



# Mestrado Engenharia Informática

## Projeto de Informática (2024/25)

### Relatório

#### DETECÇÃO DE PEDIDOS INDESEJADOS EM INVOCAÇÕES A LLMs (E02)

##### Grupo de Trabalho

Diogo Costa  
PG53783



João Brito  
PG53944



João Moreira  
PG50465



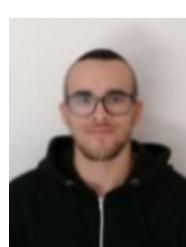
João Vale  
PG53915



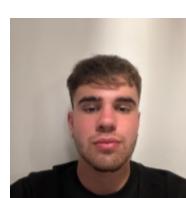
Henrique Fernandes  
A95323



Ricardo Fonseca  
PG52702



Telmo Oliveira  
PG54247



Braga, 12 de dezembro de 2024

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Motivação . . . . .	2
1.2	Objetivos do Projeto . . . . .	2
1.3	Gestão do Projeto . . . . .	3
<b>2</b>	<b>Análise de Requisitos e Modelagem de Interações</b>	<b>4</b>
2.1	Levantamento de Requisitos . . . . .	4
2.2	Requisitos Funcionais . . . . .	4
2.3	Requisitos Não Funcionais . . . . .	4
2.4	Modelagem do Sistema . . . . .	5
<b>3</b>	<b>Arquitetura e Ferramentas</b>	<b>6</b>
3.1	Ferramentas Utilizadas . . . . .	6
3.2	Estrutura da aplicação . . . . .	6
<b>4</b>	<b>Interface</b>	<b>8</b>
<b>5</b>	<b>Implementação da API</b>	<b>10</b>
5.1	Rotas da API . . . . .	10
<b>6</b>	<b>Análises</b>	<b>13</b>
6.1	Análise de Linguagem . . . . .	13
6.2	Análise de Prompt Template . . . . .	13
6.3	Análise de Off Topic . . . . .	13
6.4	Análise de On Topic . . . . .	13
6.5	Análise de Prompt Injection . . . . .	14
6.6	Análise de Toxicidade . . . . .	14
6.7	Parâmetros e Configurações . . . . .	14
<b>7</b>	<b>Estratégias de Escalabilidade</b>	<b>15</b>
<b>8</b>	<b>Análise de Resultados e Testes</b>	<b>16</b>
<b>9</b>	<b>Segunda Componente</b>	<b>20</b>
9.1	Modelo de Negócio . . . . .	20
9.2	Estratégia de Entrada no Mercado . . . . .	22
9.3	Estratégia de Crescimento . . . . .	23
<b>10</b>	<b>Conclusão e Trabalho Futuro</b>	<b>25</b>

# 1. Introdução

O uso de modelos de linguagem natural de grande escala, ou *Large Language Models* (LLMs), tem-se tornado uma realidade cada vez mais presente em setores como o atendimento ao cliente, medicina, *e-commerce* e educação, trazendo melhorias significativas na forma como as interações e operações são realizadas. LLMs, como o ChatGPT da OpenAI e o Claude da Anthropic, são amplamente adotados devido à sua capacidade de entender e gerar texto de forma altamente contextualizada, imitando interações que se assemelham à comunicação humana. Esta capacidade permite uma vasta amplitude de aplicações, proporcionando respostas e soluções de forma automatizada e personalizada.

Contudo, à medida que o uso das LLMs cresce, surgem preocupações críticas de segurança e integridade na sua implementação e uso. Um dos desafios emergentes é a deteção de pedidos indesejados e tentativas de manipulação do modelo, conhecidas como *prompt injections*. Estes ataques ocorrem quando entradas maliciosas são elaboradas para manipular o modelo e gerar respostas inapropriadas ou mesmo expor dados sensíveis que deveriam permanecer protegidos. Em sistemas que integram LLMs como intermediários, tais ataques representam um risco elevado, especialmente quando o modelo é usado em áreas onde o sigilo, precisão e segurança são essenciais.

## 1.1 Motivação

Este projeto surge como uma resposta a essas necessidades de segurança, propondo a criação de um *Software Development Kit* (SDK) *middleware* capaz de monitorizar e filtrar as interações entre utilizadores e LLMs, garantindo que tanto as solicitações como as respostas são examinadas para verificar a presença de temas sensíveis ou tentativas de manipulação. O SDK proposto será dotado de flexibilidade configurável, permitindo que os utilizadores possam definir uma lista de tópicos indesejados, ajustável conforme o contexto da aplicação. Através desta abordagem, o projeto visa assegurar que a LLM opera com a integridade e segurança adequadas ao contexto, prevenindo comportamentos ou respostas indesejadas.

Ao desenvolver este SDK, pretende-se não só oferecer uma camada de segurança adicional para sistemas baseados em LLM, mas também promover a confiança dos utilizadores e organizações na adoção dessas tecnologias, minimizando os riscos associados à manipulação das respostas do modelo. A modularidade do SDK permitirá uma integração fácil em diferentes setores, e a personalização dos parâmetros de análise garantirá a adaptação a uma ampla gama de aplicações e contextos.

## 1.2 Objetivos do Projeto

O principal objetivo deste projeto é criar uma solução configurável e modular para assegurar a integridade das interações com LLMs. Os objetivos específicos são os seguintes:

- **Análise de Pedidos e Respostas:** Desenvolver um SDK que analise pedidos e respostas, que identifique temas indesejados (como religião, política ou linguagem tóxica), que detete tentativas de *prompt injection* que podem interferir nas respostas geradas e que deteta se o prompt template está a ser revelado no texto.
- **Configuração e Personalização:** Fornecer uma lista de tópicos e parâmetros de análise personalizáveis, permitindo que os utilizadores selezionem os temas específicos de acordo com as suas necessidades de segurança e contexto operacional.
- **Escalabilidade e Suporte a Múltiplos Idiomas:** Estruturar o SDK de forma modular, permitindo fácil adaptação a diferentes volumes de uso e garantindo suporte aos idiomas português e inglês, para que o sistema seja útil em contextos multilingues e altamente escaláveis.
- **Adaptabilidade a Novas Ameaças:** Projetar o SDK para se adaptar a novas variações de ataques e tentativas de manipulação, permitindo atualizações contínuas para suportar e detetar variações mais sofisticadas de *prompt injection* e outras ameaças emergentes.

Ao abordar esses objetivos, espera-se que o SDK ofereça uma plataforma de monitorização que não só protege a integridade das interações com LLMs, mas também facilita a adaptação contínua às ameaças de segurança em evolução. Desta forma, este projeto contribuirá para a criação de soluções mais seguras, eficientes e confiáveis no uso de modelos de linguagem natural em setores sensíveis e de alta responsabilidade.

