

Report - First Submission

Índice:

1.	Context description(Task 4.1.1)	1
2.	Initial Concept Model (Task 4.1.2) without generative AI:	2
3.	Generative AI Integration (Task 4.1.3):	3
3.1.	Prompt 1	3
3.2.	Prompt 2	3
3.3.	Resposta do ChatGPT-4o ao Prompt 1	4
3.3.1.	Considerações	6
3.4.	Resposta do Google Gemini 1.5 ao Prompt 1	7
3.4.1.	Considerações	8
3.5.	Resposta do Microsoft Copilot ao Prompt 1	9
3.5.1.	Considerações	10
3.6.	Resposta do ChatGPT-4o ao Prompt 2	10
3.6.1.	Considerações	11
4.	Final Concept Model (Task 4.1.4):	12

Context description (Task 4.1.1):

Nesta etapa do projeto, pretendemos modelar conceptualmente a base de dados de uma rede social semelhante ao Facebook em termos de funcionalidade.

Esta rede social permite aos utilizadores criarem contas pessoais e interagirem entre si através delas. Esta rede social é voltada para interações diretas entre utilizadores ou através de grupos, sem ter foco em influenciadores ou comunicação em larga escala.

Para criar uma conta, os utilizadores devem estabelecer o seu nome de utilizador, que deve ser único, e uma palavra-passe. Ambos devem ter pelo menos 5 caracteres. No momento da criação da conta, é necessário fornecer a data de nascimento para garantir que todos os utilizadores têm pelo menos 13 anos. O início de sessão pode ser feito através do email ou do nome de utilizador em associação com a palavra-passe. Após criar uma conta, os utilizadores podem definir uma foto de perfil para serem mais facilmente identificados, bem como definir as suas preferências de privacidade.

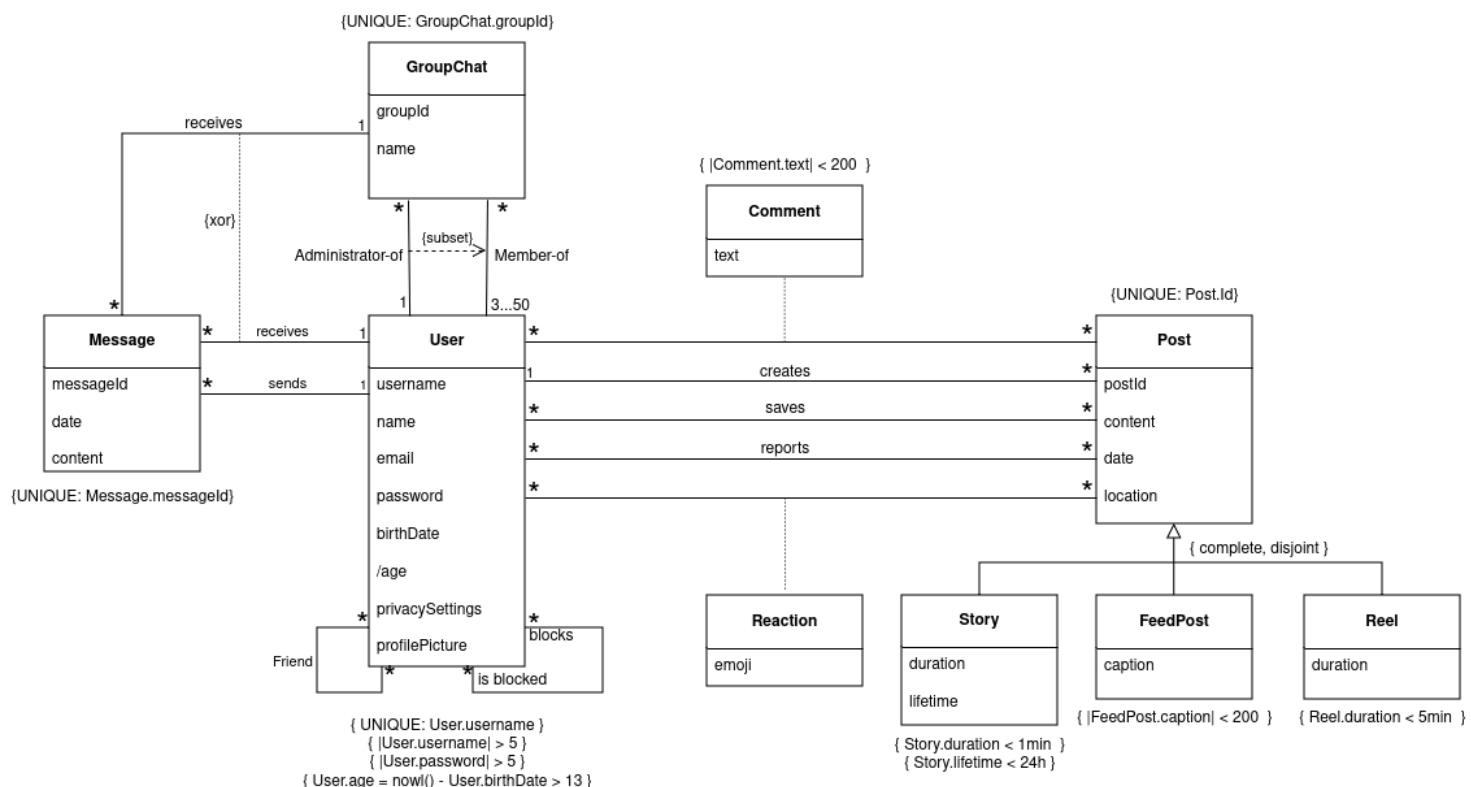
Tendo uma conta criada, os utilizadores podem começar a navegar. A rede social permite ver e criar publicações de carácter temporário ou permanente. A única publicação temporária é o Story, com tempo de vida inferior a 24 horas e duração inferior a 1 minuto. Quanto às publicações permanentes, podem ser do tipo Reel (vídeo com duração inferior a 5 minutos) ou do tipo FeedPost (publicação sem restrição de duração e com legenda com no máximo 200 caracteres).

