Transferência do Cuidado de Pacientes

Documento de Arquitetura de Software

Padrões de Arquitetura de Software - 2022/2 - UFG

Arthur Castro da Cunha
Felipe Silveira Schloegl
Joyce Beatriz Ferreira da Costa Silva

1. Introdução

1.1. Finalidade

A finalidade desse documento é definir a arquitetura e padrões arquiteturais a serem utilizados no aplicativo de Transferência de Cuidado dos Pacientes. Dentro do contexto de transferência será aplicado um contexto de web 3.0, implementando realidade aumentada no contexto médico. Ele é destinado às partes interessadas especialmente a Equipe técnica e gerentes de projeto.

1.2. Escopo

Este documento se baseia no documento de Especificação do Trabalho final para definir os atributos de qualidade a serem priorizados, estilos arquiteturais e representações das visões arquiteturais e seus componentes

1.3. Referências

Os documentos usados de referência para esse forma: a 42010-2022 - IEEE/ISO/IEC International Standard for Software, systems and enterprise - Architecture description e o documento de Especificação do trabalho final.

2. Contexto Arquitetural

2.1. Funcionalidades e Restrições arquiteturais

- RAS 1: Segurança - Requisito Não Funcional
- RAS 2: Permitir múltiplos acessos/comunicações - Requisito Não Funcional
- RAS 3: Baixa latência - Requisito Não funcional
- RAS 4: Alta transferência de dados - Requisito Não Funcional
- RAS 5: Comunicação em tempo real - Requisito Não Funcional
- RAS 6: Armazenamento de dados centralizado - Requisito de Dados

Com base nos requisitos de arquitetura de software podemos definir quais estilos de arquitetura serão definidos e quais requisitos eles irão favorecer. Os estilos arquiteturais definidos para esse projeto são cliente-servidor e camadas.

Os requisitos que levaram a escolha do **cliente-servidor** são: RAS 2, RAS 6 que são bem condizentes com as características dos servidores, podendo ser utilizado mais de um para permitir aguentar múltiplos acessos e podendo utilizar um processo de centralização de dados. Os requisitos RAS 3, RAS 4 e RAS 5, focam principalmente no desempenho, que condiz com o estilo cliente-servidor devido a sua fácil escalabilidade, possibilitando-se adaptar a demandas maiores.

O estilo arquitetural de **camadas** foi definido por priorizar a segurança sendo possível desenvolver uma camada focada em segurança, de forma a garantir a integridade das informações e evitar que as mesmas sejam vazadas ou alteradas. A

necessidade da priorização de segurança decorre da necessidade de manter as informações do paciente de forma íntegra e sigilosa

Tais estilos também favorecem a manutenabilidade por permitirem dividir o produto em componentes isolados, facilitando assim as alterações por poderem ser mais pontuais. Dessa forma os atributos que foram priorizados foram eficiência, segurança e manutenabilidade.

3. Representação da Arquitetura

Conforme descrito no tópico anterior a arquitetura será híbrida utilizando conceitos de camadas e cliente-servidor e prioriza os atributos de qualidade de eficiência, segurança e manutenibilidade.

Para representar a arquitetura será usado os pontos de vista do projetista, desenvolvedor e implantador. As visões usadas serão de desenvolvimento, lógica, segurança, física e casos de uso

Ponto de Vista	**Visão**	**Diagramas**
Projetista	Desenvolvimento	Componentes
Desenvolvedor	Lógica	Classes
	Segurança	Pontos-fracos
Implantador	Física	Implantação
	Casos de uso	Casos de Uso

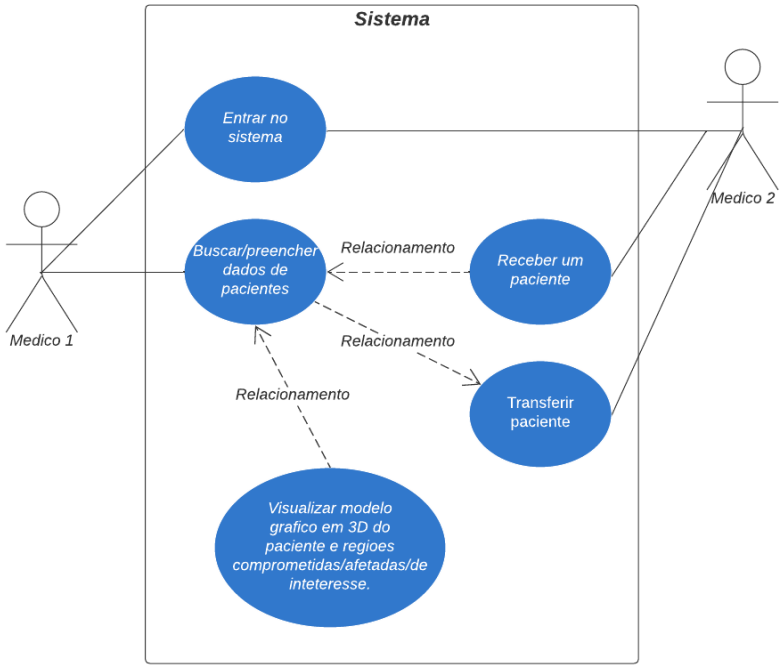
4. Ponto de vista dos Casos de Uso

4.1. Descrição

O diagrama de caso de uso resume os detalhes dos usuários do seu sistema e as interações deles com o sistema.