### 1.3 Gestão do Projeto

Para a realização deste projeto, contamos com a colaboração essencial do orientador João Macedo, engenheiro principal da Agentifai, que desempenhou um papel fundamental na clarificação dos objetivos do projeto, ajudando-nos a definir e ajustar as diretrizes para o desenvolvimento do SDK. Durante todo o processo, o João esteve disponível para esclarecer dúvidas e orientar a equipa em relação a desafios técnicos e de implementação, assegurando que o projeto estivesse alinhado com as melhores práticas.

A gestão do projeto incluiu reuniões frequentes entre os membros do grupo para coordenar as atividades e monitorizar o desenvolvimento das tarefas. Além disso, a cada duas semanas, realizávamos uma reunião com o orientador João Macedo, onde apresentávamos os últimos progressos, discutíamos dúvidas e alinhávamos os próximos passos com o apoio dele. Estes encontros quinzenais com o orientador funcionaram como pontos de controle, permitindo rever o desenvolvimento do projeto e assegurar que seguíamos na direção correta.

A distribuição das tarefas dentro do grupo foi feita de forma dinâmica, de acordo com o progresso de cada membro, assim que um elemento completava uma tarefa específica, uma nova tarefa era atribuída, de forma a garantir a continuidade do progresso do projeto e a colaboração ativa de todos. Este método flexível de gestão de tarefas permitiu que cada membro se envolvesse de forma significativa em cada fase, garantindo uma abordagem colaborativa e eficiente.

## 2. Análise de Requisitos e Modelagem de Interações

Este capítulo explora os processos necessários para entender e definir as necessidades e funcionalidades da aplicação. Esta análise é essencial para garantir que o sistema atenda às expectativas dos utilizadores e funcione de maneira eficaz. Assim, é possível delinear um fluxo claro de comunicação e operação entre as partes envolvidas, facilitando a criação de uma arquitetura de software alinhada com os objetivos do projeto. Para uma visão mais aprofundada sobre o processo de levantamento de requisitos e outros fatores relevantes para a modelagem da aplicação, recomenda-se também a leitura do documento de requisitos desenvolvido separadamente.

### 2.1 Levantamento de Requisitos

O levantamento de requisitos define necessidades funcionais e não funcionais do sistema, esta etapa documenta os critérios de validação, comunicação entre componentes e regras de filtragem, assegurando que o sistema identifica solicitações potencialmente maliciosas ou indesejadas. Seguidamente, são apresentados alguns exemplos.

### 2.2 Requisitos Funcionais

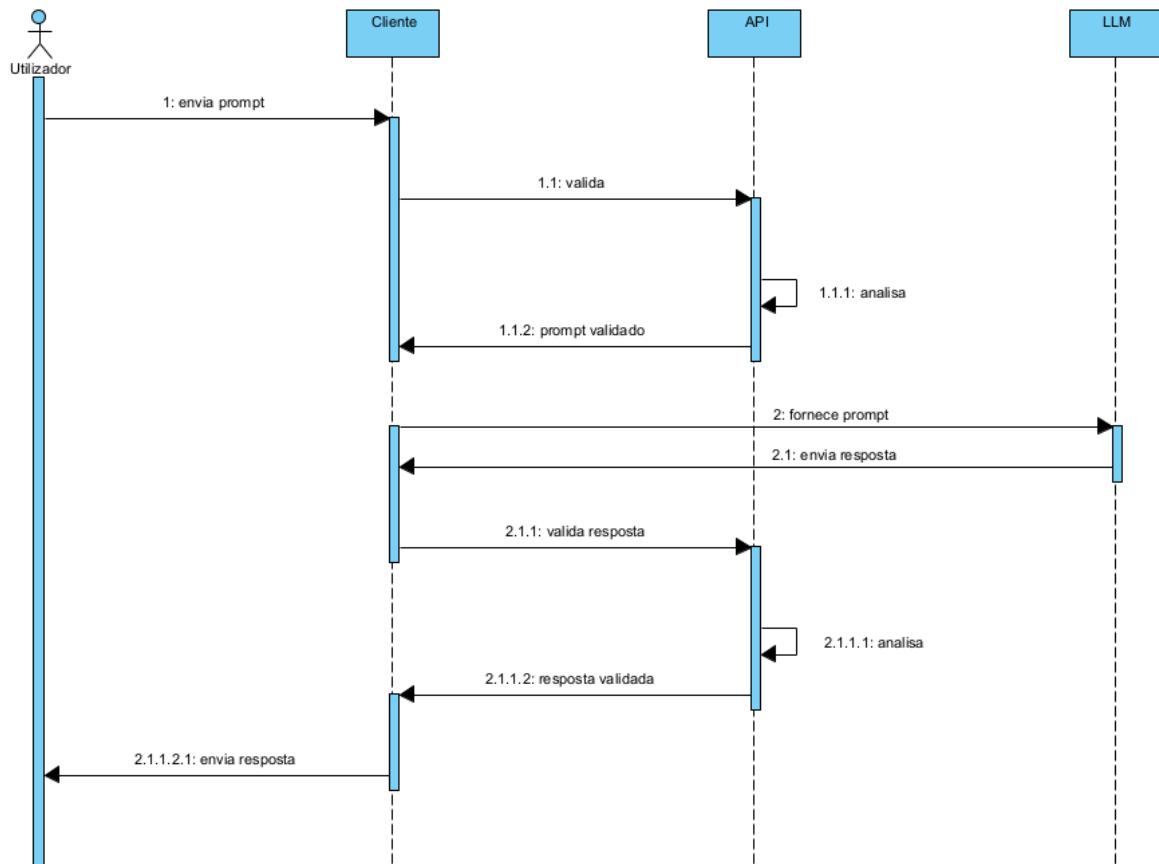
- **Deteção nos Pedidos de:** tópicos indesejados e desejados a determinar pelo utilizador, ou tentativas de prompt injection.
- **Parametrização:** permitir a parametrização e atualização da lista de tópicos indesejados e desejados, permitindo personalizações conforme o contexto.
- **Deteção nas Respostas de:** tópicos indesejados e desejados, linguagem tóxica, e revelar o prompt template nas respostas geradas pelas LLMs.
- **Monitorização:** em tempo real das invocações, identificando atividades suspeitas que incluem tentativas de violação de segurança ou tópicos indesejados.

### 2.3 Requisitos Não Funcionais

- **Usabilidade:** deve ser fácil de integrar com diferentes plataformas que utilizam LLMs, fornecendo uma documentação clara e APIs amigáveis
- **Performance:** o SDK deve ser capaz de escalar horizontalmente
- **Manutenção e Suporte:** ajustar as listas de assuntos indesejados e desejados sem precisar interromper o funcionamento do sistema
- **Cultural e Político:** o produto deve ser capaz de suportar análises de pedidos em inglês e português.