Ao navegar, os utilizadores podem interagir com publicações e os seus criadores, quer por via de reações com base numa seleção de *emojis* predefinidos, ou de um comentário de texto com até 200 caracteres. Se um utilizador gostar de uma publicação e quiser voltar a vê-la mais tarde, tem também a opção de guardá-la na secção de favoritos. Caso considerem que uma publicação contém conteúdo sensível ou inadequado, os utilizadores podem reportá-la.

Para permitir o contacto de uma maneira mais próxima entre utilizadores, a rede social tem a funcionalidade de envio e receção de mensagens. Estas podem ser enviadas diretamente de um utilizador para outro, ou para uma conversa em grupo. Os grupos têm de ter pelo menos 3 participantes (senão seria apenas uma mensagem direta) e no máximo 50, tendo obrigatoriamente um só administrador, que será um dos participantes do grupo.

Dois utilizadores podem tornar-se amigos se quiserem ver as publicações um do outro mais frequentemente. Caso um utilizador não queira ver as publicações de outro ou se não quiser que outro veja as suas, pode bloqueá-lo.

Initial Concept Model (Task 4.1.2) without generative AI:



Este diagrama foi feito inteiramente sem recurso a IA generativa. Todas as associações, restrições, multiplicidades e elementos derivados estão presentes no diagrama de acordo com as convenções UML adotadas nas aulas teóricas da UC.

Generative AI Integration (Task 4.1.3):

Por não sabermos qual ferramenta de IA geraria melhores respostas sobre o tema de modelação conceptual em UML, escolhemos enviar o *prompt* 1 a várias ferramentas de IA generalistas: ChatGPT, Google Gemini e Microsoft Copilot. Desta maneira pudemos avaliar as diferentes ferramentas e juntar as melhores sugestões de cada ferramenta. De seguida, decidimos usar o ChatGPT-4o para o *prompt* 2 porque foi a ferramenta que considerámos ter dado melhor resposta ao *prompt* 1. Os *prompts* foram dados em inglês para garantir a melhor resposta possível por parte dos modelos de IA, que são principalmente treinados para responder nessa língua.

Prompt 1:

(o prompt inclui uma imagem do diagrama)

The attached image contains an UML diagram that aims to conceptually model the database of a social network similar to Facebook.

Your task is to look for potential mistakes or interesting new features. Pay special attention to errors in the multiplicities of the associations between classes, naming conventions and association/class structure.

