# Liberum Oratio

## Descrição do Software

**Liberum Oratio** é um projeto de arquitetura de software fictício que possibilita aos seus usuários a manifestação de suas ideias e opiniões com total liberdade de expressão. A arquitetura do projeto consiste da mistura entre as arquiteturas space-based e peer-to-peer. A ideia por trás da escolha baseia-se em flexibar a independência dos usuários uns dos outros e do servidor para a maioria de suas conversas. A arquitetura space-based é utilizada em projetos onde a uma demanda por processamento ou armazenamento de grandes conjuntos de dados; contudo, não é intenção do projeto que a parte controladora da arquitetura space-based seja ativa, na verdade, ela é passiva e serve como *backbone* para o estabelecimento de comunicações com outros usuários da plataforma. A comunicação entre os usuários tentará ao máximo manter-se na arquitetura P2P para proteção de seus dados, mas é relevante constar que, quando os usuários não estão em LAN e o uso da rede é necessário, a parte orquestrante das unidades de processamento na arquitetura space-based, o componente *Main Software* mencionado nas próximas seções, utiliza-se do seu conhecimento dos endereços dos usuários para transmitir as mensagens diretamente, porém, a aplicação só manterá as mensagens em memória caso o outro usuário não esteja online, todas as mensagens são mantidas na memória dos dispositivos dos usuários e algumas possuem *time-out,* definido pelo o dono, que a apagará automaticamente quando expirar tanto do *Main Software* quanto do dispositivo do destinatário. Vale ressaltar que a aplicação viabiliza a formação de comunidades para troca de mensagens em grupo; para as comunidades, há a opção de torná-las privadas ou públicas, sendo estas encontradas na busca de comunidades na barra de busca da aplicação – assim como os usuários também podem ser contatados e, por meio dela, que a formação de amizades entre usuários é estabelecida –; as mensagens enviadas e não recebidas por usuários em comunidades são armazenadas em recipientes no servidor, que serão esvaziados quando o usuário entrar online.

A aplicação quer promover, ao máximo, a liberdade de expressão de seus usuários e manter, ao mesmo tempo, a proteção de seus dados; como a rede social visa não manter os dados de seus usuários em nuvem ou em seus servidores, procedimentos de limpeza são ativados quando há um time-out, as únicas informações mantidas, necessariamente, pela a aplicação são os dados do usuário, seus amigos e para onde enviar as mensagens, e as publicações do usuário em seu feed que não possuem time-out. A aplicação visa engajar seus usuários na seguinte ordem de relevância: manter a comunicação P2P ao máximo se possível em quanto os usuários estão em LAN; promover a limpeza de recipientes caso os usuários destino não estejam online a tempo de time-out; promover a inserção de usuários em comunidades tanto privadas quanto públicas como principal meio de estruturação de uma rede social, onde as mensagens estão armazenadas apenas nos dispositivos dos usuários da comunidade; e, com menor relevância para mensagens sem time-out, a promoção do uso de feed de usuário. Pode se dizer que para implementar a feature *liberdade de expressão – Liberum Oratio* de forma íntegra a aplicação prioriza: (1) o inacúmulo de dados, (2) a privativação de grupos se assim o desejarem, (3) a criptografia de mensagens ponta-a-ponta, (4) a inexistência do usuário ou de quão ativo este está na rede aos olhos de quem não o conhece, logo, ela favorece a invisibilidade dos usuários.O sistema possui basicamente 3 operações:

1. O usuário entra em sua conta em seu dispositivo móvel e pode juntar-se a comunidades ou procurar por amigos.
2. O usuário pode comunicar-se diretamente em LAN com algum outro usuário na mesma.
3. O sistema não mantém os dados de todas as conversas e as mensagens possuem tempo de expiração. Também, o histórico da conversa é mantido apenas pelos usuários nela envolvidos.

Detalhes adicionais:

* O usuário pode acessar o sistema por meio de dispositivos Android ou iOS. Também pode acessar por meio de um site.

## Diagrama de Casos de Uso

Diagrama

Descrição gerada automaticamente

Obs.: Para a melhor visualização das imagens, abra o documento .asta enviado em anexo.

