MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Software Design Document

Professor: Sérgio T. Carvalho

Alunos:
Bruna Do Espirito Santo Sousa
Guilherme Abraão da Silva
João Gabriel Tavares Felix Monteiro
Karlla Loane Santos Lima
Luiz Felipe Pires Carvalho

1. Overview do projeto

1.1. Instrutor

Professor Sérgio Teixeira Carvalho

Áreas de atuação: Sistemas Distribuídos, Computação Ubíqua, Computação Aplicada à Saúde, Gamificação e Desenvolvimento de Jogos em Saúde

Contato: sergiocarvalho@ufg.br

1.2. Contextualização

A arte é uma expressão universal que transcende fronteiras e conecta pessoas de diferentes origens e culturas. O ArtFusion surge como uma plataforma digital inovadora que visa promover a colaboração criativa entre artistas de todo o mundo. Ao proporcionar um espaço virtual para a criação conjunta de obras de arte, o ArtFusion pretende redefinir a forma como os artistas interagem, inspiram-se e colaboram.

O ArtFusion é uma plataforma online de criação colaborativa de arte, projetada para artistas de todas as habilidades e estilos. Com o ArtFusion, os usuários podem criar, colaborar e compartilhar obras de arte digital em tempo real, conectando-se com outros artistas em todo o mundo. Com uma interface intuitiva e uma variedade de ferramentas de desenho e pintura, a plataforma permite que os usuários expressem sua criatividade de maneira flexível e dinâmica.

A plataforma ArtFusion oferecerá aos artistas a oportunidade de se conectar e colaborar com colegas de todo o mundo, ampliando suas perspectivas e inspirações. Ao explorar e colaborar em uma variedade de projetos de arte, os usuários podem encontrar inspiração constante e novas perspectivas criativas.

O ArtFusion será desenvolvido como uma aplicação web, acessível através de navegadores de internet modernos em diferentes dispositivos, como computadores, tablets e smartphones.

Principais Funcionalidades:

O foco inicial será na implementação das seguintes funcionalidades principais:

- Criação colaborativa de obras de arte em tempo real.
- Ferramentas de desenho e pintura intuitivas.
- Recursos para compartilhamento de obras de arte.

1.3. Objetivos do projeto

O ArtFusion tem como objetivo principal oferecer uma plataforma simples, intuitiva e segura para a colaboração em tempo real entre artistas, promovendo a interação e a inspiração mútua.

Os objetivos específicos do software incluem:

1. Simplificar a experiência de criação colaborativa de arte online, com uma interface simples e intuitiva para criação e compartilhamento de arte.
2. Fornecer um ambiente interativo e em tempo real que permita a colaboração entre artistas na criação de obras de arte digital.
3. Incentivar o engajamento do público com trabalhos de artistas emergentes ou com pouco alcance nas redes sociais, incentivando a descoberta e apreciação de novas perspectivas artísticas.
4. Promover a troca de ideias e inspiração entre os membros da comunidade artística
5. Servir como um repositório criativo para artistas e entusiastas interessados em explorar e compartilhar diversas formas de arte produzidas por colaboradores da comunidade ArtFusion.

Esses objetivos orientarão o desenvolvimento do ArtFusion, garantindo uma experiência enriquecedora e significativa para todos os usuários envolvidos na plataforma.

Discussões de relevância

- Definição de que o software será uma aplicação web acessível via navegadores de internet;
- Definição de que o software terá uma arquitetura principalmente categorizada com Publish-Subscriber;
- Reunião realizada em 13/04/2024 para definição das responsabilidades dos participantes da equipe;
- Reunião realizada em 13/04/2024 para levantamento das funcionalidades que o software ArtFusion irá disponibilizar aos seus usuários;
- Pitch realizado em 17/04/2024 para apresentação do projeto às partes interessadas.

1.4. Trabalhos relacionados

Na área de colaboração artística online, plataformas como o Canva[1] permitem a colaboração em tempo real, mas com limitações na criação de desenhos à mão, onde as alterações só são enviadas após o usuário soltar o mouse, o que pode limitar a interatividade e a fluidez da colaboração.

O projeto Collabio[2], semelhante ao ArtFusion, oferece uma experiência colaborativa com uma interface intuitiva e funcionalidades de gerenciamento de salas. No entanto, assim como o Canva, este projeto também só envia as atualizações após o evento de *mouse up*, não realiza a persistência dos dados e não possui mecanismos para tratamento de conflitos de concorrência, o que pode ser um desafio em cenários com múltiplos usuários ativos simultaneamente.

[1] https://www.canva.com/
[2] https://github.com/kriziu/collabio

1.5. Tecnologias do Projeto ArtFusion

O ArtFusion é desenvolvido com React no front-end para uma experiência de usuário fluida e responsiva. A escolha dessa tecnologia se justifica pela eficiência em renderizar interfaces dinâmicas e interativas, o que é ideal para nossa plataforma de arte colaborativa em tempo real.

No back-end, o ArtFusion é construído com Java utilizando o framework Spring Boot, juntamente com PostgreSQL para o gerenciamento de dados.

Adicionalmente, utilizamos o RabbitMQ como nosso middleware de mensageria. Ele é fundamental para gerenciar a comunicação assíncrona entre os usuários da plataforma, garantindo que todas as alterações de arte sejam transmitidas eficientemente em tempo real. O RabbitMQ nos permite implementar um modelo de publish-subscribe eficaz, que é essencial para distribuir as atualizações de arte para os usuários conectados sem sobrecarregar o sistema.