4.2. Visão de Casos de Uso

1) Gestão de usuários
2) Gestão de pacientes
3) Transferência de pacientes
4) Visualização 3D do paciente com detalhamento em seus problemas



Recurso: Lucidchart

5. Ponto de vista do Projetista

5.1. Visão geral

O ponto de vista do projetista tem como objetivo definir as principais partes que o compõem, tal como os componentes, além de definir quais as suas responsabilidades. Foi escolhida por ser uma visão primordial para a compreensão do software e de todo o seu ecossistema.

O modelo arquitetural proposto para a construção deste software será composto por 3 (três) componentes essenciais: Um cliente, um servidor e um componente responsável por gerenciar os dados utilizados no software.

5.2. Visão de componentes

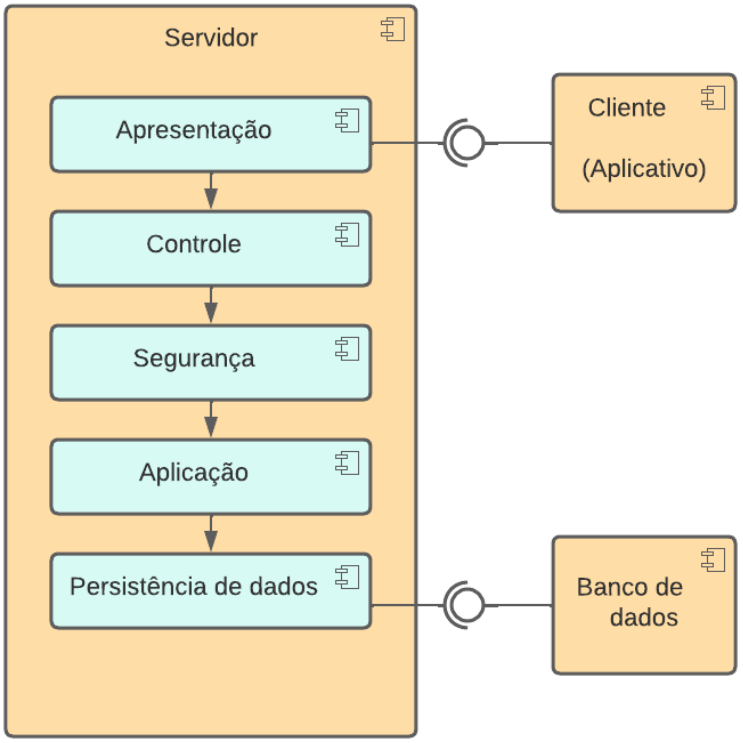
O componente cliente é a representação dos navegadores e dispositivos móveis dos usuários e a aplicação que será executada neles. Tem como papel interagir diretamente com os usuários e realizar requisições ao servidor. Ele terá

uma camada de segurança representada pela autenticação e pela criptografia dos dados.

O componente servidor é responsável pela implementação lógica do programa, receber as requisições dos clientes e realizar comunicação com o banco de dados. O componente será dividido em camadas, para poder segregar as funcionalidades e criar camadas de segurança, cada camada só poderá ser acessada pela sua camada anterior. Com exceção de camadas que recebem requisições de cliente ou realizam requisições ao banco de dados que poderão realizar/receber tais requisições.

O componente de gerenciamento de dados representa o banco de dados e os programas relacionados que visam armazenar e fornecer as informações necessárias para o funcionamento do servidor e cliente.

A interação dos componentes citados se dá com base no diagrama a seguir:



Recurso: Lucidchart

5.3. Detalhamento das Camadas

O componente servidor será dividido:

A camada de apresentação é uma que recebe as requisições externas do cliente, recebe requisições http do usuário e fornece endereços de acesso para o cliente, assim como realiza a comunicação com a camada de controle

A camada de controle serve para poder realizar a distribuição e redirecionamento das requisições passadas pela camada de apresentação para os determinados componentes da camada de segurança

A camada de segurança será responsável pela verificação da completude da informação e pela descriptografia da mesma. Ela se comunica com a camada de aplicação

A camada de aplicação é responsável pelo processamento lógico e implementação das regras de negócio. Ela se comunica com a camada de persistência de dados.

