Enunciado do Projeto em Engenharia de Conhecimento

1. Tema de Estudo

Engenharia (projeto, *design*, implementação e operação) de Sistemas de Suporte à Decisão baseados em Conhecimento.

2. Objetivos de Aprendizagem

Integrar os conteúdos das Unidades Curriculares do perfil (Bases de Dados NoSQL, Mineração de Dados e Big Data), aplicando *metodologias de Engenharia de Software*.

> **Não se pretende apenas um protótipo funcional, mas um sistema desenhado, documentado e testado.**

Mitigação de Risco: Dada a dimensão dos grupos e o tempo limitado (15 semanas), recomenda-se a adoção de uma estratégia MVP (*Minimum Viable Product*). É preferível ter um sistema simples que percorre todo o *pipeline* (do dado ao *dashboard*) na semana 8, e depois iterar, do que tentar construir tudo de uma vez e falhar na integração final.

3. Dinâmica e Metodologia das Aulas

A UC segue uma abordagem de **Aprendizagem Baseada em Projeto (PBL)**. As aulas são sessões de trabalho intercaladas:

1. **Sessões Tutoriais e Metodológicas:** Introdução à métodos, técnicas e princípios envolvidos no processo de desenvolvimento de software. Nessas ocasiões poderão ser fornecidas fichas de trabalho cujos resultados devem alimentar o Relatório Final.
2. **Reuniões Técnicas:** O docente atua como *Product Owner*. Os grupos apresentam o progresso, validam ADRs (*Architecture Decision Records*) e desbloqueiam impedimentos. É parte da avaliação os grupos apresentarem dúvidas ou bloqueios técnicos.
3. **Coordenação e Desenvolvimento:** Espera-se que os grupos aproveitem as sessões de aula para as tarefas de maior complexidade ou que exijam consenso (design, arquitetura, integração), deixando as tarefas individuais de codificação para o trabalho autónomo.

4. Descrição do Problema

4.1. Enunciado Global

Todos os grupos deverão produzir um **Sistema de Suporte à Decisão (SSD)**. O sistema deve integrar um *Dashboard* Analítico e um *Assistente Inteligente* (Chatbot).

> **Aviso de Exequibilidade:** O foco é a Engenharia de Backend e Dados. Não se recomenda o desenvolvimento de interfaces complexas. Sugere-se a utilização de bibliotecas de *Data Apps* como Streamlit, Chainlit, ou Gradio para garantir que o esforço é alocado na inteligência do sistema e não na camada UX.

4.2. Enunciado Específico

Cada grupo escolhe um domínio único. Exemplos:

- Análise de Investimentos (Séries temporais + Notícias financeiras);
- Monitorização de Saúde (Dados de sensores + Literatura médica);
- E-Commerce Inteligente (Catálogo de produtos + Reviews de clientes).

5. Requisitos Técnicos

5.1. Pipeline de Dados e Conectividade

1. **Ingestão:** Recolher >100k registos de fontes heterogéneas (APIs, *crawlers*, datasets).
2. **Persistência:**
    1. **Estruturada (SQL):** Para dados transacionais ou muito estruturados (ex: PostgreSQL).
    2. **Não Estruturada (NoSQL):** Para documentos, *logs* ou dados multi-modais (ex: MongoDB).
    3. **Vetorial (Vetorial):** Para armazenar *embeddings* e alimentar o sistema de RAG (ex: Chroma).
3. **Conectividade Externa (Opcional):** Implementar o *Model Context Protocol* (MCP) para a aquisição de dados externos.

> **Opcional:** O grupo pode explorar o conceito de *soft-data* por meio do qual se usa LLMs como classificador. Neste caso, pode-se considerar o uso de PEFT/LoRA para melhoria de performance.

5.2. Inteligência NeSy

Os grupos têm liberdade para desenhar a arquitetura de IA, desde que o sistema final adote uma abordagem Neuro-simbólica (NeSy). Portanto, o sistema não deve ser apenas um modelo de linguagem (LLM) isolado, mas sim uma solução composta que integre capacidades IA, nas suas diferentes formas, com funções determinísticas.