2. Requisitos

Neste tópico, detalhamos os requisitos funcionais e não funcionais do projeto. Ressaltamos que, devido ao tempo limitado para desenvolvimento, foram priorizados apenas os requisitos de alta e média prioridade. Dessa forma, garantimos que os aspectos mais críticos para o sucesso do projeto sejam abordados primeiro, assegurando funcionalidades essenciais e desempenho adequado.

2.1. Histórias de usuários

ID	História de usuário	Prioridade
HU01	COMO usuário, QUERO poder fazer login com meu e-mail e senha PARA salvar meus projetos e preferências	ALTA
HU02	COMO usuário, QUERO catalogar minhas criações por meio de tags para direcioná-las ao público e a interações específicas	BAIXA
HU03	COMO usuário, QUERO publicar meus esboços para compartilhá-los com outros usuários interessados	ALTA
HU04	COMO usuário, QUERO salvar meus esboços em um espaço privado para ser visualizado apenas por mim	ALTA

HU05	Como usuário, QUERO poder alterar o modo de visualização de minhas obras para disponibilizá-las para visualização ou mantê-las privadas	BAIXA
HU06	Como usuário, QUERO alterar o modo de interação de minhas obras para permitir ou restringir a colaboração de outros artistas	BAIXA
HU07	Como usuário, QUERO ser capaz de visualizar projetos de outros criadores por meio de um espaço de descoberta para colaborar com outras obras ou buscar inspirações	BAIXA
HU08	Como usuário, QUERO interagir com projetos de outros usuários para colaborar com obras e participar ativamente da comunidade da plataforma	ALTA
HU09	Como usuário, QUERO compartilhar projetos com outros usuários por e-mail para restringir o acesso apenas a artistas selecionados	ALTA
HU10	Como usuário, QUERO visualizar minhas obras criadas para revisar meu progresso e compartilhá-las com outros usuários da plataforma	ALTA
HU11	Como usuário, QUERO colaborar em tempo real com outros artistas na criação de obras de arte digital para facilitar a colaboração remota e simultânea	ALTA
HU12	Como usuário, QUERO que minhas alterações nos projetos sejam refletidas para outros colaboradores, para manter a sincronização e coesão entre os membros da equipe	ALTA

2.2. Requisitos funcionais

ID	**Requisito**	**Prioridade**
RF01	O sistema deve permitir que os usuários façam login utilizando e-mail e senha	ALTA
RF02	Os usuários devem poder se cadastrar no sistema fornecendo informações básicas, como nome, e-mail e senha	ALTA
RF03	Os usuários devem ser capazes de criar desenhos digitais usando ferramentas como pinceis, lápis, borracha e formas geométricas	ALTA
RF04	O sistema deve permitir a seleção de diferentes tamanhos de pincéis,	ALTA

	espessuras de linhas e cores	
RF05	O sistema deve permitir aos criadores de projeto convidar outros usuários específicos para colaborar em suas obras de arte, concedendo-lhes acesso com base em convites individuais	ALTA
RF06	O sistema deve permitir que os usuários colaborem simultaneamente em uma arte.	ALTA
RF07	Os desenhos criados devem ser salvos em formato de arquivo (por exemplo, PNG, JPEG)	BAIXA
RF08	O sistema deve permitir que os usuários explorem uma galeria de arte compartilhada para visualizar e editar obras de outros usuários	BAIXA
RF09	O sistema deve permitir que os usuários alterem a permissão de visualização da obra, podendo escolher entre opções como privado (visível apenas para o criador) e público (visível para todos os usuários da plataforma)	BAIXA
RF10	O sistema deve permitir que os usuários alterem a permissão de colaboração da obra, podendo escolher entre opções como privada (apenas o criador pode editar) e pública (qualquer usuário pode editar)	BAIXA

2.3. Requisitos não-funcionais

Tipo	**ID**	**Requisito**	**Prioridade**
Desempenho	**RNF01**	O sistema deve responder a alterações nas obras em 0,5 segundos	ALTA
Desempenho	**RNF02**	O sistema deve sincronizar as alterações entre os usuários em 1,5 segundos (considerando uma boa conectividade de internet)	ALTA
Desempenho	**RNF03**	O sistema deve suportar a colaboração em tempo real entre 5 usuários simultaneamente	ALTA
Usabilidade	**RNF04**	O sistema deve possuir uma interface intuitiva e de fácil utilização, permitindo que usuários de diferentes níveis de habilidade interajam de forma eficaz com as ferramentas de	ALTA

		desenho e pintura	
Usabilidade	**RNF05**	O sistema deve garantir que as edições feitas por diferentes usuários no mesmo projeto sejam sincronizadas de forma consistente e sem conflitos	ALTA
Confiabilidade	**RNF06**	O sistema deve estar disponível 99% do tempo	MÉDIA
Portabilidade	**RNF07**	O software deve ser compatível com os principais navegadores (Chrome, Firefox, Edge)	MÉDIA

3. Fundamentos de Sistemas Distribuídos e Concorrência relacionados ao projeto

3.1. Princípios de sistemas distribuídos

O ArtFusion utiliza conceitos fundamentais de sistemas distribuídos, destacando-se o uso do padrão publish-subscribe, filas de mensagens e sockets. No padrão publish-subscribe, os clientes se inscrevem para receber mensagens específicas, funcionando o servidor como um mediador que recebe e distribui os dados. Assim, quando um usuário faz uma alteração no canvas, essa alteração é transmitida a todos os usuários na mesma sessão que estão inscritos para receber atualizações.