Answer according to the current best UML practices.

Prompt 2:

In your previous answer you hinted at the idea of making the attribute `User.privacySettings` a class of its own. We want to accept that suggestion and want your assistance in taking that step.

The first part of your task is now to list and describe the attributes of the new class "PrivacySettings". Briefly explain the reason for each attribute.

The second part of the task is to validate if the multiplicity one to one is correct for the association between "User" and "PrivacySettings". If this is not the correct multiplicity, please help us understand why that is the case.

Resposta do ChatGPT-4o ao Prompt 1:

Upon reviewing the provided UML diagram for a social network database model, I've identified several points that could be improved and features that may be interesting to explore further. Let's break it down:

1. Multiplicities and Associations:

- GroupChat and User Association:

The association between GroupChat and User has the multiplicity "3..50" on the user side. This implies that a group chat must have at least 3 and at most 50 members, which may be valid in a specific case, but if the intent is to model general group chats (like in Facebook or WhatsApp), a lower bound of 2 (for two people to chat) might be more reasonable.

The Administrator-of relationship indicates that a User can be an administrator of exactly 1 group chat. This restriction feels limiting. It would be more flexible to allow users to administrate multiple chats (e.g., change multiplicity to "").*

- User and Message Association:

The {xor} constraint between the two receives associations might need clarification. If the intent is to distinguish between different message recipients (e.g., group vs. individual), it would be clearer to define distinct associations or explicitly document the purpose of the constraint.

*The multiplicity of Message to User is shown as * on the receiving side, which implies a message can be sent to multiple users. This makes sense in the context of group messages, but the constraint should be clear to distinguish between 1-to-1 and group messaging scenarios.*

- User and Post Association:

*The association creates between User and Post uses a multiplicity of 1 on the user side and * on the post side, which is fine. However, the multiplicity on the User side might better be shown as 0..1 if we want to allow for the possibility that posts can exist without a direct user creator (e.g., system-generated posts).*

- User and Reaction:

There's a direct relationship where a User creates a Reaction, but no association is shown between Reaction and the Post or Comment being reacted to. For a complete model, the Reaction should be linked to the content it's reacting to (whether it's a Post, Story, or Comment).

2. Naming Conventions:

- Field and Attribute Naming:

In Message, attributes such as date and content are generic. Consider renaming date to something more specific like sentDate or timestamp to improve clarity.

Similarly, the attribute age in User seems redundant given the presence of birthDate. Instead of storing age directly (which can change over time), it's generally better to compute it dynamically from birthDate. The { User.age = now() - User.birthDate > 13 } constraint already reflects this.

- Attributes in Reaction:

Reaction has a single attribute, emoji. While this works, if you anticipate supporting more complex reactions (e.g., likes, hearts, or custom text reactions), it might be worth renaming it to reactionType or adding flexibility for future features.

3. Class Structure and Inheritance:

- *Post Inheritance Hierarchy:*

The Post class has three subclasses (Story, FeedPost, and Reel) with a complete and disjoint constraint. This setup looks good for modeling different types of posts, but the distinctions between Story and Reel are somewhat unclear. Both have a duration attribute, and while Story has a lifetime, Reel doesn't specify such a parameter. Consider defining clear rules for content expiration to avoid overlap between these concepts.

- *Comment Class:*

The Comment class seems simple but isolated. In a typical social network, comments are often linked to posts or other comments, creating a recursive structure (i.e., comments on comments). You might consider adding a self-association or connecting Comment to Post or FeedPost to model threaded discussions or post comments.

4. Potential Enhancements:

- *Privacy Settings:*

User has a privacySettings attribute, but the details are unspecified. If the system is intended to reflect real-world social network features, privacy settings could be modeled more explicitly. You might include associations between privacySettings and Post, Message, or even ProfilePicture to reflect per-item visibility.

- *User Blocking:*

There's a blocks association indicating that a User can block other users. However, it might be worth considering adding additional relationships to reflect what happens when a user is blocked. For example, does it affect messaging, post visibility, or group membership?

