Sistema Especialista para Recomendação de Trilha Acadêmica

Este trabalho pode ser realizado em grupos de até 4 alunos. **Grupos com mais de 4 membros terão o trabalho anulado.** Leia todo este texto antes de começar e siga o seguinte código de ética: você pode discutir as questões com colegas, professores e amigos, consultar livros da disciplina, bibliotecas virtuais ou físicas, e a Internet em geral, em qualquer idioma. Você pode usar qualquer ferramenta de Inteligência Artificial para dúvidas, mas não para fazer o trabalho no seu lugar. O trabalho é seu e deve ser realizado por você. **Indicativo de plágio, ou trabalhos realizados por Inteligência Artificial, resultarão na anulação do trabalho**.

Os trabalhos entregues serão avaliados por uma ferramenta de Inteligência Artificial, que verificará a originalidade do texto, a autoria do código e a correção dos códigos apresentados. O trabalho deverá ser entregue por meio de um link para um repositório público no GitHub.

1. Objetivo

Pesquisar e praticar conceitos de Sistemas Especialistas para desenvolver um programa em **Prolog** que auxilie estudantes de cursos de tecnologia a escolher uma trilha de especialização. O sistema funcionará de forma interativa, fazendo perguntas sobre as aptidões e interesses do estudante para, ao final, sugerir uma ou mais trilhas acadêmicas, justificando a recomendação com base em uma base de conhecimento predefinida.

2. Descrição do Trabalho

O objetivo é desenvolver um sistema especialista capaz de:

1. **Interagir** com o usuário por meio de um questionário dinâmico.
2. **Coletar** as respostas do usuário sobre suas preferências, habilidades e afinidades com diferentes áreas da tecnologia.
3. **Utilizar um motor de inferência** baseado em regras para processar as respostas e calcular a aderência do perfil do aluno a cada uma das trilhas de especialização disponíveis.
4. **Apresentar** um resultado claro, listando as trilhas recomendadas em ordem de compatibilidade.
5. **Justificar** a recomendação, explicando quais respostas levaram àquela sugestão.
6. **Hospedar** o código-fonte, a base de conhecimento, os arquivos de teste e a documentação em um repositório público no GitHub. Além disso, a critério do aluno, o sistema deve poder ser executado em um ambiente online.

2.1. Base de Conhecimento: Fatos e Regras

A lógica do sistema será definida por uma base de conhecimento em Prolog, contendo fatos e regras.

Fatos: Descrevem o domínio do problema. A base de fatos deve conter, no mínimo:

- **5 Trilhas de Especialização:** (ex: inteligencia_artificial, desenvolvimento_web, seguranca_da_informacao, ciencia_de_dados, redes_e_infraestrutura).
  - Exemplo: trilha(ciencia_de_dados, 'Análise e interpretação de grandes volumes de dados para extrair conhecimento.').
- **Características de Perfil:** Habilidades e interesses associados a cada trilha, com um peso de relevância (1 a 5).
  - Exemplo: perfil(ciencia_de_dados, matematica_estatistica, 5).
  - Exemplo: perfil(desenvolvimento_web, design_visual, 3).
- **Perguntas para o Usuário:** Um conjunto de perguntas para mapear os interesses do usuário às características de perfil.
  - Exemplo: pergunta(1, 'Você tem afinidade com matemática e estatística?', matematica_estatistica).

Regras: Descrevem a lógica de raciocínio do sistema. As regras devem ser capazes de:

- Conduzir o questionário de forma interativa.
- Armazenar as respostas do usuário dinamicamente (usando assertz/1).
- Calcular uma pontuação de compatibilidade para cada trilha com base nas respostas.
- Determinar a(s) trilha(s) mais recomendada(s).
- Exibir os resultados de forma organizada.

3. Motor de Inferência e Interface com o Usuário

O motor de inferência será o próprio mecanismo de resolução do Prolog. A implementação deve focar na criação de predicados que controlem o fluxo do programa:

- Um predicado principal (*iniciar/0*) que começa a interação.
- Predicados para fazer as perguntas, ler e validar as respostas (s/n).
- Um predicado para calcular as pontuações de todas as trilhas.
- Um predicado para ordenar e exibir o ranking final de recomendações.

A interação deve ser clara e ocorrer inteiramente pelo terminal, ou pela área de consulta do Swish Prolog Online. O sistema não deve possuir interface gráfica ou menus complexos.

4. Arquivos de Teste

Para validar a lógica do sistema, devem ser fornecidos no mínimo **3 arquivos de teste** em Prolog. Cada arquivo representará um perfil de aluno diferente, com um conjunto predefinido de respostas.