## 2.4 Modelagem do Sistema

O diagrama de sequência abaixo ilustra o fluxo de interação ideal entre o utilizador, o cliente, a API e a LLM no processo de validação de prompts e respostas. A comunicação começa quando o utilizador envia um prompt, que é inicialmente recebido pelo cliente e enviado para análise na API. Em seguida, a API verifica a conformidade do prompt e o cliente envia o prompt validado para a LLM. A LLM retorna uma resposta que passa novamente pelo cliente e pela API para uma última validação antes de ser enviada de volta ao utilizador. Este processo assegura que qualquer prompt ou resposta que possa conter pedidos indesejados seja identificado e filtrado antes de alcançar o destinatário final.



**Figura 2.1:** Diagrama de Sequência

## 3. Arquitetura e Ferramentas

### 3.1 Ferramentas Utilizadas

Para a implementação do SDK, utilizamos as seguintes ferramentas:

- **Flask:** Uma microframework de web em Python, leve e fácil de usar, ideal para criar APIs e pequenas aplicações web.
- **Transformers:** Uma biblioteca desenvolvida pela Hugging Face, usada para trabalhar com modelos de processamento de linguagem natural (NLP), como BERT, GPT, e T5.
- **Torch:** Uma biblioteca de deep learning desenvolvida pelo Facebook AI Research (FAIR), que facilita a criação e o treino de redes neurais.
- **SentencePiece:** Uma biblioteca de tokenização desenvolvida pela Google, usada em processamento de linguagem natural para dividir textos em unidades menores, como sub-palavras ou tokens, o que facilita o trabalho com modelos de NLP.
- **Googletrans:** Biblioteca que permite acesso à API de tradução do Google Translate.
- **Docker:** Plataforma de contêineres que permite criar, distribuir e executar aplicações de forma isolada e portátil.
- **Python-Levenshtein:** Uma biblioteca que implementa o algoritmo de distância de Levenshtein, usado para medir a similaridade entre strings, sendo útil em aplicações de comparação e correção de texto.
- **FuzzyWuzzy:** Uma biblioteca baseada em Python-Levenshtein, que oferece uma interface amigável para comparar strings e calcular similaridades, muito utilizada em processamento de texto.
- **NLTK (Natural Language Toolkit):** Um conjunto de bibliotecas para o processamento de linguagem natural em Python, que oferece ferramentas para tokenização, stemming, análise gramatical e muito mais.

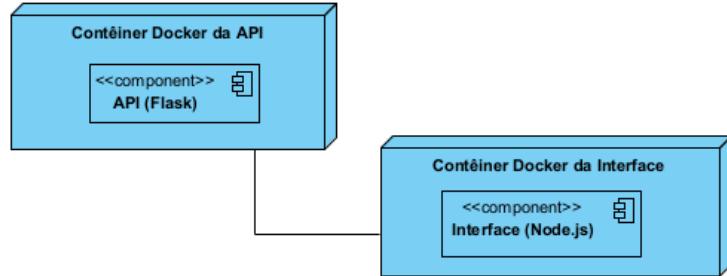
### 3.2 Estrutura da aplicação

A arquitetura desta aplicação é composta por dois principais componentes: a API e a interface, ambos executados em contêineres Docker. Esta estrutura permite a modularidade, facilitando a escalabilidade e a manutenção do sistema, além de garantir um ambiente isolado e controlado para cada serviço.

#### Componentes:

- **API:** construída em Flask, é responsável por processar os pedidos, executar as análises e manipular as requisições recebidas pela aplicação. A API recebe prompts ou respostas de LLM's, realiza a tradução para inglês (quando necessário) e executa as análises específicas para cada solicitação.

- **Interface:** foi desenvolvida em Node com o propósito de demonstrar a API, oferecendo uma plataforma interativa onde os utilizadores podem consultar as análises disponíveis, inserir prompts ou respostas de LLMs, e visualizar os resultados.



**Figura 3.1:** Diagrama de Deployment

Ambos os contêineres são geridos pelo Docker, o que facilita a gestão dos serviços e permite que os contêineres se comuniquem entre si, utilizando uma rede interna configurada pelo Docker. Esta configuração assegura que a interface pode aceder a API de forma direta e eficiente, sem exposição desnecessária à rede externa.

## 4. Interface

Como referido anteriormente, a interface desenvolvida serve apenas para o propósito de demonstrar a API, permitindo que o cliente visualize as análises realizadas sobre os inputs dos utilizadores e os outputs das LLMs. Está implementada em um servidor Node.js, que integra as funcionalidades da API com uma interface de utilizador dinâmica e interativa.

A página inicial da interface exibe uma barra de botões que permite alternar entre os dois tipos de análise (análise de um input do prompt do utilizador e análise de um output da LLM), ajustando os parâmetros e campos conforme a seleção feita pelo utilizador. Nesta página, o utilizador encontra campos de texto e checkboxes onde pode inserir o prompt e os parâmetros específicos de cada análise. Ao submeter o formulário, o botão de envio ativa uma página de loading que exibe um spinner, avisando o utilizador que a requisição está em processamento pela API e proporcionando uma experiência mais interativa e informativa.

The figure consists of three screenshots of a web application interface titled "Validation Form".

- Validate Input:** This screenshot shows the initial state of the form. It has two buttons at the top: "Validate Input" (blue) and "Validate Output" (gray). Below these are two text input fields: "Enter the prompt:" containing placeholder text "Hey, how are you?" and "Enter LLM response:". Underneath the inputs is a section titled "Select analyses to apply:" containing five checkboxes:
  - DetectPromptInjection - An analysis that identifies and flags attempts to manipulate the system by injecting prompt-based instructions.
  - OnTopic - An analysis that detects if the input text contains content related to on-topic subjects, based on predefined categories.
  - OffTopic - An analysis that detects if the input text contains content related to off-topic subjects, based on predefined categories.
  - LanguageDetector - An analysis that detects the language of the input text based on a pre-trained model.
  - PromptTemplate - An analysis that detects if the output text contains the prompt template.A blue "Submit" button is located at the bottom.
- Validate Output:** This screenshot shows the form after a selection has been made. The "Validate Input" button is now gray, and the "Validate Output" button is blue. The "Enter LLM response:" field is empty. The "Select analyses to apply:" section is identical to the first screenshot. A blue "Submit" button is located at the bottom.
- Loading:** This screenshot shows the form during processing. The "Validate Input" and "Validate Output" buttons are now grayed out. A large blue circular spinner is centered on the page, indicating that the request is being processed by the API.

**Figura 4.1:** Páginas do Validate Input, Validate Output e Loading

Após o processamento dos dados pela API Flask, o utilizador é redirecionado para a mesma página, onde os resultados são exibidos abaixo de cada análise selecionada. Para as categorias DetectToxicity, DetectPromptInjection, OnTopic e OffTopic, os resultados são apresentados em gráficos, com uma escala de cores dinâmica, de maneira a facilitar a leitura e interpretação dos resultados.