Os sockets são utilizados para estabelecer uma comunicação de baixo nível entre os clientes e o servidor, permitindo um fluxo contínuo de dados. Essa abordagem assegura que as interações dos usuários sejam transmitidas rapidamente, minimizando a latência e melhorando a responsividade da aplicação.

Ainda, o ArtFusion emprega filas de mensagens dentro de sua arquitetura de sistemas distribuídos para gerenciar a troca de dados em tempo real entre os usuários de forma eficiente e confiável. Utilizando o RabbitMQ, uma implementação do protocolo Advanced Message Queuing

Protocol (AMQP), o sistema organiza e processa as alterações feitas no canvas, garantindo que sejam entregues em sequência correta e sem perdas. Esse mecanismo assegura a entrega de mensagens em ambientes de rede instáveis, armazenando temporariamente as alterações até que elas possam ser completamente sincronizadas.

A integração desses conceitos — publish-subscribe, sockets e filas de mensagens — confere ao ArtFusion a capacidade de suportar uma plataforma distribuída interativa e otimizada para a colaboração artística em larga escala.

3.2. Comunicação de sistema distribuído e mensageria

A figura a seguir (*Figura 1*) ilustra o fluxo de comunicação dentro do sistema distribuído utilizado pelo ArtFusion, focado na arquitetura de mensageria empregada. A representação detalha como os dados são transferidos entre diferentes componentes da plataforma, e como as filas são utilizadas para distribuir as atualizações de desenho em tempo real.

Os tipos de mensagens transitadas se dividem em mensagens gerais do Topic sala, que são as alterações propagadas (denotadas pelo vermelho) e mensagens específicas da Queue do usuário, que são as alterações que somente um usuário está demandando, como no momento de carregar a arte para uma máquina somente.