5. Conclusion:

- *Fix Multiplicities: Clarify the GroupChat and User relationships and better define the Message receiving logic.*

- *Enhance Reaction and Comment Classes: Add associations to specify what content is being reacted to or commented on.*

- *Refine Naming and Constraints: Improve field names like age, and refine password/username constraints to ensure security.*

- *Expand Class Structures: Consider adding associations for privacy settings and user blocking behavior to reflect richer user interactions.*

Considerações:

A resposta do ChatGPT, apesar de ser a mais completa e útil, evidenciou a sua dificuldade em analisar certas partes do nosso modelo, parecendo desconhecer a utilização de classes de associação e tendo frequentemente interpretado mal algumas das multiplicidades de associações, sugerindo até a criação de associações já lá presentes. Para sermos mais objetivos, ignoramos as sugestões que nos pareciam mais forçadas e redundantes. Desta forma, apenas consideramos as recomendações abaixo.

A sugestão de reduzir o limite mínimo de membros de um grupo não será adotada para evitar redundância entre mensagens “um para um” e mensagens para conversas em grupo.

Escolhemos não adotar a sugestão de permitir que um post exista sem criador direto, já que na nossa rede social todos os posts estarão associados a um (e apenas um) utilizador.

Apesar de considerarmos subentendido que o atributo “date” na classe “Message” se refere à data de envio, reconhecemos que pode ser mais claro e por isso vamos mudar o seu nome para “sentDate”.

Em relação à classe de associação “Reaction”, a nossa intenção é que o atributo “emoji” possa representar diferentes tipos de reação, pelo que não sentimos necessidade de implementar a sugestão do ChatGPT-4o. Consideramos também que a diferença entre Story (publicação temporária) e Reel (publicação permanente) é clara.

Na nossa rede social, não pretendemos permitir comentários a comentários, pelo que não vamos implementar a sugestão de autoassociação da classe “Comment” como sugerido pelo ChatGPT-4o.

Concordamos que o atributo “privacySettings” na classe “User” não é claro e reconhecemos que podemos torná-lo mais detalhado; assim sendo, decidimos criar uma classe “PrivacySettings” para lidar com várias definições de privacidade de cada utilizador.

Quanto à funcionalidade de “Block”, acreditamos que deve ser implementada pelo programa de gestão da rede social, cabendo à base de dados apenas guardar informação acerca de quais utilizadores foram bloqueados ou bloquearam outros utilizadores.

Decidimos não restringir mais os atributos “password” e “username” para não criar entraves ao livre uso da rede social pelos utilizadores.

Resposta do Google Gemini 1.5 ao Prompt 1:

Analysis of the UML Diagram

Overall Assessment:

The UML diagram provides a solid foundation for modeling a social network. However, there are a few areas where improvements can be made to enhance clarity, consistency, and accuracy.

Specific Issues:

1. Multiplicity Errors:

- **GroupChat and Member-of:** The multiplicity of **Member-of** should be ***** on both sides to indicate that a group chat can have many members and a member can belong to many group chats.
- **Post and User:** The multiplicity of **User** in the **creates** association should be **1** as a post can only be created by one user.
- **Story and User:** The multiplicity of **User** in the **creates** association should be **1** for the same reason as above.
- **FeedPost and User:** The multiplicity of **User** in the **creates** association should be **1** for the same reason as above.
- **Reel and User:** The multiplicity of **User** in the **creates** association should be **1** for the same reason as above.

2. Naming Conventions:

- **Member-of:** This association name is somewhat ambiguous. Consider renaming it to **belongsTo** or **isMemberOf** for better clarity.
- **Reaction:** This class name is generic. Consider using a more specific name like **PostReaction** or **CommentReaction** to indicate the type of reaction.

3. Class Structure:

- **Post:** The **location** attribute might be better represented as a separate class **Location** associated with **Post** to allow for more complex location data (e.g., latitude, longitude, address).
- **User:** Consider adding a **createdAt** attribute to track when a user account was created.
- **Story and Reel:** These classes could potentially inherit from a common **Time-BasedContent** class to share attributes like **duration** and **lifetime**.

4. Missing Associations:

- **Comment and Post:** There should be an association between **Comment** and **Post** to indicate that comments can be made on posts.
- **Reaction and Comment:** Reactions can also be made on comments, so an association between **Reaction** and **Comment** is needed.