**Figura 4.2:** Página com os resultado das Análises

Caso ocorra um erro ao conectar-se à API, o utilizador é redirecionado para uma página de erro, que inclui uma mensagem informativa sobre o problema e um botão de retorno à página inicial, oferecendo uma forma rápida e fácil de recomeçar a interação.

## 5. Implementação da API

A API foi desenvolvida para facilitar a interação com um conjunto de análises especializadas, oferecendo aos utilizadores a capacidade de consultar informações detalhadas e realizar operações de validação de prompts e das respostas das LLMs. Cada rota inclui logs dos pedidos e das suas respostas, permitindo o monitoramento e a análise de uso da API, o que reforça a transparência e a rastreabilidade das interações realizadas.

É importante mencionar que a tradução de português para inglês (quando o prompt está em português) pode introduzir alguns erros nas análises e uma maior latência na resposta. No entanto, essa abordagem é vantajosa, uma vez que os modelos de análise mais bem treinados e eficientes estão disponíveis em inglês. Criar modelos com desempenho comparável em português seria um desafio complexo e demorado, o que torna essa estratégia uma escolha prática para garantir eficiência.

Com as rotas disponíveis, a API proporciona uma interface robusta capaz de interagir com as análises, garantindo resultados precisos e facilitando a compreensão das informações apresentadas. A seguir, são detalhadas as rotas disponíveis:

### 5.1 Rotas da API

#### /analyses

- **Input:** nada
- **Output:**

```
{  
    "analyses": [  
        {  
            "name_of_analysis": "string",  
            "description": "string",  
            "params": {  
                "name_of_parameter": {  
                    "description": "string",  
                    "type": "type of parameter (int/string/...)"  
                },  
                "type": "prompt(input) or response(output) or both(all)"  
            }  
        }  
    ]  
}
```

Descrição: Esta rota utiliza o método GET e tem como objetivo retornar uma lista das análises disponíveis da nossa aplicação. Cada análise é acompanhada pelas suas respectivas descrições, que fornecem informações detalhadas sobre o que cada uma implica, e pelos parâmetros necessários para a sua execução. Esta estrutura permite que os utilizadores compreendam facilmente as opções disponíveis e como utilizá-las de forma eficaz.

## /analysis/<analysis\_name>

- **Input:** o nome da analysis presente no parâmetro de rota
- **Output:** um dicionário que contém somente uma chave que é a análise pretendida

```
{  
    "name_of_analysis": {  
        "description": "string",  
        "params": {  
            "name_of_parameter": {  
                "description": "string",  
                "type": "type of parameter (int/string/...)"  
            }  
        },  
        "type": "prompt(input) or response(output) or both(all)"  
    }  
}
```

Descrição: Esta rota utiliza o método GET e tem como objetivo retornar a análise solicitada no pedido. A resposta inclui uma descrição detalhada da análise, além dos parâmetros necessários para a sua execução. Isto permite que os utilizadores compreendam claramente a análise específica que estão a requisitar e como utilizá-la adequadamente.

## /validate\_input

- **Input:**

```
{  
    "prompt": "string",  
    "analyses": [  
        {  
            "name_of_analysis": "string",  
            "params": ['values']  
        }  
    ]  
}
```

- **Output:**

```
{  
    "status": "fail or success of running the analyses",  
    "prompt": "string",  
    "analysis_name": {  
        "message": "string",  
        "score": "string"  
    }  
}
```

Descrição: Esta rota utiliza o método POST recebendo um prompt junto com uma lista de análises e os seus respetivos parâmetros. Se o prompt estiver em português, a nossa API irá traduzi-lo para inglês, pois grande parte dos modelos suportam exclusivamente o idioma inglês, garantindo assim maior precisão nos resultados. Após a tradução, criamos uma thread para cada análise especificada, permitindo que todas sejam processadas simultaneamente. Ao final do processo, retornamos as respostas de cada análise relacionada ao prompt, proporcionando uma visão completa e detalhada dos resultados obtidos.

## /validate\_output

- Input:

```
{  
    "prompt": "string",  
    "analyses": [  
        {  
            "name_of_analysis": "string",  
            "params": ["values"]  
        }  
    ]  
}
```

- Output:

```
{  
    "status": "fail or success of running the analyses",  
    "prompt": "string",  
    "analysis_name": {  
        "message": "string",  
        "score": "string"  
    }  
}
```

Descrição: Esta rota utiliza o método POST recebendo uma resposta de uma LLM, junto com uma lista de análises e os seus respetivos parâmetros. Se a resposta da LLM estiver em português, a nossa API irá traduzi-la para inglês, uma vez que os modelos suportam apenas o idioma inglês, assegurando maior precisão nos resultados. Após a tradução, criamos uma thread para cada análise especificada, permitindo que todas sejam processadas simultaneamente. Ao final do processo, retornamos as respostas de cada análise relacionada a resposta da LLM, proporcionando uma visão completa e detalhada dos resultados obtidos.

## 6. Análises

As análises foram configuradas para suportar entradas em inglês e português, ao receber um prompt em português, ele é automaticamente traduzido para inglês, com auxílio da biblioteca googletrans, garantindo que as análises subsequentes, que operam em inglês, possam processar corretamente textos em português.

### 6.1 Análise de Linguagem

A análise de linguagem é feita pela função *language\_detector*, que utiliza os modelos XLM-RobertaTokenizer e AutoModelForSequenceClassification, esta função identifica o idioma do texto entre várias opções, como inglês, espanhol, francês, português, entre outros. A entrada é tokenizada e o modelo calcula a probabilidade para cada idioma, o idioma com maior probabilidade é então retornado como o detetado. Esta análise pode ser realizada tanto para o input do utilizador como para o output da LLM.

### 6.2 Análise de Prompt Template

A análise do prompt template visa identificar semelhanças relevantes entre o texto gerado (output) e um template específico, avaliando a presença de partes de texto que possam indicar uma replicação excessiva do template original. Este processo é realizado pela função *detect\_template\_in\_output*, que utiliza tanto a distância de Levenshtein como a biblioteca FuzzyWuzzy para calcular a similaridade entre n-gramas (sequências de palavras consecutivas) no template e no output. A função calcula uma média de similaridade para cada par de n-gramas e armazena aqueles que superaram um limite de similaridade individual predefinido. A função calcula, ainda, uma média geral de similaridade, se esta média estiver acima de um limite (threshold) global, considera-se que o output reflete o template de forma significativa. Esta análise é utilizada para avaliar o output da LLM, indicando se o conteúdo gerado aproxima-se de forma significativa do template fornecido.

### 6.3 Análise de Off Topic

A análise de Off Topic é realizada pela função *off\_topic\_classifier*, utilizando o modelo facebook/bart-large-mnli para classificação "zero-shot". Esta função compara o texto com uma lista de tópicos indesejados, calculando uma pontuação de afinidade para cada tópico. Se algum tópico indesejado ultrapassa o limite predefinido (threshold), quer dizer que o texto de entrada possui esse tópico. Esta análise pode ser realizada tanto para o input do utilizador como para o output da LLM.