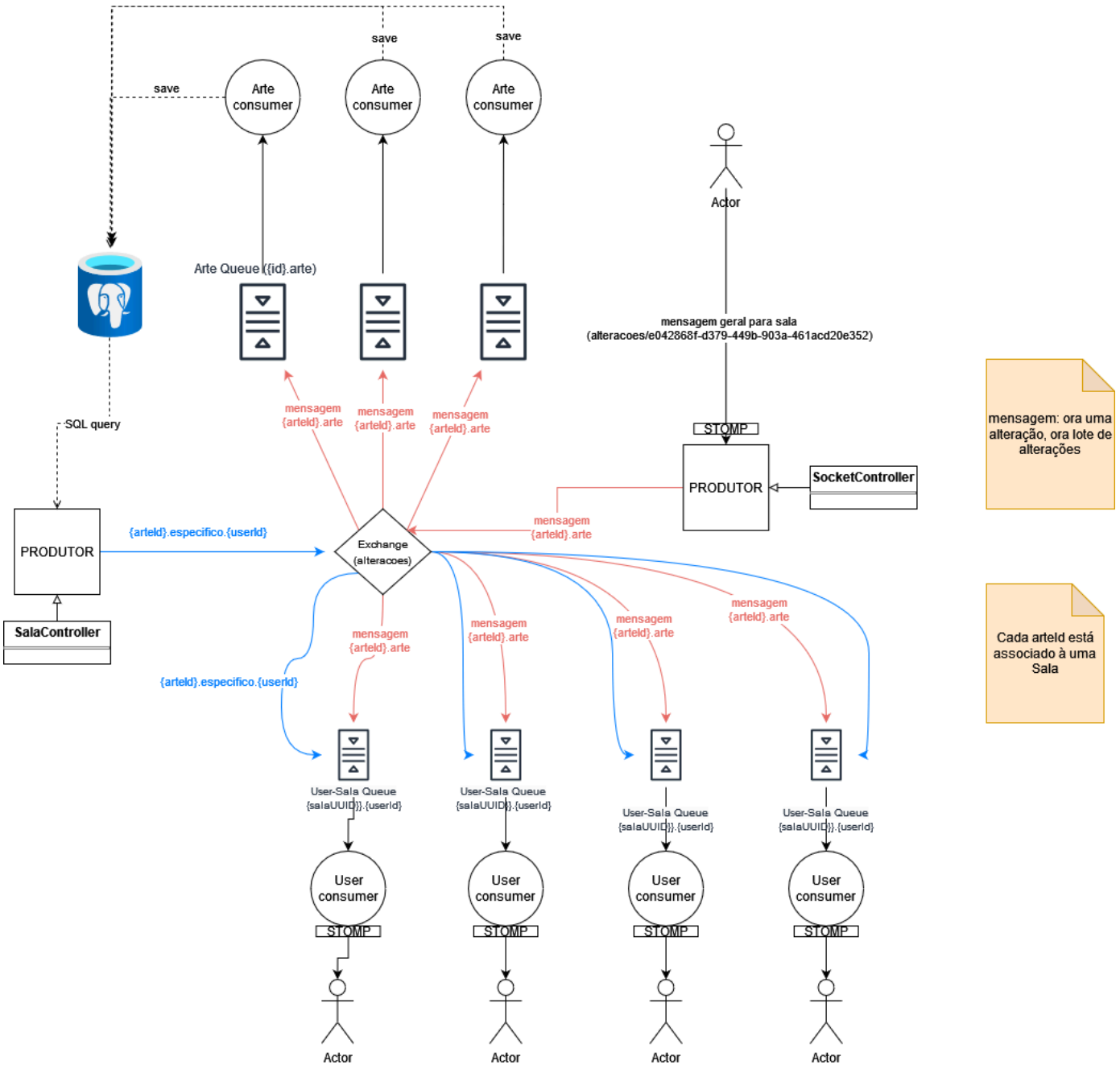

Figura 1

O diagrama de sequência a seguir (*Figura 2*) apresenta o processo de comunicação via socket e chamadas HTTP no ArtFusion. Ele ilustra a interação detalhada entre os clientes e o servidor, demonstrando como os clientes se conectam ao servidor e como as mensagens são enviadas e recebidas através do protocolo de sockets **STOMP**.

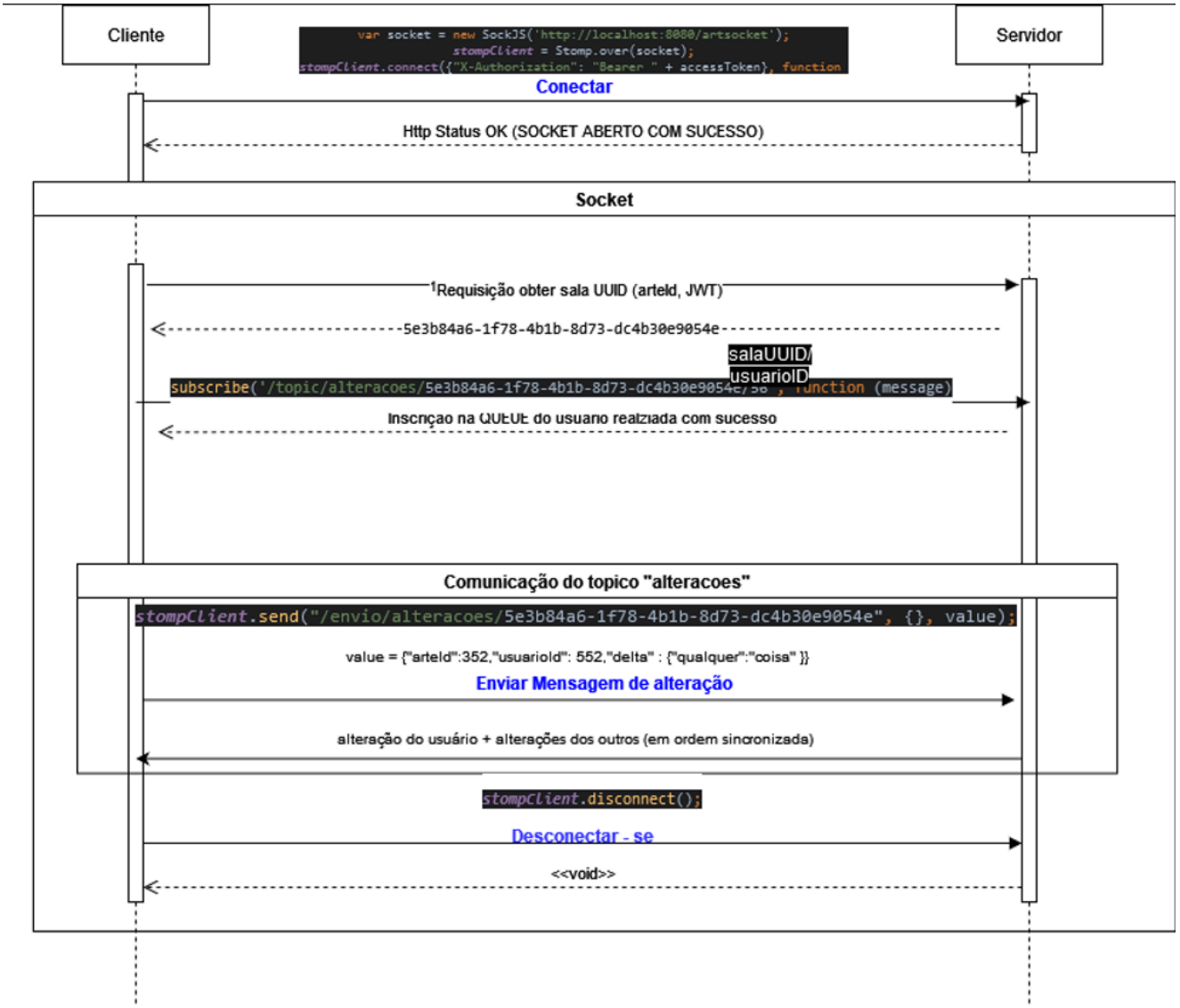

Figura 2

3.3. Robustez em sistemas distribuídos: coordenação, consenso, consistência e tolerância a falhas.

Diante à arquitetura distribuída, a tolerância à falhas e coordenação são fatores imprescindíveis para um bom software, em nosso sistema encontramos as seguintes problemáticas e soluções:

- **Desconexão do socket (tolerância à falhas)**: caso o usuário se desconecte do servidor durante uma sessão, o próprio lado do cliente tentará se reconectar, e automaticamente a arte é recarregada em sua versão mais recente, garantido a consistência dos dados da arte visualizada pelo cliente.
- **Replicação de alterações concisas:** cada alteração é antes processada pelo servidor para então ser refletida na tela do usuário, dessa forma é mantido um consenso entre todos os integrantes da sala quanto ao estado da arte, não importando as alterações ou o fato de estarem em máquinas distribuídas.
- **Limpeza de salas:** o servidor está equipado com um *Job* assíncrono que vigia salas "ide" (ociosas), caso não haja alterações durante 20 minutos, a sala é marcada para exclusão, dessa forma, mensagens em buffer são limpas do servidor e *leaks* de memória são evitados.