Interesting New Features:

- **Hashtag class:** To support hashtag searching and trending topics.

- **PrivacySetting class:** *To provide more granular control over user privacy settings (e.g., who can see posts, who can send friend requests).*
- **Analytics class:** *To track user behavior and engagement metrics.*

By addressing these issues and incorporating new features, the UML diagram can be improved to provide a more comprehensive and accurate model of the social network database.

Considerações:

Em regra geral, o Google Gemini também forneceu algumas sugestões válidas e pertinentes de uma forma mais sucinta, apesar de mostrar alguma dificuldade na leitura do diagrama, já que todas as recomendações que faz relativamente à multiplicidade de associações já lá constam. Tal como o ChatGPT, não reconhece as classes de associação. Eis as sugestões que achamos relevantes:

Concordamos que a classe “Reaction” tem um nome genérico e por isso vamos mudar o seu nome para “ReactionToPost”.

Quanto à estrutura das classes, decidimos não implementar nenhuma das sugestões (já que o atributo “location” será apenas uma string a indicar o local onde a publicação foi feita), não manter informação sobre o momento da criação de uma conta e desnecessário criar uma classe da qual herdaram Reel e Story apenas por partilharem o atributo “duration”.

Em relação à sugestão de associar “Comment” e “Post”, esta parece-nos melhor modelada pela atual estrutura em que um utilizador se associa a uma publicação através de um comentário. Como já foi referido, não pretendemos permitir reações nem comentários a outros comentários.

Concordamos que permitir o uso de “Hashtag” nas publicações seria uma funcionalidade interessante, mas consideramos que deve ser um atributo da classe “Post” e não uma classe por si só. Por esse motivo, vamos acrescentar o atributo “hashtag” na classe “Post”.

Como referido anteriormente, vamos criar uma classe chamada “privacySettings” para permitir um grau de detalhe maior na escolha das referidas definições. Consideramos que não faz sentido incluir uma classe “Analytics” já que o objetivo da nossa rede social é permitir aos utilizadores interagirem entre si de forma pessoal, e não comunicar para um grande público.

Resposta do Microsoft Copilot ao Prompt 1:

Here are some potential improvements and checks to ensure the UML diagram for the database of a social network is accurate and follows best practices:

1. **Multiplicity:**

- Ensure that the multiplicities (e.g., 1.., 0..1) on each association line accurately reflect the real-world relationships between entities. For example, a user can create many posts (1..) but each post is created by exactly one user (1).
- Verify that multiplicities are correct for bidirectional relationships. For example, a user can have many friends (0..*) and each friend relationship involves two users (1).

2. **Naming Conventions:**

- Use consistent and meaningful names for classes and attributes. Class names should be nouns (e.g., `User`, `Post`, `Comment`), and attribute names should be descriptive (e.g., `username`, `content`, `timestamp`).
- Avoid abbreviations and ensure names are intuitive for anyone reviewing the diagram.

3. **Association Structure:**

- Clearly define the direction of associations with arrows if they are unidirectional.
- Use association classes if an association itself has attributes. For example, if a "Friendship" association has a "since" date, represent it as an association class.
- Avoid unnecessary associations and ensure each relationship adds value to the model.

4. **Class Structure:**

- Ensure each class has a clear purpose and contains relevant attributes and methods. For example, the `User` class should have attributes like `username`, `email`, and methods like `login()`, `logout()`.
- Utilize inheritance where applicable. For example, if there are different types of users (e.g., `AdminUser`, `RegularUser`), consider using inheritance to model shared attributes and methods.

5. **Generalization and Specialization:**

- Use generalization (inheritance) to represent "is-a" relationships. For example, if `AdminUser` and `RegularUser` inherit from `User`, use a generalization arrow.
- Ensure that subclasses (specialized classes) add specific attributes or methods that are not present in the superclass (general class).

6. **Attributes and Methods:**

- Ensure attributes are named using camelCase or snake_case consistently.
- Specify the data types of attributes (e.g., `String`, `int`, `Date`).
- Include key operations (methods) within each class as needed.

7. **Documentation and Annotations:**

- Add notes or annotations to clarify any complex relationships or business logic.