### 6.4 Análise de On Topic

A análise de On Topic ocorre pela função *on\_topic\_classifier* e é praticamente igual à análise de Off Topic, a única diferença é que esta recebe uma lista de tópicos desejados em vez de uma

lista de tópicos indesejados. Esta análise, tal como a análise de Off Topic, pode ser realizada tanto para o input do utilizador como para o output da LLM.

## 6.5 Análise de Prompt Injection

A análise de prompt injection ocorre com a função *detect\_prompt\_injection*, que utiliza o modelo protectai/deberta-v3-base-prompt-injection-v2, esta análise identifica tentativas de manipulação de prompt em sistemas interativos, como chatbots ou assistentes virtuais. Se o modelo detetar uma alta probabilidade de injeção (acima do threshold), a função alerta para uma tentativa de manipulação. Como este tipo de análise serve para garantir que interações com sistemas automatizados não sejam manipuladas, esta análise só se realiza ao input do utilizador.

## 6.6 Análise de Toxicidade

A análise de toxicidade é realizada pela função *detect\_toxicity*, usando o modelo unitary/toxicbert, esta função avalia o texto quanto à presença de linguagem tóxica ou ofensiva e retorna uma pontuação para toxicidade. Se a pontuação exceder o threshold, o texto é classificado como "contém toxicidade". Esta análise é direcionada para o output da LLM, permitindo a filtragem de linguagem ofensiva por parte da LLM.

## 6.7 Parâmetros e Configurações

No sistema descrito, o cliente possui flexibilidade para ajustar e configurar as análises conforme as suas necessidades específicas. Primeiramente, o cliente define o tipo de entrada, ou seja, se o texto em questão é um input de um utilizador ou um output gerado por uma LLM (modelo de linguagem). Além disso, o cliente pode selecionar quais as análises que deseja realizar, este ajuste permite que o sistema seja adaptado a diferentes cenários de uso, evitando análises desnecessárias e focando apenas nos aspectos mais relevantes para o contexto. Outro ponto de personalização é a configuração de templates específicos para a análise de prompt template, permitindo ao cliente estabelecer frases ou estruturas específicas que devem ser detectadas no texto. Por fim, para as análises de off-topic e on-topic, o cliente tem a possibilidade de definir listas personalizadas de temas que guiarão as classificações de off-topic e on-topic. Esta flexibilidade oferece ao sistema uma abordagem robusta e personalizada, atendendo às necessidades específicas de análise de cada cliente.

## 7. Estratégias de Escalabilidade

Os modelos de linguagem de grande escala (LLMs) oferecem um grande potencial para transformar vários setores, aplicações e serviços, o que tem resultado num aumento do seu uso por diferentes empresas e serviços em diversas áreas. Devido a isto, também surgem novos riscos que requerem atenção, nomeadamente, uma utilização inadequada dos mesmos e, por isso, neste âmbito, entra a API especificada neste relatório que atuará como middleware entre a solicitação de um utilizador e a LLM.

Com isto, à medida que a necessidade do uso desta API por diferentes utilizadores cresce e as escalas de carga de trabalho aumentam, aparece a necessidade de garantir que este serviço seja capaz de se adaptar ao aumento de utilizadores e lidar com mais cargas de trabalho, uma vez que sem uma estratégia adequada, o sistema pode enfrentar dificuldades para lidar com picos de tráfego repentinos, resultando num pior desempenho com atrasos, falhas ou até mesmo interrupções no serviço.

### Estratégias:

- **Load balancer para distribuir a carga por diferentes instâncias da API:** Uma estratégia possível seria adotar uma vertente que melhora a escalabilidade horizontal. Com este serviço instanciado num servidor na nuvem e acessível por quem pretendesse usar o serviço, seria de boa prática adicionar réplicas deste servidor à infraestrutura, resultando em mais instâncias disponíveis a serem usadas por mais utilizadores. Com mais recursos disponíveis, também surge a necessidade de distribuir estes recursos de forma eficiente, papel que será desempenhado por um load balancer.

Esta tecnologia atuará como intermediário entre o utilizador e a infraestrutura onde a API está instanciada. O load balancer analisa qual o melhor servidor para atender a solicitação de cada utilizador e, em seguida, encaminha a requisição para o servidor mais adequado. Desta forma, esta tecnologia distribui os recursos existentes de

- **Otimização do uso de GPU:** Como o sistema desenvolvido utiliza vários modelos de processamento de linguagem natural, modelos estes que podem ser carregados e descarregados para a GPU, podem tornar-se custosos num ambiente de alta demanda. Com isto em mente, procuramos métodos de combater um declínio no desempenho da aplicação, visando proporcionar a melhor experiência possível a todos os utilizadores.

- **Auto Scaling em Nuvem:** Com o sistema hospedado numa plataforma em nuvem, muitas soluções oferecem infraestruturas que se ajustam automaticamente à demanda. Estas plataformas monitorizam continuamente as condições do ambiente e os recursos, ajustando o número de réplicas e alocando GPUs para cada uma delas conforme necessário. Além disso, muitas dessas infraestruturas contam com uma estrutura Multi-GPU e de Distribuição em GPU, permitindo que o sistema balanceie a carga de trabalho de forma automática, direcionando novas requisições para GPUs menos ocupadas. Além disso, em sistemas de inferência em tempo real, o uso de várias GPUs possibilita o processamento simultâneo de múltiplas requisições.

## 8. Análise de Resultados e Testes

### 1. Teste ao modelo de Análise de Linguagem

De maneira a avaliar o desempenho do modelo de deteção de linguagem “xlm-roberta-base”, foram efetuados testes, utilizando o dataset “wecover/OPUS\_Tatoeba”, que contém vários textos e o respetivo idioma. Visto que o nosso SDK só vai receber inputs em português ou em inglês, foram selecionadas 1.000 amostras em inglês e 1.000 amostras em português para o teste, e a previsão da deteção da linguagem foi feita com base num threshold de 0,5. Os resultados foram os seguintes:

- **Accuracy:** 0.81
- **Precision:** 0.85
- **Recall:** 0.81
- **F1 Score:** 0.81

Estes resultados indicam que o modelo é equilibrado e confiável para deteção de idioma entre as duas linguagens escolhidas. Com um F1 score de 0.81, o modelo está a capturar bem ambos os idiomas, mantendo tanto a precisão quanto o recall num nível alto e平衡ado. No entanto, foi identificado que o modelo apresenta desempenho inferior em frases mais curtas, com menos de 5 palavras, evidenciado por resultados inconsistentes em certos casos. Esta limitação pode ser atribuída à falta de informações linguísticas suficientes em textos curtos, o que torna mais difícil a distinção precisa entre idiomas.

