📜 Requisitos da Aplicação

A implementação deverá seguir um conjunto de **requisitos funcionais** e **não funcionais**.

✅ Requisitos Funcionais (RF)

1️⃣ Cadastro de Usuário

- O sistema deve permitir que usuários se cadastrem informando um **nome de usuário e uma senha**.
- A senha deve ser armazenada de forma segura utilizando **hashing com bcrypt**.

2️⃣ Autenticação de Usuário

- O sistema deve permitir que usuários façam login informando **usuário e senha**.
- O login deve validar a senha comparando-a com o hash armazenado.
- Um **Token JWT** deve ser gerado após um login bem-sucedido.

3️⃣ Autorização com Token JWT

- O sistema deve exigir um **Token JWT válido** para acesso a funcionalidades protegidas.
- O Token JWT deve conter informações do usuário e ter **tempo de expiração configurável**.

4️⃣ Criptografia de Mensagens com AES

- As mensagens devem ser **criptografadas com AES** antes de serem armazenadas.
- A criptografia deve utilizar o **modo CBC (Cipher Block Chaining)** e um **IV aleatório** para cada mensagem.
- O usuário autenticado pode solicitar a descriptografia da mensagem.

5️⃣ Proteção da Chave AES com RSA

- Cada usuário deve possuir um **par de chaves RSA (pública e privada)**.
- A **chave AES usada na criptografia da mensagem** deve ser protegida com **criptografia assimétrica (RSA)**.
- Apenas o destinatário correto pode descriptografar sua chave AES para recuperar a mensagem.

6️⃣ **Validação e Expiração de Tokens JWT**

- O sistema deve validar **tokens expirados** e rejeitá-los automaticamente.
- Se um usuário tentar acessar o sistema com um token inválido ou expirado, o acesso deve ser negado.

❌ Requisitos Não Funcionais (RNF)

1️⃣ **Segurança dos Dados**

- As **senhas nunca devem ser armazenadas em texto plano**.
- A chave AES nunca deve ser transmitida sem **criptografia RSA**.
- O sistema deve utilizar algoritmos seguros e atualizados.

2️⃣ **Eficiência e Desempenho**

- O **tempo de autenticação e criptografia** deve ser otimizado para garantir **boa performance**.
- Os algoritmos implementados devem balancear **segurança e tempo de execução**.

3️⃣ **Facilidade de Uso e Manutenção**

- O código deve ser **bem documentado** para permitir fácil manutenção.
- O sistema deve permitir a **renovação de chaves RSA** em caso de comprometimento.

4️⃣ **Portabilidade**

- O sistema deve ser **compatível com diferentes plataformas** que suportam Python.
- O código deve ser modularizado para **possível expansão futura**.