1. **Seleção de Ferramentas:** O sistema deve ter pelo menos 2 ferramentas distintas (ex: uma ferramenta de busca vetorial e uma ferramenta de cálculo/SQL).
2. **Autonomia de Dados:** O sistema deve mostrar autonomia na seleção de fontes de dados:
    1. **Exploração Semântica (RAG/MCP):** Capacidade de responder a perguntas baseadas em documentos não estruturados recolhidos (Notícias, PDFs, Relatórios).
    2. **Operações Estruturadas (Tools/Function Calling):** Capacidade de interagir com dados estruturados (SQL/NoSQL) ou APIs para obter respostas exatas (ex: cálculos).

> **Obs:** A escolha entre Modelos Proprietários (via API como OpenAI) ou Modelos Abertos/Locais (Ollama) fica a critério do grupo, devendo ser baseada na complexidade da tarefa e no hardware

disponível. O grupo deve documentar esta decisão, considerando o *trade-off* entre latência, custo e privacidade dos dados.

Opcional: O grupo pode explorar o conceito de *Agentic AI* buscando orquestrar dois ou mais agentes especialistas.

5.3. Interface de Suporte à Decisão

Em concordância com o aviso de exequibilidade (foco no Backend), a interface deve ser desenvolvida utilizando frameworks de Data Apps (Streamlit, Chainlit ou Gradio).

1. **Chat Integrado ao Contexto:** O chat não é apenas uma caixa de texto; ele deve ter permissão para alterar o estado do dashboard. Por exemplo, se o utilizador pede "Compare as vendas deste ano com o anterior", o agente executa a query e o Streamlit/Gradio atualiza o painel.
2. **Visualização de Métricas:** O dashboard deve apresentar visualizações claras dos dados processados (KPIs, tabelas, gráficos de séries temporais) alimentados diretamente pelas pipelines de dados estruturados. Ao efetuar interações, o agente pode oferece comentários e insights adicionais.

   Opcional: O grupo pode explorar o conceito de *Conversational UX* por meio do qual se faz uso de *scripts* de conversa para uma interação melhor direcionada com o utilizador.

6. Engenharia de Software e Processo

A ser complementado pelas Sessões Tutoriais e Metodológicas.

6.1. DevOps (Simplificado)

- **Repositório:** Git com estratégia simplificada (GitHub Flow ou Trunk-Based). Pull Requests (PRs) são obrigatórios para fundir código na `main`.
- **Containerização:** O projeto deve ter um `Dockerfile` e um `docker-compose.yml`. O objetivo é que o docente consiga rodar o projeto sem instalar bibliotecas Python manualmente.
- **CI Básico:** Um pipeline simples (GitHub Actions) que corre apenas os testes unitários e um linter (ex: `ruff` ou `flake8`) a cada PR.

6.2. MLOps (Simplificado)

1. **Prompts Organizados:** Proibido ter prompts *hardcoded* no meio das funções. Devem estar em ficheiros separados (ex: `prompts.yaml` ou `consts.py`) para facilitar a edição.
2. **Logging:** O sistema deve gerar um Log de Execução que registe para cada interação *(Isto serve para o grupo depurar onde o Agente errou).*

6.3. Arquitetura (Simplificado)

1. **Modelagem C4 (Context, Containers, Components and Code):** O diagrama deve evidenciar claramente a fronteira entre os componentes **Determinísticos** (Backend API, Banco de Dados SQL) e os componentes **Aproximados** (LLM, Vector Storage). Deve estar explícito onde ocorre o *Router/Decision Making.*
2. **Design de Interação:** Diagramas de Sequência para os fluxos principais (ex: Fluxo "Pergunta do Utilizador" → "Router" → "Query SQL" → "Visualização").

O desenho do sistema deve preceder a implementação. É obrigatória a entrega de diagramas que explicitem a natureza híbrida da solução.

6.4. Estratégia de Testes (MVP)

1. **Unitários:** Testem apenas as funções "core" (ex: a função que limpa os dados ou a função que gera o JSON para o gráfico).
2. **Integração:** Um teste "Smoke Test" que garante que a aplicação sobe e conecta ao Banco de Dados.
3. **Avaliação Agêntica:** Criar um script simples (`eval.py`) com 5 a 10 perguntas de controlo, criar forma para avaliar o *resoning*.

A cobertura de testes não precisa ser 100%, o foco é no caminho crítico.

7. Gestão do Projeto e Planeamento

A capacidade de entregar valor dentro do prazo é um objetivo de aprendizagem central.