### 2. Teste ao modelo de Análise de Prompt Template

Os resultados são analisados com base em parâmetros predefinidos, com um threshold de relevância de 70% e uma análise utilizando n-grams de até 6 palavras. O valor 6 foi escolhido para os n-grams como um equilíbrio entre precisão e relevância. Um tamanho menor poderia resultar em correspondências mais curtas, aumentando o risco de falsos positivos, onde palavras comuns poderiam ser confundidas com partes do template. Com n-grams de 6 palavras, garantimos que apenas segmentos significativos do template sejam identificados, reduzindo deteções irrelevantes. Testes que não atingirem estes valores podem ser considerados não relevantes, no entanto, estes critérios podem ser ajustados conforme as necessidades específicas do cliente. Para avaliar a eficácia do nosso sistema de deteção de exposição de templates confidenciais, iremos apresentar três testes representativos. Cada teste simula uma situação distinta:

**Deteção Completa:** O output contém segmentos significativos do template confidencial, resultando numa elevada correspondência e superando os limites estabelecidos.

- Output da LLM: The model generated this response based on a confidential template that should have remained hidden. Confidentiality was compromised.
- Template: This is a confidential template that should remain hidden.
- Resultado da análise:

**Response:** Detected significant parts of the template in output: [{"phrase": "This is a confidential template that", "max\_similarity": "85.47%"}, {"phrase": "is a confidential template that should", "max\_similarity": "94.87%"}, {"phrase": "a confidential template that should remain", "max\_similarity": "91.12%"}, {"phrase": "confidential template that should remain hidden.", "max\_similarity": "86.66%"}]

**Overall similarity:** 89.53%

**Deteção Parcial:** O output apresenta apenas partes do template, mas a correspondência global poderá não atingir o limiar de significância.

- Output da LLM: The secret message must remain hidden from unauthorized people.
- Template: This is a confidential template that should remain hidden from unauthorized access.
- Resultado da análise:

**Response:** Detected parts of the template, but not significant enough.

**Overall similarity:** 58.31% (Not significant)

**Sem Deteção:** O output não contém qualquer elemento do template, demonstrando a capacidade do sistema em reconhecer outputs completamente originais.

- Output da LLM: The model provided an entirely original response with no reference to the template.
- Template: This is a confidential template that should remain hidden.
- Resultado da análise:

**Response:** No significant parts of the template detected in output.

**Overall similarity:** N/A

### 3. Teste ao modelo de Análise de On/Off Topic

Foram realizados testes para avaliar modelo de classificação on-topic/off-topic utilizando dados do conjunto 20 Newsgroups. Nestes testes, categorias relacionadas a ciência e tecnologia foram marcadas como on-topic, enquanto categorias como desporto e política foram definidas como off-topic. A classificação foi realizada com o modelo "facebook/bart-large-mnli", usando zero-shot classification, onde um texto é considerado on-topic se alcançar uma pontuação acima de 0.5 em temas como "ciência", "tecnologia" ou "inteligência artificial". Os resultados mostraram uma boa precisão e recall razoável, conforme detalhado a seguir:

- **Accuracy:** 0.70
- **Precision:** 0.73
- **Recall:** 0.94
- **F1 Score:** 0.82

Os resultados obtidos demonstram que o modelo apresentou um desempenho sólido na classificação on-topic/off-topic, especialmente em termos de recall (0.94), indicando uma excelente capacidade de identificar textos relevantes para os tópicos definidos. A precisão de 0.73 reflete um bom equilíbrio na correta identificação de textos on-topic, embora com algum espaço para melhoria na exclusão de exemplos fora do contexto. O F1 Score de 0.82 confirma uma boa consistência global do modelo, tornando-o uma ferramenta confiável para cenários onde a identificação de tópicos específicos é essencial, mas com potencial de ajustes para melhorar a precisão em contextos mais desafiantes.

#### **4. Teste ao Modelo de Análise de Prompt Injection**

Para avaliar o desempenho do modelo de deteção de injeção de prompt, "protectai/deberta-v3-base-prompt-injection-v2", foram realizados testes utilizando o dataset "jackhhao/jailbreak-classification". Este conjunto de dados contém 1.306 amostras, das quais 640 foram classificadas como "benignas" e 666 como "jailbreak". A previsão do modelo foi feita com um threshold de 0,5. Os resultados obtidos foram os seguintes:

- **Accuracy:** 0.91
- **Precision:** 0.99
- **Recall:** 0.83
- **F1 Score:** 0.91

Estes resultados indicam que o modelo possui um desempenho altamente confiável para a deteção de injeção de prompt, demonstrando uma excelente precisão (0,99), o que significa que a maioria das previsões de "jailbreak" foram corretas. O modelo também mantém um bom equilíbrio geral, com uma taxa de F1 Score de 0,91, garantindo eficácia tanto na captura quanto na exclusão de exemplos de injeção. Este desempenho torna o modelo adequado para uso em ambientes que exigem alta precisão na identificação de comportamentos indesejados.

#### **5. Teste ao modelo de Análise de Toxicidade**

Foram realizados testes para avaliar o desempenho do modelo de deteção de toxicidade "unitary/toxic-bert", utilizando o conjunto de dados OLID (Offensive Language Identification Dataset), focado na identificação de linguagem ofensiva em tweets. Os dados foram pré-processados para transformar as anotações 'OFF' (ofensivo) e 'NOT' (não ofensivo) em valores binários, com 1 para ofensivo e 0 para não ofensivo. Foram selecionadas 1.000 amostras ofensivas e 1.000 amostras não ofensivas para o teste, e a previsão de toxicidade foi feita com base num threshold de 0,5. Os resultados foram os seguintes:

- **Accuracy:** 0.75
- **Precision:** 0.89
- **Recall:** 0.57
- **F1 Score:** 0.70

Estes resultados indicaram que o modelo mostrou uma boa eficácia em classificar corretamente textos não ofensivos e uma precisão relativamente alta ao identificar linguagem ofensiva. No entanto, o recall de 0.57 revela uma certa limitação do modelo em identificar todos os textos ofensivos, refletindo em um F1 Score de 0.70. Estes resultados sugerem que, embora o modelo "unitary/toxic-bert" tenha um desempenho consistente na identificação de textos não ofensivos e uma boa precisão para a classificação de textos ofensivos, o modelo poderia beneficiar de ajustes adicionais para aprimorar a identificação de textos tóxicos de forma mais abrangente, especialmente em contextos variados de linguagem ofensiva.

## 9. Segunda Componente

Nesta secção, será explorado o processo de transformação do projeto inicial num produto comercializável. Serão analisados os diferentes aspetos necessários para monetizar o sistema, incluindo a definição de um modelo de negócio, bem como estratégias para entrada no mercado e de crescimento.

