



Documento para Levantamento de Requisitos – Feedback Analysis (AluMind)

Equipe responsável pela demanda:

Lucas Alves (varela.alves@outlook.com)

Artefato 01: Classificar feedbacks

Requisitos Funcionais

- **RF001:** A aplicação deve ser capaz de receber feedbacks a partir de uma rota POST "/feedbacks;
- **RF002:** A aplicação deve ser capaz de armazenar os feedbacks recebidos em uma tabela para injestão na rota;
- **RF003:** A aplicação deve ser capaz de classificar esse feedback recebido como "POSITIVO", "NEGATIVO" ou "INCONCLUSIVO";
- **RF004:** A aplicação deve ser capaz de elencar as features sugeridas no feedback, em forma de lista, e definir um "code" para cada um, assim como uma razão para sua relevância;
- **RF005:** A aplicação deve ser capaz de armazenar o resultado final de sentimento em uma tabela "sentiments" com o mesmo identificador do feedback enviado;
- **RF006:** A aplicação deve ser capaz de armazenar cada "code"(ID) e "reason" extraído nas tabelas chamadas codes e reasons;
- **RF007:** A aplicação deve ser capaz de armazenar em uma tabela de relação a ligação entre um elemento da tabela de sentiments para vários elementos da tabela code;
- **RF008:** A aplicação deve ser capaz de armazenar em uma tabela a ligação entre um elemento da tabela code e um da reason.
- **RF008:** A aplicação deve ser capaz de retornar, na rota /feedback, um objeto com os campos: id, sentiment e requested_features[{code, reason}].
- **RF009:** Os services, routes, utils e models da aplicação devem possuir testes unitários.

Requisitos Não Funcionais

- **RNF001:** A aplicação deve ser capaz de processar e classificar pelo menos 1 feedback por segundo;
- **RNF002:** A aplicação deve se preocupar em se proteger contra ataque de injeção (SQL injection);
- **RNF003:** O código-fonte deve seguir padrões de codificação definidos e ser de fácil manutenção por outros desenvolvedores;





- **RNF004**: O endpoint “feedbacks” deve ser capaz de filtrar e negar feedbacks classificados como spam.

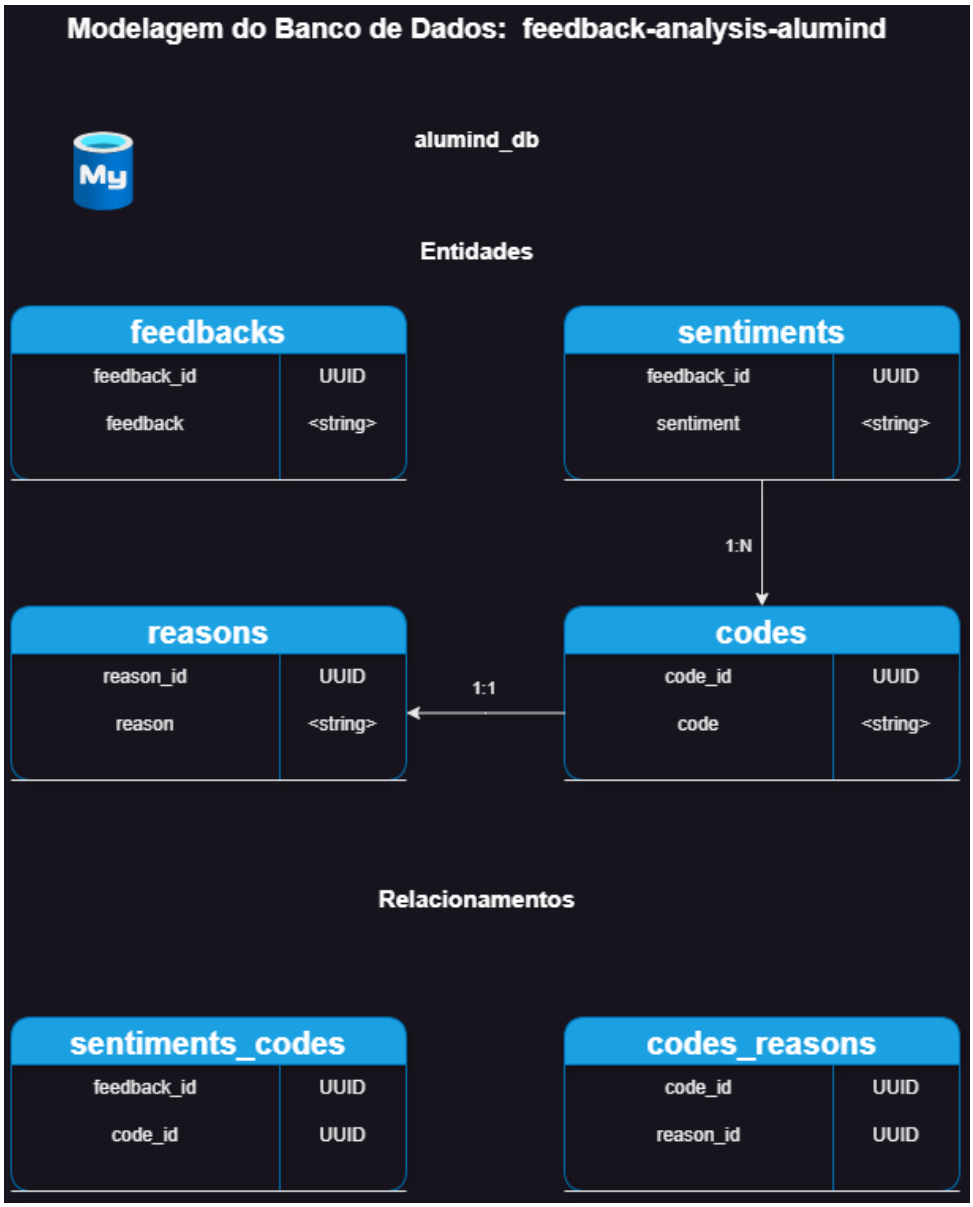
Artefato 02: Gerar um relatório web
Requisitos Funcionais • **RF001**: A aplicação deve ter uma página web para exibição do relatório de feedbacks; • **RF002**: O relatório exibido deve mostrar, em forma de tabela paginada, todos os feedbacks dados até o momento; • **RF003**: Na tabela exibida, devem estar presentes as colunas: id, feedback, status e uma para exibir mais detalhes; • **RF004**: A coluna “exibir mais detalhes” deve abrir um dialog com o feedback original e o objeto processado (sentiment, request_features[ ]) • **RF005**: O relatório deve conter um gráfico do tipo pizza para mensurar a quantidade de cada tipo de feedback; • **RF006**: O relatório deve conter um gráfico do tipo barras para contabilizar as features mais pedidas através dos feedbacks.
Requisitos Não Funcionais • **RNF001**: O código-fonte deve seguir padrões de codificação definidos e ser de fácil manutenção por outros desenvolvedores; • **RNF002**: A interface de usuário da página web de relatórios deve ser intuitiva e de fácil navegação para diferentes perfis de usuário (stakeholders).