- *Use stereotypes (e.g., <>, <>) to provide additional context where necessary.*

8. **Consistency and Clarity:**

- *Ensure the diagram is easy to read and understand. Avoid clutter by spacing out classes and associations.*
- *Use colors or styles to differentiate between various types of classes or relationships if it enhances readability.*

Considerações:

O Microsoft Copilot apresentou uma abordagem diferente em comparação ao Google Gemini e ao ChatGPT, focando-se maioritariamente na descrição técnica de como o diagrama deve ser feito e quais os cuidados a ter na sua sintaxe, do que na análise crítica do diagrama que enviamos, não tecendo comentários nem recomendações, que era o principal objetivo do nosso prompt.

Embora tenha demonstrado um conhecimento mais aprofundado sobre os diagramas em UML, já que referiu a utilização de classes de associação e de hierarquia, algo não reconhecido pelos outros modelos de inteligência artificial abordados, não nos foi apresentada informação relevante para alterar ou fazer adições pertinentes ao nosso diagrama.

A sugestão de incluir os métodos na classe não está de acordo com as normas da modelação conceptual em UML. Criar uma herança a partir de "User" para distinguir entre "AdminUser" e "RegularUser" não teria sentido, já que essa distinção existe apenas no contexto de conversas em grupo.

Sendo assim, o Microsoft Copilot foi a IA que menos contribuiu para o nosso projeto, apesar de termos podido aproveitar a sua resposta para analisar e rever toda a sintaxe e as convenções usadas num diagrama de UML.

Resposta do ChatGPT-4o ao Prompt 2:

*Here's a list of attributes for the new **PrivacySettings** class, which will encapsulate the user's privacy preferences. Each attribute serves a functional role in managing various aspects of user interaction and visibility on the platform:*

*Class: **PrivacySettings***

1. *profileVisibility: Defines the visibility of the user's posts. This can include options like "Public", "Friends Only", "Friends of Friends", or "Private".*
2. *locationSharingConsent: Determines whether the user allows their location to be shared in real time with other users. Options include "Always", "While Using App", or "Never".*

3. *directMessagePrivacy*: Specifies who is allowed to send direct messages to the user. Options might be "Everyone", "Friends Only", or "No One".
 4. *cookiesConsent*: Manages the user's preferences regarding the use of cookies for tracking and personalization. Possible values could be "Accepted", "Rejected", or "Custom" to allow selective cookie permissions.
 5. *taggingPermission*: Controls who can tag the user in posts, photos, or comments. Settings could be "Friends Only", "Friends of Friends", or "No One".
 6. *activityStatusVisibility*: Determines whether the user's online activity status (e.g., "last seen" or "active now") is visible to other users. Settings might include "Visible", "Friends Only", or "Hidden".
 7. *searchVisibility*: Specifies whether the user's profile can be found via search engines or internal search features within the app. Options might be "Searchable" or "Hidden".
-

Multiplicity of the *User-PrivacySettings* Association:

The one-to-one (1:1) multiplicity between *User* and *PrivacySettings* is correct. Each user is expected to have a unique and personal set of privacy preferences, so it makes sense for each *User* to be associated with exactly one *PrivacySettings* instance.

Considerações:

Alguns dos atributos propostos pelo ChatGPT-4o foram descartados. A seguinte lista de atributos descartados descreve os motivos para isso:

- *locationSharingConsent*: a nossa rede social não prevê a partilha de localização entre utilizadores.
- *taggingPermission*: a nossa rede social não permite "tagging" (identificação) de utilizadores em publicações ou comentários.
- *searchVisibility*: a nossa rede social não prevê motores de busca internos.