3.4. Princípios de Concorrência

Diante a problemática de múltiplas alterações em um mesmo desenho de forma simultânea, temos a concorrência em diversas partes do funcionamento do sistema.

- **Requisições de abertura de sala simultâneas**: em nossa arquitetura, uma arte pode ter somente uma sala ativa, e para abrir a sala, o usuário precisa requerer o servidor para a tal operação. Dito isso, para preservar a regra, a requisição concorrente de sala para arte precisou ser tratada com **Semáforos**, de forma que cada sala tivesse o seu próprio. Dessa forma, no momento de criação de sala, somente um usuário poderia criar por vez, por enquanto que os seguintes apenas conseguiam o UUID da sala já aberta
- **Alterações concorrentes entre si**: com a situação recorrente de múltiplas pessoas alterando o mesmo pixel, foi adotado o enfileiramento do Rabbit MQ padrão. Configuramos a prioridade de sobrescrição de pixel para para a alteração que chegou por último no servidor, pois o desenho se comporta como uma folha de arte, podendo

eu com o meu pincel passar por cima do que já foi desenhado. O mesmo se aplica à borracha.

- **Alterações concorrentes ao carregamento da arte**: ao carregar o estado da arte, um conjunto de alterações antigas da mesma é inserido na tela do usuário. Dessa forma, não teria sentido a arte sendo carregada passar por cima do desenho feito em tempo real. Assim, foi utilizado o enfileiramento com prioridade do Rabbit MQ, com as mensagens de estado da arte sendo consumidas primeiro (alta prioridade) e as mensagens de alterações recentes sendo jogadas para o final da fila de consumo do usuário. Assim, caso o usuário desenhe durante o carregamento da arte, suas alterações ficarão por cima do estado antigo da arte, mantendo dessa forma a integridade de dados.

3.5. Problemas de Conflitos no Front-end

Em um ambiente colaborativo onde múltiplos usuários desenham simultaneamente, utilizar um único canvas para todos os inputs, incluindo aqueles recebidos via socket de outros usuários, pode levar a problemas devido à forma como os pontos são processados e renderizados. Isso ocorre porque cada novo ponto adicionado, seja local ou remoto, continua a linha do último ponto registrado, independentemente de qual usuário o gerou. Como resultado, surgem conexões indesejadas entre os desenhos dos usuários, gerando linhas cruzadas e uma representação visual incorreta no canvas.

Para resolver este problema, adotamos a estratégia de utilizar dois canvas separados: um para o desenho local do usuário e outro para receber e exibir as atualizações via socket. Essa abordagem garante que o desenho local do usuário permaneça visível e responsivo, mesmo com potenciais atrasos de rede, pois é renderizado em um canvas dedicado. Simultaneamente, um segundo canvas principal é usado exclusivamente para as atualizações recebidas. Essa separação garante que a integridade visual da "arte final" seja mantida, refletindo as contribuições de todos os usuários conforme processadas e sincronizadas pelo servidor.

Experimentamos inicialmente outras soluções, como o uso de bibliotecas de gerenciamento de estado, onde cada ação no canvas seria monitorada como um elemento observável. As alterações, tanto locais quanto remotas, atualizam o estado desses elementos. No entanto, em ambiente de desenho colaborativo, com alto volume e frequência de alterações, isso gerou uma sobrecarga considerável no navegador, afetando a performance de renderização e escalabilidade da aplicação.

Portanto, o uso de um canvas secundário para desenhos locais se mostrou a opção mais eficaz e simples, otimizando tanto a experiência do usuário quanto a consistência e precisão visual entre todos os participantes da sessão colaborativa.

4. Resultados

4.1. Design arquitetural

A arquitetura do ArtFusion utiliza o padrão publish-subscriber (Figura 3). Esse padrão fornece um modelo de comunicação assíncrona que facilita a criação de aplicações.

O padrão pub-sub aplicado possui 4 componentes principais:

- Mensagens: Dados de comunicação enviados do remetente ao destinatário. Podem ter qualquer tipo de dados, desde que sejam expressos em texto.
- Tópicos: Cada mensagem possui um tópico associado. O tópico atua como um agrupador de tráfego de mensagens. Ele é expresso pelo UUID da sala.
- Queues: Cada mensagem de retorno volta para a Queue (fila) do usuário. Ela é expressa pelo UUID da sala e pelo ID do usuário
- Assinantes: É o destinatário da mensagem. Eles precisam se assinar em filas próprias
- Publicador: É o componente que envia mensagens. O publicador cria as mensagens e as envia para o Tópico. As mensagens do tópico são redirecionadas para as Queues associadas.