## Requisitos Não-Funcionais (Atributos de Qualidade)

* **Log:** Manter o registro das ações feitas no servidor da aplicação.
* **Robusteza:** O algoritmo tem que ser funcional e performático.
* **Estabilidade:** É importante que a aplicação seja estável e que não afete o uso dos usuários que eventuais instabilidades.
* **Usabilidade:** A aplicação deve ser de fácil entendimento e uso ao usuário.
* **Proteção de ambiente:** O ambiente de servidor deve estar protegido de ameaças externas e de falhas de outros componentes presentes no projeto; bem como, os usuários deve estar protegidos de outros usuários, em termos de software, que não possuem nenhuma relação direta com ele.
* **Segurança:**  É importante para a aplicação que os seus usuários estejam seguros de controle externo e suas informações bem longe do alcance de pessoas não desejadas.
* **Backup:** É importante que a aplicação seja capaz de reiniciar a partir de seu último estado salvo em tempo hábil.

## Padrões Arquiteturais Escolhidos

* Arquitetura Space-Based
* Peer-To-Peer

## Arquitetura: Componentes

* Listar componentes
* Monta o diagrama com os componentes listados e o padrão arquitetural escolhido.
* Quantos programas no mínimo?
  + App iOS
  + App Android
  + Web App
  + Backend

Diagrama

Descrição gerada automaticamente

## Arquitetura: Componentes X Tecnologias

* Java, MySQL, Python, HTML, CSS, JavaScript e PHP.

# Análise Arquitetural

|  |  |  |
| --- | --- | --- |
| **Atributo de Qualidade** | **Descrição do Cenário** | |
| Cenário 1: Log | **Fonte** | Aplicação. |
| **Estímulo** | Muitas ações intercaladas. |
| **Artefato** | Server. |
| **Ambiente** | O desenvolvedor precisa manter o registro de ações ocorridas no servidor. |
| **Resposta** | Cada ação é registrada em um logger. |
| **Medida** | Automaticamente. |
| Cenário 2: Robusteza | **Fonte** | Aplicação. |
| **Estímulo** | Resistível a falhas de ambiente. |
| **Artefato** | Server. |
| **Ambiente** | Funcionamento normal do software mesmo quando algo indesejado ocorre. |
| **Resposta** | O sistema notifica os administradores de alguma situação. |
| **Medida** | Os administradores estudam os logs da aplicação e resolvem o problema. |
| Cenário 3: Estabilidade | **Fonte** | Aplicação. |
| **Estímulo** | Disponibilidade dos serviços aos usuários. |
| **Artefato** | Main Software. |
| **Ambiente** | Funcionamento normal do software. |
| **Resposta** | Analisar o tráfego de rede e tomar ação para melhorá-lo. |
| **Medida** | Administradores controlam o fluxo de conexões à aplicação principal. |
| Cenário 4: Usabilidade | **Fonte** | Usuário. |
| **Estímulo** | Facilidade de uso. |
| **Artefato** | ProcessingUnit / Client. |
| **Ambiente** | Funcionamento de fácil interação e aprendizado. |
| **Resposta** | Design de interfaces de usuário eficientes. |
| **Medida** | Designers de UX fazem a testagem das UIs. |
| Cenário 5: Proteção de Ambiente | **Fonte** | Aplicação. |
| **Estímulo** | Inviolabilidades de sistema. |
| **Artefato** | Main Software. |
| **Ambiente** | Funcionamento seguro da aplicação. |
| **Resposta** | Desenvolvimento de código seguro. |
| **Medida** | Durante desenvolvimento e testagem. |
| Cenário 6: Segurança | **Fonte** | Aplicação. |
| **Estímulo** | Privacidade dos usuários. |
| **Artefato** | Application. |
| **Ambiente** | Funcionamento seguro da aplicação. |
| **Resposta** | Desenvolvimento de mecanismos de validação. |
| **Medida** | Utilizado a todo momento pela aplicação. |
| Cenário 7: Backup | **Fonte** | Aplicação. |
| **Estímulo** | Desenvolvimento de aplicações *Always Up*. |
| **Artefato** | Main Software. |
| **Ambiente** | Funcionamento da aplicação contínuo. |
| **Resposta** | Manter o estado da aplicação mais recente e iniciá-la a partir dele caso haja uma falha. |
| **Medida** | Ativo sempre e utilizado em momentos críticos. |