### 9.1 Modelo de Negócio

Um modelo de negócio é essencial para definir como uma empresa cria, entrega e captura valor. Este modelo ajuda a estruturar ideias, identificar o público-alvo, entender fontes de receita e custos, além de guiar decisões estratégicas e atrair investidores. Com base nisto, procedemos à criação do seguinte modelo de negócio referente ao nosso trabalho.

#### Proposta de Valor

Com o objetivo de destacar o nosso serviço e posicioná-lo como a melhor opção para o público-alvo em relação à concorrência, elaboramos a seguinte proposta de valor. Esta proposta busca reforçar todas as qualidades e vantagens que o sistema oferece, apresentando aos clientes os seguintes valores:

- Garantir segurança e conformidade nas interações com LLMs em diversos setores, tais como, financeiro, saúde, protegendo contra diferentes tipos de ataques, nomeadamente, tentativas de prompt injection, deteção de linguagem tóxica, prevenção de referências a tópicos não autorizados, entre outros.
- Fornecer uma camada de proteção reputacional, garantindo que conteúdos tóxicos, sensíveis ou inadequados não sejam gerados. Esta camada atua para preservar a imagem da organização, promovendo interações seguras e alinhadas aos valores éticos e normativos, além de fortalecer a confiança dos clientes nos serviços oferecidos.
- Disponibilizar uma ferramenta intuitiva que pode ser facilmente configurada e personalizada.

A solução inclui uma monitorização dos inputs fornecidos pelos clientes e os outputs gerados pelos modelos de deteção, garantindo um controlo rigoroso e preciso sobre as suas interações. Além disso, atende a necessidades estratégicas cruciais, como a proteção da reputação das marcas, a mitigação de riscos financeiros e o incentivo à adoção responsável de inteligência artificial. Com foco em segurança, escalabilidade e flexibilidade, o middleware posiciona-se como uma ferramenta indispensável para empresas que buscam utilizar LLMs com confiança e eficiência.

## Público-Alvo

O produto será destinado a empresas e organizações que disponibilizam e utilizam serviços baseados em LLMs, setores regulamentados e em áreas que exigem elevados padrões de conformidade, segurança e tecnologia. Em particular, a solução atende a casos em que a resposta gerada por um LLM a partir de uma solicitação inicial de um cliente é devolvida ao mesmo. Entre os principais públicos-alvo destacam-se empresas que utilizam chatbots como solução de atendimento ao cliente; empresas que disponibilizam assistentes virtuais baseados em inteligência artificial. O produto visa atender às exigências específicas destes segmentos, fornecendo uma camada adicional de controlo, confiabilidade e conformidade.

## Canais de Divulgação

Os canais de divulgação escolhidos visam maximizar a visibilidade e o impacto do produto no mercado-alvo:

- Website oficial com documentação, demonstrações e código do SDK em open source;
- Parcerias estratégicas com fornecedores de LLMs e provedores de nuvem;
- Participação em eventos e conferências de IA e segurança cibernética;
- Anúncios em diferentes plataformas e redes sociais (Linkedin, etc.);

## Fontes de Receita

Uma das fontes de receita possíveis seria, portanto, um modelo de assinatura que oferece acesso à plataforma em troca dos seguintes pagamentos mensais:

Plano	Preço	Requisições
Free Plan	€0/mês	Até 50 requisições/mês
Starter Plan	€49,99/mês	Até 50.000 requisições/mês
Enterprise Plan	Sob consulta com base nas necessidades	Personalizado conforme as necessidades do cliente

**Tabela 9.1:** Comparação de Planos

Além disso, também oferecemos aos clientes a opção de pagar conforme o uso dos recursos do serviço, com taxas variáveis. Assim, a plataforma torna-se mais acessível para clientes menores e também promove o incentivo para clientes maiores expandirem o uso do sistema.

## Custos

A estrutura dos custos da solução proposta reflete os investimentos iniciais necessários para desenvolver, manter e escalar o produto, garantindo a sua competitividade no mercado.

### Desenvolvimento do Produto:

- **Custos de desenvolvimento inicial:** Contratação de 2 a 3 pessoas com o intuito de desenvolver o sistema e ainda trabalhar no marketing do produto, pesquisa do mercado e prototipagem.

- **Manutenção e evolução contínua:** Atualizações para garantir compatibilidade com novas versões de LLMs. Adição de novas funcionalidades com base no feedback de clientes. Correção de bugs e melhorias na segurança do sistema.

#### **Infraestrutura Tecnológica:**

- **Servidores e hospedagem em nuvem:** Hospedagem do middleware em provedores de nuvem como AWS, Google Cloud ou Azure. Custos de armazenamento e processamento das requisições realizadas pelos clientes.

#### **Marketing e Aquisição de Clientes:**

- **Custos com campanhas publicitárias:** Investimentos em Google Ads, LinkedIn Ads e campanhas em redes sociais. Criação de conteúdo promocional, como vídeos e materiais gráficos.
- **Participação em eventos e feiras:** Inscrição e preparação para feiras de tecnologia e inovação.
- **Comissões de vendas e parcerias:** Percentuais pagos a integradores e consultorias que revendem ou recomendam o produto.

#### **Parcerias Estratégicas:**

- **Custo de administração de parcerias:** Negociação e manutenção de contratos com integradores, provedores de nuvem e fornecedores de LLMs. Desenvolvimento de materiais técnicos e de integração para parceiros.

Com isto em mente e após a realização da pesquisa, chegamos à seguinte estimativa do investimento inicial necessário:

- **Desenvolvimento de Produto:** €1.000-2.000/mês o salário de uma pessoa com as capacidades mencionadas acima. Inicialmente, com 2 a 3 pessoas empregadas ficaria €2.000-6.000/mês
- **Infraestrutura Tecnológica:** Uma plataforma para hospedar o serviço como também fornecer várias funcionalidades para escalar o produto, armazenamento, etc, corresponderia a uma estimativa de \$200-\$400/mês
- **Marketing e Aquisição de Clientes:** Uma estratégia com foco em Google Ads e LinkedIn Ads, criação de materiais promocionais, esforços para criar uma presença consistente em redes sociais e fortalecer a marca corresponderia a €4.000/mês
- **Parcerias Estratégicas:** Estabelecer e manter parcerias estratégicas, criar documentação técnica, programas de co-marketing seria uma estimativa de €3.000/mês

Em suma, numa fase inicial do projeto vemos a necessidade de investir um valor de pelo menos €10.000 por mês.

## **9.2 Estratégia de Entrada no Mercado**

A entrada no mercado será realizada de forma a maximizar a visibilidade do produto, assegurar uma adoção inicial sólida e estabelecer credibilidade junto ao público-alvo. Para isto, a estratégia será dividida em fases com ações direcionadas a mercados prioritários e à criação de uma base de clientes sólida.