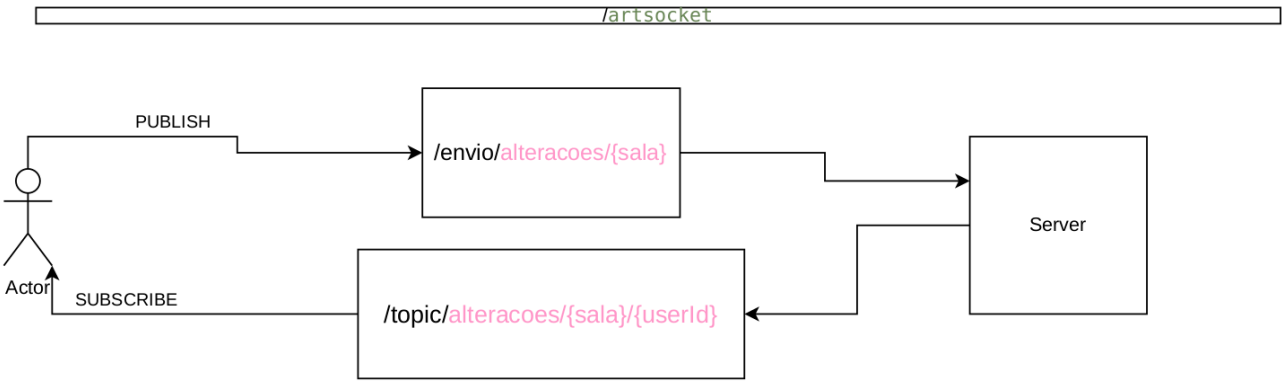

Figura 3

Arquitetura de Alto Nível

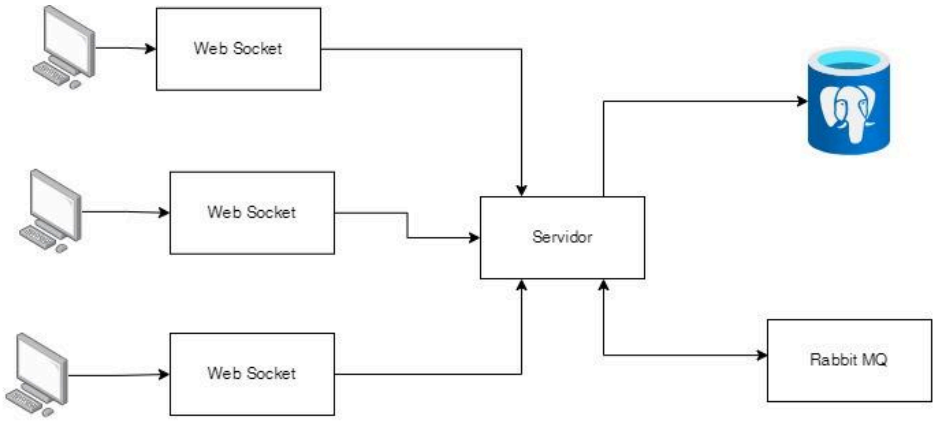

Figura 4

- Web Socket: Os clientes conectam-se ao servidor usando WebSockets. Esses WebSockets permitem comunicação em tempo real, essencial para colaboração simultânea em arte.
- Servidor: O servidor centraliza as conexões WebSocket e gerencia a comunicação entre os clientes. Ele também interage com o banco de dados e a fila de mensagens.
- PostgreSQL: Banco de dados usado para armazenar dados persistentes, como informações de usuários, projetos de arte e histórico de alterações.
- RabbitMQ: Sistema de mensageria que facilita a comunicação assíncrona entre diferentes partes do sistema e auxilia a resolução de concorrência através do enfileiramento.