- Exemplo (perfil_teste_1.pl):

```
% Perfil: Aluno com forte inclinação para lógica e dados.
resposta(1, s). % Resposta para a pergunta 1
resposta(2, n). % Resposta para a pergunta 2
...
```

- O programa principal deverá ser capaz de carregar (*consult/1*) um desses arquivos para executar os testes de forma automatizada, sem a necessidade de entrada manual, para verificar se a recomendação gerada é a esperada para aquele perfil.

5. Hospedagem no GitHub

O projeto deve ser hospedado em um repositório público no GitHub. O repositório deve ser organizado e conter:

- O código-fonte do sistema especialista (.pl). A base de conhecimento (pode estar no mesmo arquivo ou separada). Os 3 (ou mais) arquivos de teste de perfil.
- Documentação (README.md) explicando como compilar/consultar e executar o programa, tanto no modo interativo quanto no modo de teste. A documentação deve conter o nome da instituição, disciplina, professor e nome dos alunos do grupo, em ordem alfabética, seguido do usuário de cada aluno no GitHub.
- O repositório deve ter um histórico de commits claro, e as contribuições de cada membro devem estar registradas, preferencialmente por meio de *pull requests* ou *commits*. **Atenção: a participação de cada aluno será validada por meio dos logs de interação no Github. Se não quiser ser penalizado em nota, não mude o nome de usuário durante o desenvolvimento do sistema.**

6. Divisão de Tarefas Sugerida

O trabalho pode ser dividido entre até quatro alunos, com cada um responsável por uma parte do sistema.

Aluno 1: Modelagem da Base de Conhecimento

- **Responsabilidades:** Pesquisar e definir as trilhas de especialização, as características de perfil associadas a cada uma e os pesos de importância. Elaborar as perguntas que serão feitas ao usuário.
- **Entregável:** Um arquivo Prolog (*base_conhecimento.p*l) contendo todos os **fatos** (*trilha/2, perfil/3, pergunta/3*) que descrevem o domínio do problema.

Aluno 2: Implementação do Motor de Inferência

- **Responsabilidades:** Desenvolver as **regras** em Prolog que calculam a pontuação de compatibilidade de um aluno para cada trilha. Implementar a lógica principal de raciocínio do sistema.
- **Entregável:** O conjunto de predicados (*recomenda/2, calcula_pontuacao/3*, etc.) que formam o núcleo lógico do sistema.

Aluno 3: Interface com o Usuário e Fluxo de Execução

- **Responsabilidades:** Implementar os predicados responsáveis pela interação com o usuário (fazer perguntas, ler respostas). Criar o fluxo principal que chama os outros módulos em ordem (iniciar -> perguntar -> calcular -> exibir).
- **Entregável:** Os predicados de entrada e saída de dados *(iniciar/0, faz_perguntas/0, exibe_resultado/1*) e o controle do programa.

Aluno 4: Testes, Documentação e Gerenciamento do Repositório

- **Responsabilidades:** Criar os 3 arquivos de teste com perfis de alunos distintos. Desenvolver predicados de teste para carregar esses perfis e validar os resultados. Manter o repositório no GitHub, escrever o README.md e gerenciar a integração do trabalho dos outros membros.
- **Entregável:** Arquivos de teste (*perfil_X.pl*), predicados de teste e o README.md completo.

7. Avaliação

O trabalho será avaliado com base nos seguintes critérios:

- **Lógica e Funcionalidade (70%):**
  - A base de conhecimento é coesa e bem modelada.
  - As regras de inferência produzem resultados lógicos e consistentes.
  - O sistema é totalmente funcional no modo interativo e de teste.
- **Organização e Legibilidade do Código (15%):**
  - Código Prolog claro, comentado e bem estruturado.
  - Repositório GitHub organizado, com README.md claro e commits significativos.
- **Robustez (15%):**
  - O sistema lida adequadamente com as entradas do usuário.
  - Os testes cobrem diferentes perfis de forma a validar a lógica do sistema.

ATENÇÃO: nesta especificação existem definições de nomes de predicados, funções e arquivos. O aluno deve usar estes nomes nos seus códigos. A falta destes nomes resultará em perda de pontos durante a avaliação.

8. Entrega e Prova de Autoria

Um aluno do grupo será sorteado para responder a uma pergunta sobre o funcionamento ou implementação de qualquer parte do código. A falha na resposta implicará em uma redução de 35% na nota final do grupo. Se algum aluno do grupo faltar na prova de autoria, o grupo automaticamente perde os 35%.

O link para o repositório público no GitHub deve ser entregue na data estipulada. O repositório deve conter todos os artefatos mencionados na seção 5.