A camada de persistência de dados é a camada responsável por realizar as requisições ao componente de gerenciamento de dados e retorná-los à camada de aplicação.

As camadas permitem uma segmentação do projeto em componentes menores de micro-serviços que podem ser substituídos e alterados com maior facilidade e que realizam funções específicas, satisfazendo a característica de manutenibilidade.

A camada de segurança e as restrições de conexão entre as camadas permitem se adequar com a característica necessária de segurança por tornarem os acessos mais restritos e implementando mais validações para os mesmos.

6. Ponto de vista do Desenvolvedor

6.1. Visão geral

O ponto de vista do desenvolvedor é direcionado aos projetistas e desenvolvedores do software e tem como objetivo definir as principais partes responsáveis por definir as funcionalidades e restrições do software, tal como as classes.

6.2. Visão lógica

A visão lógica está preocupada com a funcionalidade que o sistema fornece aos usuários finais. Os diagramas UML são usados para representar a visão lógica e incluem diagramas de classes e diagramas de estado.

6.2.1. Detalhamento das classes

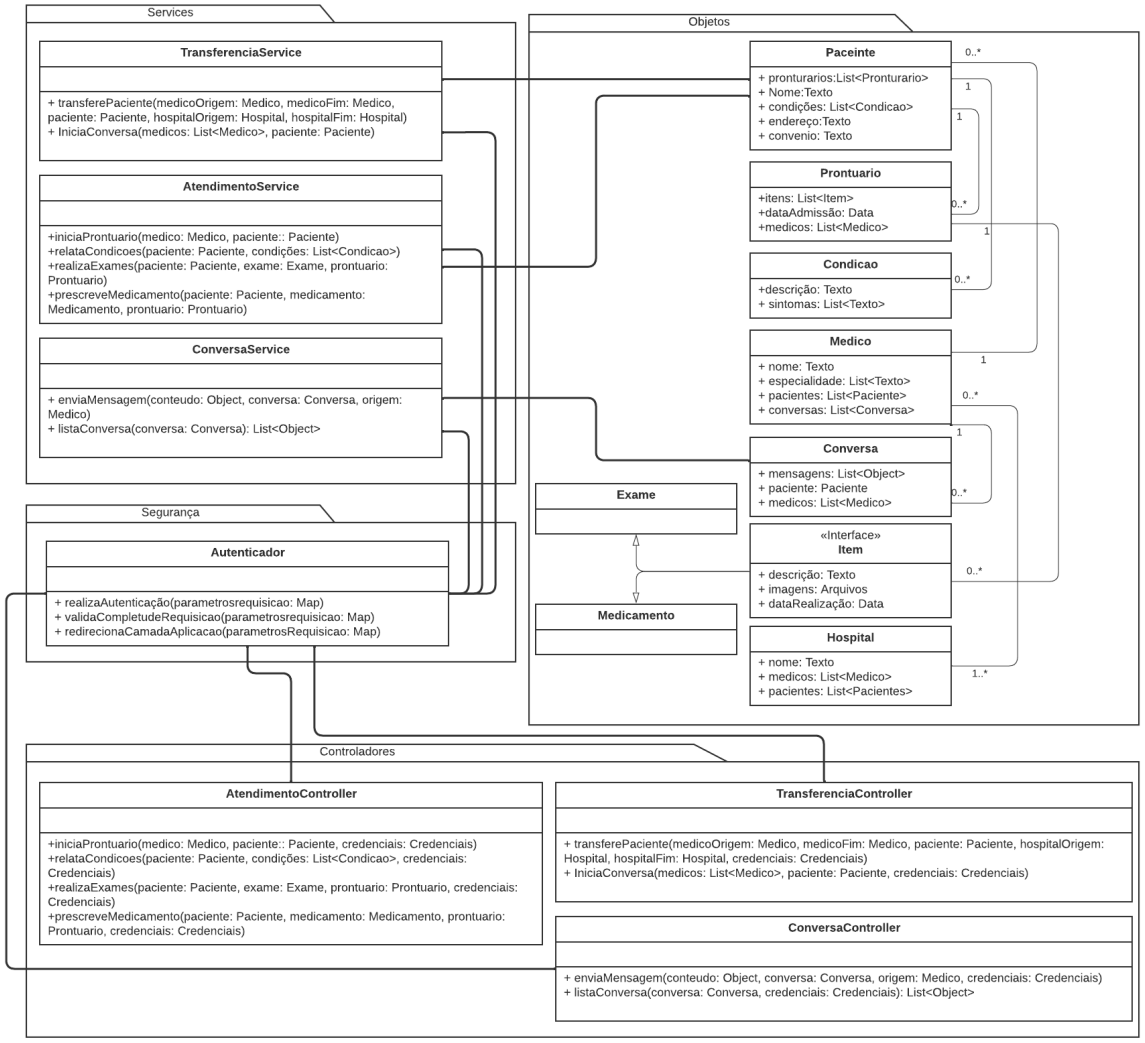
Conforme definido o projeto será dividido em camadas, abaixo está o detalhamento das classes e tarefas realizadas por elas nas camadas de persistência de dados, aplicação, segurança e controle.