7.1. Gestão de Escopo e Requisitos

1. **Backlog e User Stories:** O projeto deve ser decomposto em Histórias de Utilizador (ex: *"Como analista, quero comparar vendas por ano para identificar tendências"*).
2. **Priorização:** Dada a restrição temporal (15 semanas), o grupo deve classificar os requisitos para definir o MVP (Minimum Viable Product).
3. **Definição de Pronto:** Uma funcionalidade só está "feita" quando o código está no repositório, testado e documentado.

7.2. Gestão de Recursos e Equipa

Embora todos devam programar, sugere-se a divisão de responsabilidades claras:

1. **Data Engineer:** Focado na ingestão, ETL e Base de Dados.
2. **AI Engineer:** Focado nos Prompts, RAG, Tools e Avaliação.
3. **Full-stack/Backend:** Focado na API, Integração e Dashboard.
4. **Project Master:** Focado na gestão do projeto e qualidade/ESG.

O uso de uma ferramenta de gestão de tarefas para evidenciar o progresso nas reuniões com o docente (*Product Owner*) é incentivado.

7.3. Planeamento de Deploy e Entrega

1. **Ambiente de Produção Simulado:** O sistema não deve rodar apenas "na máquina do aluno". A entrega final deve incluir a configuração para *deploy* (ex: Docker Compose robusto ou *deploy* gratuito em Render/Railway/HuggingFace Spaces).
2. **Gestão de Configuração:** Segredos (API Keys, Passwords) **nunca** devem estar no código (*hardcoded*). Devem ser geridos via variáveis de ambiente (`.env`) e solicitados por uma interface de configuração, caso necessário.

8. Entregáveis e Apresentação

8.1. Relatório Técnico-Científico

1. **Problema e Solução:** O que fizeram e para quem.
2. **Decisões de Engenharia (ADRs):** Justificação das escolhas tecnológicas.
3. **Implementação:** Como funciona o *pipeline* e os agentes.
4. **Análise Crítica:** O que correu bem, o que correu mal e métricas de desempenho.

   Considere os **Requisitos Técnicos** do projeto como temas orientadores na elaboração do relatório.

8.2. Apresentação e Demo

1. **Foco:** Demonstração ao vivo (*Live Demo*).
2. **Slide de Arquitetura:** Obrigatório mostrar como os componentes comunicam.
3. **Lições Aprendidas:** Reflexão honesta sobre as dificuldades de integração.

   Apresentação de 20 minutos. Suegere-se o estudo de iniciativas como 3-minutes thesis para maior impacto.

9. Metodologia de Avaliação

Avaliação contínua para premiar a consistência.

Componente	Peso	Detalhe
A. Processo (Contínuo)	**60%**	Planeamento, Git limpo, CI/CD, ADRs e presença nas tutorias. Exige plano semanal.
B. Relatório	**30%**	Qualidade técnica e clareza da documentação.
C. Produto e Demo	**10%**	O sistema funciona? A defesa oral foi sólida?

9.1. Regra de Ouro da Avaliação Contínua

Na **segunda semana**, o Coordenador apresenta o cronograma.

- Semana 1-4: Dados e BD.
- Semana 5-9: Lógica e IA.
- Semana 10-13: Integração e UI.
- Semana 14-15: Polimento.
- Semana 16: Apresentação

  O grupo que chegar à semana 10 sem ter a BD pronta terá a nota penalizada.

9.2. Avaliação Individual

A nota final de cada aluno deriva da nota global do projeto, ajustada pelo fator de Avaliação por Pares. Ao final do curso, o grupo distribui consensualmente um fator (entre 0,5 e 1,5) pelo esforço individual, garantindo que a soma dos fatores iguale o número de membros. Exemplo, num grupo de 3, se A=1, B=1.5 e C=0.5 (total = 3), significa que A participou na medida do esperado, B se envolveu com mais esforço e C teve um empenho inferior.

10. Integridade Académica

Os autores do presente trabalho académico devem agir com integridade e declarar expressamente que não recorreram a práticas de plágio, nem a qualquer forma de uso indevido, manipulação ou falsificação de informações e/ou resultados em nenhuma das etapas conducentes à sua elaboração. Para além disso, devem ainda ter conhecimento do Código de Conduta Ética da Universidade do Minho e assegurar o seu integral cumprimento ao longo de todo o processo de elaboração deste trabalho.

O uso de IA (ChatGPT/Copilot) é permitido como "copiloto", mas proibido como "piloto automático". * Código gerado por IA sem compreensão, que o aluno não saiba explicar na oral, será considerado fraude. * Todo o código deve ser revisto e testado pelos humanos do grupo.