Arquitetura do Servidor

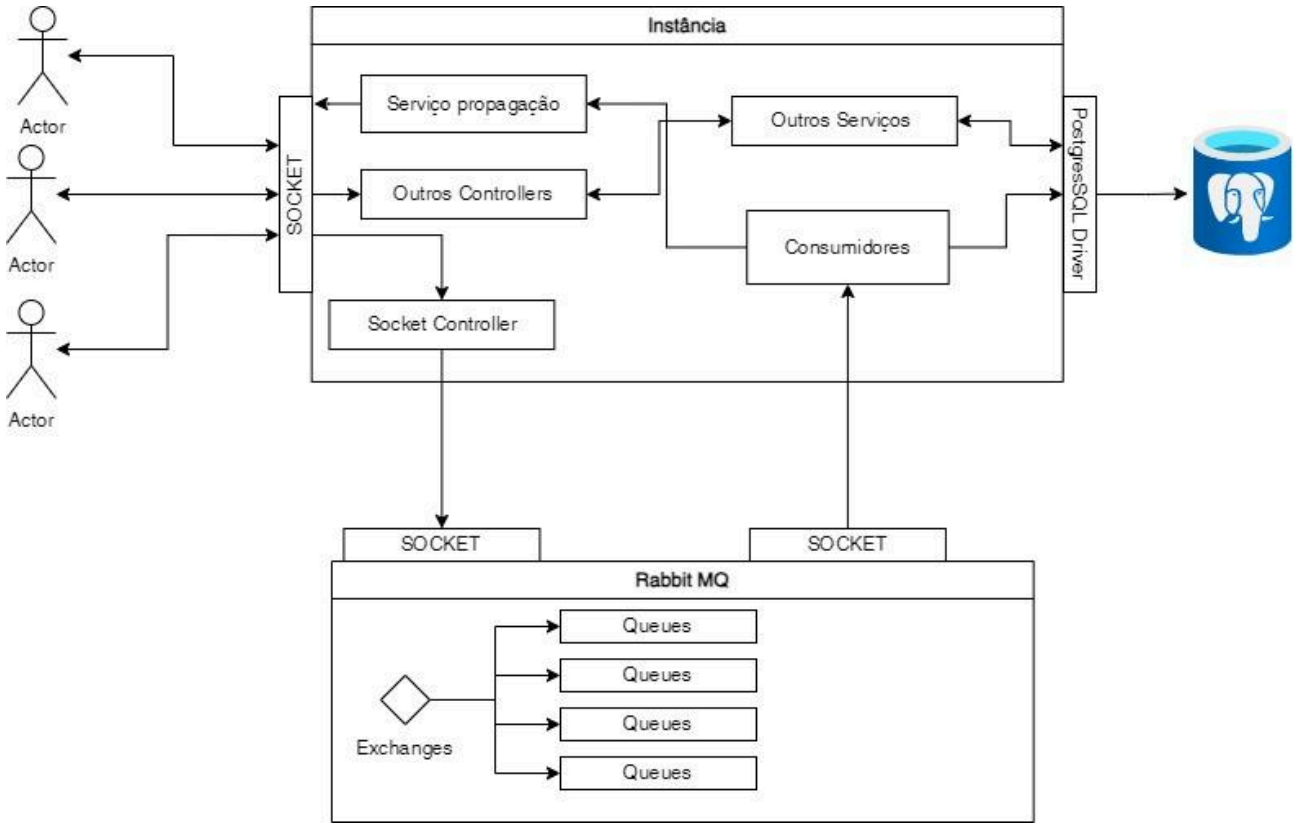

Figura 5

- Instância
    - Atores: Representam os usuários que interagem com a aplicação via WebSockets.
    - Socket Controller: Controlador responsável por gerenciar as conexões de WebSocket.
    - Serviço de Propagação: Propaga mensagens e atualizações para outros serviços e usuários conectados.
    - Outros Controllers e Serviços: Lidam com outras funcionalidades do sistema que não estão diretamente relacionadas aos WebSockets.
    - Consumidores: Serviços que consomem mensagens da fila RabbitMQ para processar e armazenar dados.
    - PostgreSQL Driver: Interface para comunicação com o banco de dados PostgreSQL.
- RabbitMQ
    - Exchanges: Distribui as mensagens para as filas apropriadas.

- Queues (do Rabbit MQ) : Filas onde as mensagens são armazenadas até serem processadas pelos consumidores.

