `binary`, and `JSON`.

At the bottom of the interface, there are buttons for `Pretty`, `Raw`, `Preview`, `Visualize`, and `JSON` (with a dropdown arrow). There is also a copy icon (`C`) and a refresh icon (`R`).

```
1 {  
2     "id_colaborador": 1,  
3     "nm_colaborador": "João Silva (Salário Atualizado)",  
4     "ds_email": "joao.silva@symbio.com",  
5     "dt_admissao": "2020-01-15",  
6     "salario": 2800.0,  
7     "id_cargo": 10  
8 }
```

DELETE

DELETE https://symbio-java-k2x1.onrender.com/colaboradores/4

Send

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

```

1 "nm_colaborador": "João Silva (Salário Atualizado)",
2 "ds_email": "joao.silva@symbio.com",
3 "dt_admissao": "2020-01-15",
4 "salario": 2800.00,
5 "id_cargo": 10
6
7
  
```

Body Cookies Headers (12) Test Results

Pretty Raw Preview Visualize Text

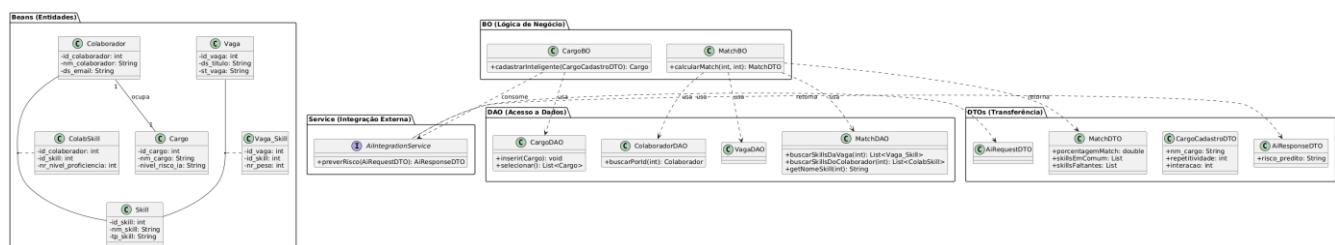
204 No Content 2.38 s 471 B Save Response

1

MODELO DE ENTIDADE-RELACIONAMENTO

T_SYM_CARGO	T_SYM_COLABORADOR	T_SYM_COLAB_SKILL
PK CONSTRAINT CK_SYM_RISCO_IA CHECK(nivel)	PK CONSTRAINT FK_SYM_COLAB_CARGO FOREIGN	PK CONSTRAINT CK_SYM_NIVEL_SKILL CHECK(nr_nivel)
PK CONSTRAINT UK_SYM_NM_CARGO UNIQUE (nm)	PK CONSTRAINT UK_SYM_EMAIL_COLAB UNIQUE (ds_email)	PK CONSTRAINT FK_SYM_CS_SKILL_FOREIGN KEY
PK CONSTRAINT PK_SYM_CARGO (id_cargo)	PK CONSTRAINT PK_SYM_COLABORADOR (id_colaborador)	PK CONSTRAINT FK_SYM_CS_COLAB FOREIGN KEY
– Atributos	– Atributos	– Atributos
id_cargo NUMBER(5) GENERATED BY DEFAULT AS	id_colaborador NUMBER(10) GENERATED BY DEF	id_colaborador NUMBER(10) NOT NULL
nm_cargo VARCHAR(50) NOT NULL	nm_colaborador VARCHAR(100) NOT NULL	id_skill NUMBER(5) NOT NULL
ds_cargo VARCHAR(100)	ds_email VARCHAR(100) NOT NULL	nr_nivel_proficiencia NUMBER(1) NOT NULL
nivel_risco_ia VARCHAR(10) NOT NULL	dt_admissao DATE NOT NULL	
– Restrições	salario NUMBER(10,2) NOT NULL	
	id_cargo NUMBER(5) NOT NULL	
	– Restrições	
T_SYM_SKILL	T_SYM_VAGA	T_SYM_VAGA_SKILL
PK CONSTRAINT CK_SYM_TP_SKILL CHECK(tp_skill)	PK CONSTRAINT CK_SYM_STATUS_VAGA CHECK(st_vaga)	PK CONSTRAINT CK_SYM_PESO_SKILL_CHECK(nr_peso)
PK CONSTRAINT UK_SYM_NM_SKILL UNIQUE (nm)	PK CONSTRAINT PK_SYM_VAGA (id_vaga)	PK CONSTRAINT FK_SYM_VS_SKILL_FOREIGN KEY
PK CONSTRAINT PK_SYM_SKILL (id_skill)	id_vaga NUMBER(5) GENERATED BY DEFAULT AS	PK CONSTRAINT FK_SYM_VS_VAGA_FOREIGN KEY
– Atributos	ds_titulo VARCHAR2(100) NOT NULL	id_vaga NUMBER(5) NOT NULL
id_skill NUMBER(5) GENERATED BY DEFAULT AS	dt_abertura DATE NOT NULL	id_skill NUMBER(5) NOT NULL
nm_skill VARCHAR(50) NOT NULL	st_vaga VARCHAR2(10) NOT NULL	nr_peso NUMBER(2) NOT NULL
tp_skill VARCHAR(10) NOT NULL		
ds_skill VARCHAR(100)		
– Restrições		

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=dFaqvPYalJc>

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.