Artefato 03: Criar um resumo semanal
Requisitos Funcionais • **RF001**: A aplicação deve gerar um um email a cada 1 semana e enviar para alguns stakeholders selecionados; • **RF002**: O email enviado ter um texto gerado com LLM; • **RF003**: O email enviado deve conter a % de feedbacks positivos e negativos; • **RF004**: O email enviado deve conter o Top5 de funcionalidades pedidas e os motivos para implementá-las.
Requisitos Não Funcionais • **RNF001**: O código-fonte deve seguir padrões de codificação definidos e ser de fácil manutenção por outros desenvolvedores; • **RNF002**: O email gerado deve ter um grau de formalidade aceitável para o tipo de comunicação.

Discussão de Requisitos

“Nessa tarefa, você pode considerar que outro time de desenvolvimento ficará responsável por agrupar e enviar os feedbacks para sua aplicação.”

- Como outro time é responsável pela coleta de feedbacks e a requisição da rota /feedbacks no documento já tem um id, conclui que esses dados podem estar hospedados em outros bancos ou até na nuvem. Logo, a tabela utilizada “feedbacks” não tem relacionamento com as demais.







“Cada Feedback é composto por um identificador e por um texto, que é o feedback em si próprio. Você deve criar um endpoint em sua aplicação que receba os feedbacks e classifique-os a partir do seu sentimento.”

- Como cada feedback recebido é um UUID e no retorno o mesmo UUID é enviado, nota-se que a análise de sentimento gerado no endpoint deve ser o mesmo UUID para ser possível relacionar essas duas entidades, que podem estar em bancos de dados separados (para este experimento, ambas estão no mesmo banco de dados por uma questão de praticidade).

“Além disso, cada feedback contém possíveis funcionalidades sugeridas. Cada funcionalidade sugerida tem um código que a identifica unicamente e uma descrição do porquê a funcionalidade é importante.”

- A criação de códigos que identificam cada funcionalidade não ficou clara como deve ser feita. Para o escopo deste projeto, sugeriu-se a identificação de códigos para o que foi sugerido, utilizando LLMs, e averiguação no banco de dados se já não existe o mesmo nome de codigo armazenado;
- O porquê de cada código também é armazenado como uma entidade separada e que se conecta de forma 1:1 com a tabela de códigos.

“O relatório pode ser uma tabela simples. [...] Além disso, gostaríamos de alguma forma conseguir acesso mais detalhado à feedbacks específicos, caso a gente julgue necessário.”

- Não ficou claro quais informações precisam ser exibidos na tabela, nem se ela precisa de mecanismos de filtro, etc. Logo, o aprimoramento dessa funcionalidade foi retida como melhoria futura, no intuito de cumprir todos os requisitos essenciais.

“Implementação de um sistema de filtragem no endpoint de envio de feedbacks que assegure que apenas feedbacks legítimos e não classificados como spam sejam processados e armazenados.”

- Seguindo a lógica de ingestão de dados por um rota secundária por outra equipe, é possível ter feedbacks do tipo spam no banco de dados (tabela feedbacks, que pode ser externa ao banco dessa API). No entanto, eles não serão analisados pela rota /feedbacks, pois ela faz essa filtragem antes de analisá-lo e inserir dados nas colunas do banco de dados que armazenam os dados da análise.