## 1. Identificação do P blico-Alvo Priorit rio

Numa fase inicial e com o intuito de ganhar credibilidade e consolidar um espaço no mercado, pretendemos adotar uma abordagem orientada a desenvolvedores. O objetivo é atender às suas necessidades, interesses e desafios, apresentando produtos, ferramentas, serviços e tecnologias que facilitem o seu trabalho e melhorem a sua produtividade. Com esta base estabelecida, será mais acessível direcionar esforços para empresas que utilizam e oferecem serviços baseados em LLMs.

## 2. Parcerias Estrat gicas

Estabelecer colaborações com pessoas influentes no setor:

- **Provedores de LLMs:** Numa fase inicial, oferecer o produto como uma solução complementar, integrada aos seus serviços.
- **Fornecedores de Infraestrutura em Nuvem:** Parcerias com AWS, Azure e GCP para facilitar a distribuição e escalabilidade.

## 3. Plano de Marketing

A abordagem de marketing será baseada em:

- **Marketing de Conteúdo:**
  - Publicação de artigos técnicos e white papers sobre segurança em LLMs.
  - Participação em conferências para educar o mercado sobre os riscos associados a LLMs e as soluções disponíveis.
- **Plataformas Digitais:**
  - Criação de uma página web dedicada com informações claras sobre o produto, exemplos de casos de uso e documentação técnica.
  - Campanhas de LinkedIn Ads e Google Ads para alcançar gestores em empresas-alvo e ainda atingir diversas comunidades de developers a partir de outras plataformas como o Reddit.

## 4. Plano de Preços

A estratégia de preços será utilizada para atrair empresas de diferentes tamanhos:

- **Modelo Freemium:** Disponibilizar uma versão gratuita limitada para aumentar a adesão inicial.
- **Planos escalonados:** Oferecer assinaturas ajustáveis ao tamanho e às necessidades da empresa.

## 9.3 Estrat gia de Crescimento

A estratégia de crescimento do serviço, após estar implementada no mercado, será baseada em três pilares fundamentais: expansão de mercado, inovação contínua e fortalecimento da presença de marca. O objetivo é aumentar a adoção do produto, atingir novos públicos e consolidar a posição no mercado de soluções de segurança e conformidade para LLMs.

### 1. Expans o de Mercado

Foco em Diversificação Geográfica

- **Fase 1:** Consolidar a presença em mercados locais e regionais, com foco inicial em developers com dificuldades resolvidas pelo no produto e, posteriormente, empresas de médio e grande porte nos setores de tecnologia e serviços financeiros.
- **Fase 2:** Expandir para mercados regulamentados e setores de alta demanda, como saúde e jurídico que possuem maior maturidade no uso de LLMs.
- **Fase 3:** Penetrar em mercados emergentes adaptando o produto às necessidades locais.

## 2. Inovação Contínua

### Melhoria do Produto:

Investir em pesquisa e desenvolvimento para melhorar a precisão e a eficiência da deteção de ameaças, como novos tipos de ataques. Adicionar suporte para mais idiomas e aumentar a capacidade de personalização e desenvolver funcionalidades avançadas, como dashboards de análise em tempo real.

### Feedback de Clientes:

Criar um programa estruturado de recolha de feedback para entender as necessidades emergentes dos clientes e adaptar rapidamente o produto. Implementar uma estratégia de co-criação com os principais clientes para desenvolver novos recursos que atendam a casos de uso específicos.

## 3. Fortalecimento da Presença de Marca

### Ações de Branding

A produção de conteúdo técnico visa posicionar a marca como líder em segurança e conformidade para LLMs, utilizando artigos, white papers e webinars especializados.

### Eventos e Conferências

Aumentar a participação em eventos globais de inteligência artificial e segurança cibernetica para aumentar a visibilidade e atrair novos clientes.

### Comunidades e Advocacy

- Desenvolver uma comunidade ativa de utilizadores que possam compartilhar casos de sucesso e promover o produto.
- Criar um programa de parceiros e embaixadores para aumentar o alcance da solução em diferentes mercados.

## 10. Conclusão e Trabalho Futuro

O projeto desenvolvido atendeu às expectativas do grupo e entregou uma solução prática e eficiente, alinhada aos objetivos propostos. A API/SDK foi concebida com uma arquitetura modular e bem estruturada, garantindo facilidade de integração e utilização. Além disso, destaca-se pela flexibilidade na configuração e personalização, permitindo adaptações para atender a diferentes cenários e necessidades específicas.

A qualidade das análises foi satisfatória, considerando as limitações temporais do projeto, a utilização de modelos pré-treinados, aliada à tradução para inglês necessária em certos casos, possibilitou a obtenção de resultados consistentes. Apesar do impacto na performance devido ao processo de detecção de idioma e tradução, esta abordagem revelou-se uma solução prática dentro do contexto do projeto, garantindo compatibilidade com modelos mais avançados.

Foi também desenvolvida uma interface de demonstração que facilita a validação prática das funcionalidades implementadas, enquanto o código foi escrito de acordo com boas práticas de programação, assegurando clareza, manutenção e simplicidade no processo de deployment.

Para o trabalho futuro, será fundamental focar no desenvolvimento de modelos nativos que eliminem a necessidade de detetar o idioma do texto para análise e o traduzir. Esta abordagem permitirá que o SDK processe diretamente textos em português e outros idiomas relevantes, reduzindo a latência associada à deteção do idioma e tradução, e eliminando potenciais erros que possam impactar a precisão das análises.

Por fim, outro aspecto crucial será o aprimoramento contínuo das análises realizadas pelo SDK, que inclui adaptar o sistema para detetar ameaças emergentes e variações mais sofisticadas de ataques, como novas formas de prompt injection. A capacidade de atualização contínua será essencial para garantir que o SDK se mantenha relevante perante um panorama de ameaças em constante evolução.

## Bibliografia

- [1] Facebook AI Research (FAIR). *PyTorch: Tensors and Dynamic Neural Networks in Python with strong GPU acceleration*. 2016.
- [2] Inc. Docker. *Docker: The Open Platform for Distributed Applications*. 2013.
- [3] Hugging Face. *Transformers: State-of-the-Art Natural Language Processing for Pytorch, TensorFlow, and JAX*. 2016. URL: <https://huggingface.co/transformers/>.
- [4] Python Software Foundation. *Python-Levenshtein: C Extension for Fast String Matching*. 2021.
- [5] Google. *googletrans: Python Library to Interact with Google Translate API*. 2020.
- [6] Google. *SentencePiece: A Neural Network-Based Text Encoder*. 2018.
- [7] NLTK Project. *NLTK: Natural Language Toolkit*. 2001.
- [8] Pallets Projects. *Flask: A Microframework for Python*. 2010. URL: <https://flask.palletsprojects.com/>.
- [9] SeatGeek. *FuzzyWuzzy: Python Library for String Matching and Similarity Calculation*. 2020.