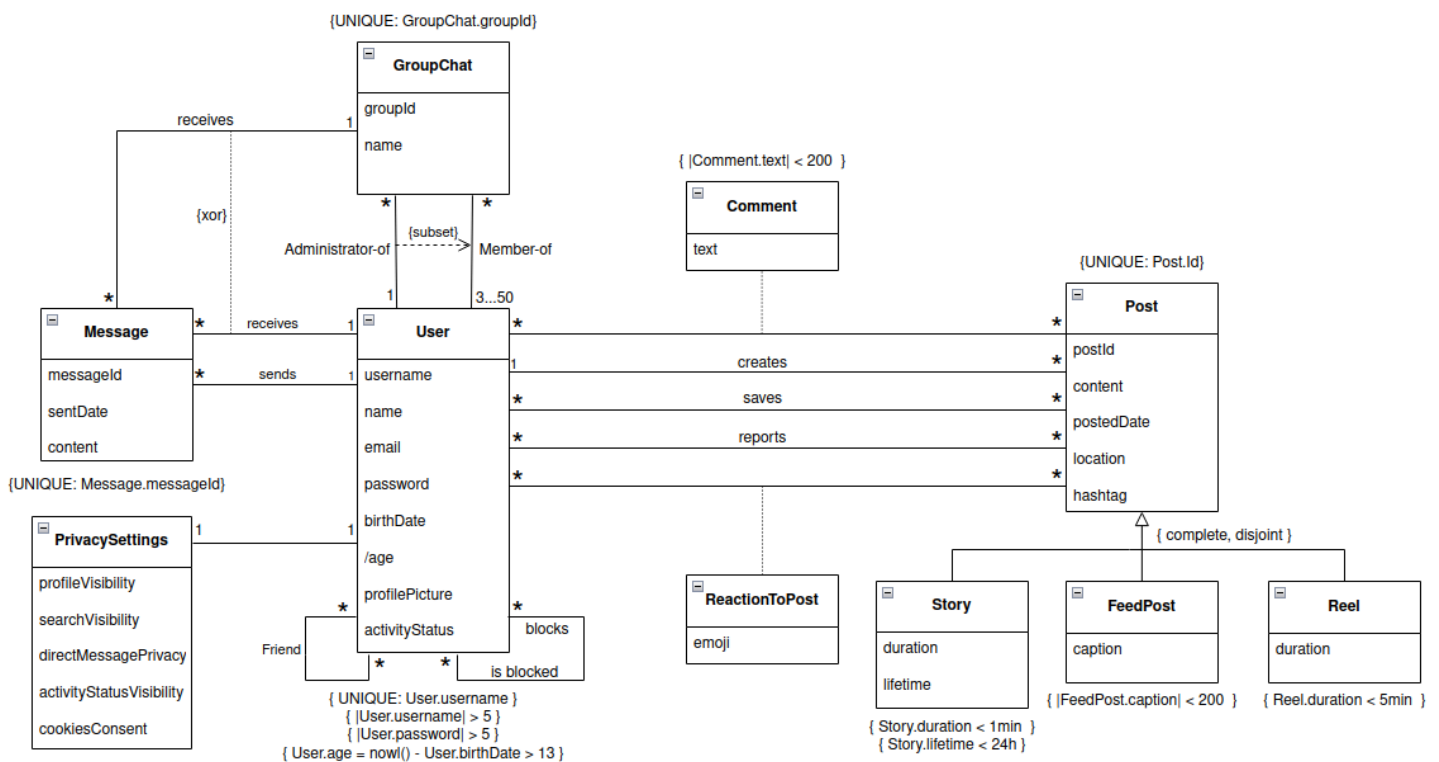
Decidimos adotar as restantes recomendações, o que implicou também incluir o atributo "activityStatus" na classe "User" para permitir que os utilizadores saibam quando outros estão ativos na rede social. Os atributos da classe terão as funções sugeridas pelo ChatGPT-4o.

Final Concept Model (Task 4.1.4):

As alterações que efetuamos com o auxílio de IA generativa foram:

- alteração do nome dos atributos “date” nas classes “Message” e “Post” para “sentDate” e “postedDate” para tornar mais clara a sua função;
- criação da classe “PrivacySettings” com os atributos referidos na resposta do ChatGPT ao *prompt* 2, o que implicou acrescentar o atributo “activityStatus” à classe “User”;
- alteração do nome da classe “Reaction” para “ReactionToPost” para clarificar o seu intuito.

As alterações são discutidas em maior pormenor na análise das respostas da IA generativa. Este foi o resultado do diagrama de acordo com todas as sugestões que achamos relevantes a partir da assistência dos diferentes modelos de IA:



Trabalho realizado por:

- António Lourenço Gomes Barrote Rodrigues, up202306331
- Gonçalo Miguel Ferreira Santos, up202306340
- Leandro Filipe Pais Resende, up202306343