|  |  |
| --- | --- |
| **Análise do Cenário 1** | |
| **Sumário** | Integração de um Logger para a aplicação. |
| **Objetivo de Negócio** | Permitir o conhecimento das ações tomadas pela a aplicação. |
| **Atributo de Qualidade** | Log. |
| **Abordagem** | Desenvolvimento de um componente de registros(logger) para a aplicação. |
| **Riscos** | Nenhum. |
| **Tradeoffs** | Tempo para a implementação do Logger pode interferir no desenvolvimento da aplicação. |

|  |  |
| --- | --- |
| **Análise do Cenário 2** | |
| **Sumário** | Dar qualidade ao funcionamento da aplicação. |
| **Objetivo de Negócio** | Assegurar os entes envolvidos da qualidade do software. |
| **Atributo de Qualidade** | Robusteza. |
| **Abordagem** | Desenvolvimento consciente e funcional da aplicação com práticas de clean code. |
| **Riscos** | Os desenvolvedores talvez demoraram mais. |
| **Tradeoffs** | Tempo de desenvolvimento. |

|  |  |
| --- | --- |
| **Análise do Cenário 3** | |
| **Sumário** | Garantir uma aplicação estável. |
| **Objetivo de Negócio** | Assegurar os entes da estabilidade da aplicação. |
| **Atributo de Qualidade** | Estabilidade. |
| **Abordagem** | Análise do status quo da aplicação e uso de desenvolvimento por meio de metodologias ágeis. |
| **Riscos** | Instabilidade e desagrado dos usuários. |
| **Tradeoffs** | Experiência de usuário em detrimento de um MVP – minimum viable product. |

|  |  |
| --- | --- |
| **Análise do Cenário 4** | |
| **Sumário** | Garantir facilidade de uso. |
| **Objetivo de Negócio** | Facilitar o uso da aplicação pelos usuários. |
| **Atributo de Qualidade** | Usabilidade. |
| **Abordagem** | Desenvolvimento de UIs por designers de UX. |
| **Riscos** | Demora de prototipagem e muitos ciclos de desenvolvimento. |
| **Tradeoffs** | Qualidade e performance da aplicação em detrimento de facilidade de uso geral. |

|  |  |
| --- | --- |
| **Análise do Cenário 5** | |
| **Sumário** | Garantir proteção aos componentes. |
| **Objetivo de Negócio** | Garantir que os componentes estejam protegidos de mal funcionamento ou funcionamento indevido de outros componentes da aplicação. |
| **Atributo de Qualidade** | Proteção de ambiente. |
| **Abordagem** | Desenvolvimento a exceções. |
| **Riscos** | Ineficiência. |
| **Tradeoffs** | Garantia de bom funcionamento em detrimento de máxima performance. |

|  |  |
| --- | --- |
| **Análise do Cenário 6** | |
| **Sumário** | Tornar segura a aplicação. |
| **Objetivo de Negócio** | Garantir aos entes envolvidos segurança e privacidade de seus dados. |
| **Atributo de Qualidade** | Segurança. |
| **Abordagem** | Implementação de mecanismos de segurança. |
| **Riscos** | Complexidade de desenvolvimento. |
| **Tradeoffs** | Segurança de uso da aplicação em detrimento de complexidade e performance. |

|  |  |
| --- | --- |
| **Análise do Cenário 7** | |
| **Sumário** | Manutenibilidade da aplicação. |
| **Objetivo de Negócio** | Garantir uma aplicação que está sempre de pé, mesmo quando falhas críticas ocorrem. |
| **Atributo de Qualidade** | Backup. |
| **Abordagem** | Salvar os estados da aplicação. |
| **Riscos** | Tamanho de backup possivelmente grande. |
| **Tradeoffs** | Desenvolvimento deste componente e implementação dele em detrimento de performance. |