4.2. Design dos dados no back-end

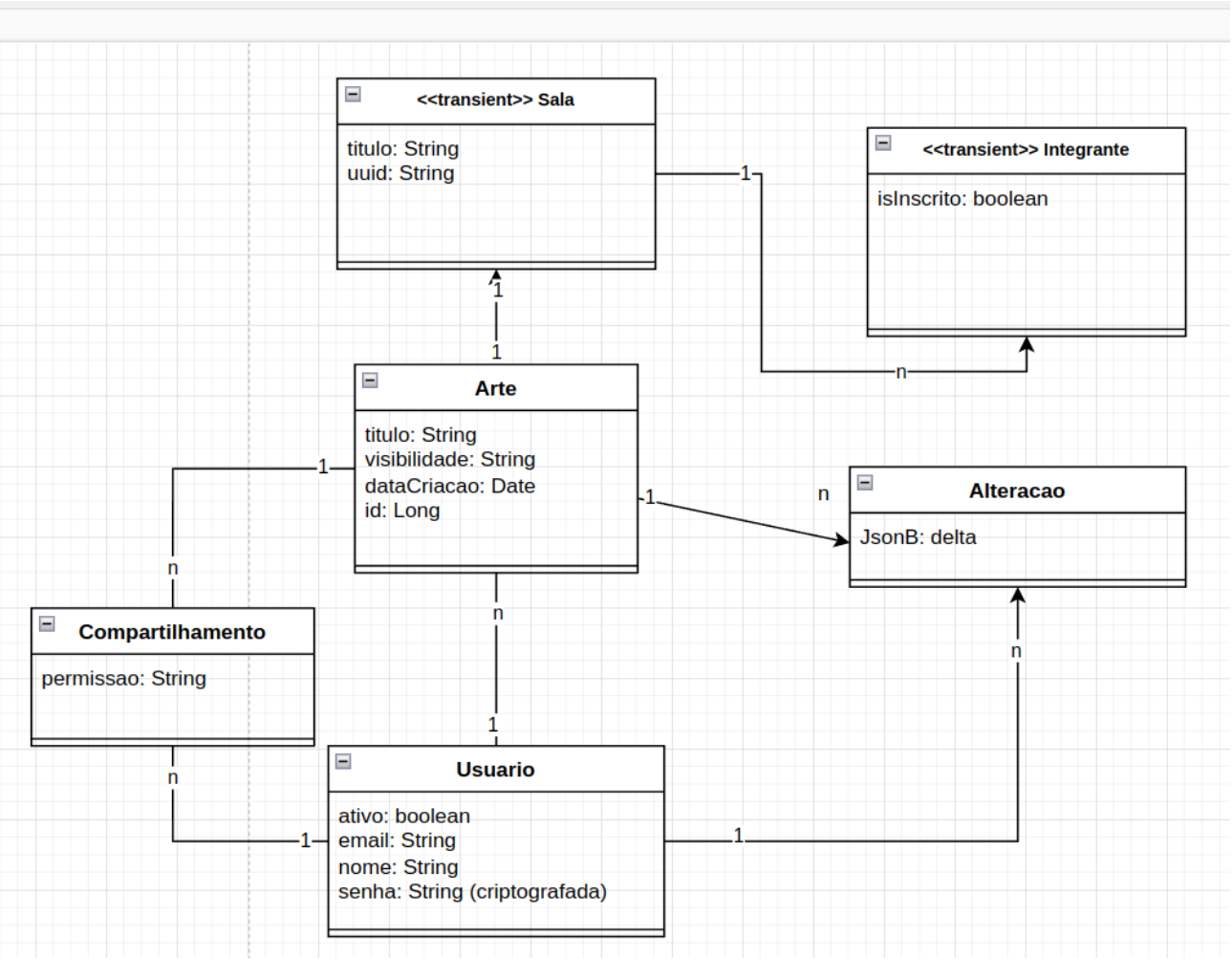

Figura 6

4.3. Manipulação de dados no front-end

No ArtFusion, o manuseio e a transmissão de dados entre o front-end e o back-end são cruciais para a funcionalidade de desenho colaborativo. A interface do usuário, desenvolvida em React, gerencia diferentes tipos de ferramentas de desenho que o usuário pode selecionar, como

círculos, linhas, quadrados e triângulos. A lógica para manipulação desses elementos é definida da seguinte forma:

- Para formas geométricas (círculos, linhas, triângulo e retângulo), o objeto `element` é construído com atributos que definem as coordenadas iniciais (`x1`, `y1`), as coordenadas finais (`x2`, `y2`), além do tipo de ferramenta, a largura da linha e a cor (*Figura 7*).

- Para ferramentas de desenho livre, lápis ou borracha, o objeto `element,` ao invés das coordenadas, mantém um array de pontos que representam os movimentos do mouse durante o desenho, além dos mesmos atributos de estilo.

```
element = {
      id: uuid(),
      x1: startPos.x,
      y1: startPos.y,
      x2: offsetX,
      y2: offsetY,
      type: toolType,
      lineWidth,
      color,
};
```

Figura 7

Quando uma alteração é feita, os dados do elemento são encapsulados em um JSON que inclui o `id` da arte, o `id` do usuário e o próprio elemento (`delta`), que é então transmitido via WebSocket para o servidor. Este formato de dados permite que o servidor processe e distribua as alterações para todos os usuários conectados à mesma sessão de desenho, garantindo que todos vejam as atualizações em tempo real.

5. Limitações, trabalhos futuros e perspectivas do Projeto

5.1. Limitações

As limitações do projeto incluem questões relacionadas à renderização em tempo real e a dependência de uma conexão de internet estável. Uma dessas limitações é a possível latência na visualização das alterações, pois o canva local, que reflete as ações imediatas do usuário, está

posicionado abaixo do canvas principal, que exibe as atualizações processadas pelo servidor. Em alguns casos, se houver atraso na comunicação via socket, os desenhos do usuário podem parecer atrasados em relação à posição atual do cursor, criando uma experiência de desenho menos fluida e responsiva.

Outra limitação significativa é a necessidade de uma conexão constante com a internet. O sistema não oferece suporte offline, o que impede os usuários de salvar suas alterações localmente para sincronização posterior.

Além disso, atualmente, o sistema exige uma infraestrutura de servidor robusta. Devido ao alto volume de dados gerados por múltiplas sessões de desenho simultâneas a plataforma requer significativa capacidade de memória e processamento.

5.2. Trabalhos futuros

Para os trabalhos futuros, planejamos abordar as limitações atuais e enriquecer a experiência do usuário na plataforma. As principais melhorias incluem:

- Melhorias na latência de visualização, buscando novas formas de otimizar a renderização do canvas.
- Funcionalidades de desfazer e refazer, permitindo que o usuário corrija erros ou reconsidere alterações no desenho.
- Implementação de suporte offline, permitindo que os usuários desenhem sem uma conexão ativa e sincronizem suas alterações com o servidor assim que houver conexão. Para isso, deverá, também, ser implementado novos algoritmos de resolução de conflitos.
- Introdução de um histórico de alterações, que permita aos usuários visualizar as alterações realizadas, incluindo quem realizou cada mudança, proporcionando maior transparência e controle sobre o processo colaborativo.
- Melhorar a escalabilidade da plataforma para lidar com um grande número de usuários e interações simultâneas.

6. Referências

AWS. Amazon Web Services, 2023. What is Pub/Sub Messaging? (O que são mensagens Pub/Sub?). Disponível em : <https://aws.amazon.com/pt/what-is/pub-sub-messaging/>. Acesso em: 22 de jul. de 2024.

RabbitMQ, Disponível em: <https://www.rabbitmq.com/>. Acesso em: 26 de jul. de 2024.

M. van Steen and A.S. Tanenbaum, Distributed Systems, 3rd ed., distributed-systems.net, 2017.