A camada de persistência de dados é composta pelos objetos sendo eles os Pacientes, Prontuários, Condições, Médicos, Conversas, Itens e Hospitais. Essas classes representam as entidades que se relacionarão no projeto e suas informações relevantes.

A camada de aplicação ou service é a camada onde os processos serão realizados. É composta das classes TransferenciaService, responsável por realizar o processo de transferência de paciente entre médicos e hospitais e iniciar conversas entre os médicos responsáveis pelo paciente. A classe AtendimentoService é responsável por realizar os processos relativos aos atendimentos do paciente, cadastrando procedimentos, medicamentos e condições de saúde do paciente e outras atividades relacionadas ao atendimento médico do paciente. A classe ConversaService é responsável por administrar as ações realizadas pelos médicos durante a comunicação de informações sobre o paciente.

A camada de segurança será composta de um Autenticador, que receberá as requisições da camada de controle, irá validar as credenciais e as informações passadas como parâmetros e irá redirecionar para os serviços corretos na camada de aplicação conforme necessário.

A camada de controle é composta pelo TransferenciaController que irá receber as requisições do cliente a respeito de ações na tela de transferência de pacientes. Do AtendimentoController que irá receber as requisições relativas às ações da tela de atendimento. E o ConversaController receberá e redirecionará as ações relativas a tela de conversa entre os médicos



Recurso: [Lucidchart]

7. Visão de segurança

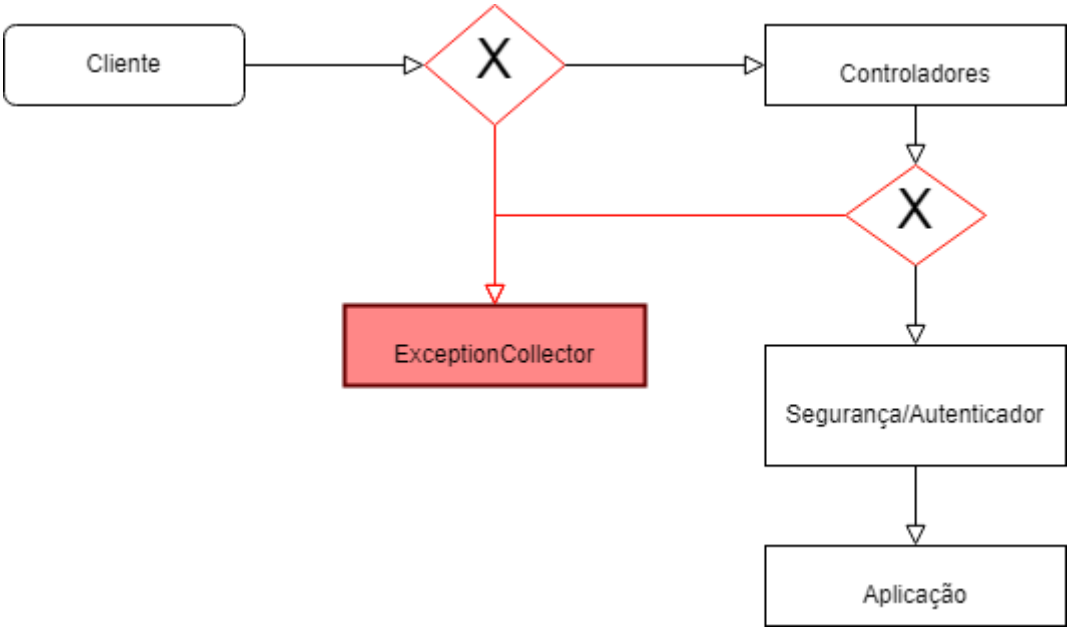
Devido a manipulação de informações pessoais dos usuários foi definido que a segurança é uma característica essencial para o projeto, dessa forma foi definido utilizar uma visão de segurança para estabelecer alguns padrões que serão utilizados

A utilização da arquitetura em camadas já estabelece alguns padrões de segurança como as interfaces implementadas nas mesmas, delimitando padrões de comunicação, assim como a existência da camada de segurança e os tratamentos de exceções em cada camada.

7.1.1. Detalhamento da segurança

As classes responsáveis pela captura de erros são os Controladores e o Autenticador. Os controladores irão validar a consistência das informações passadas na requisição e se foram recebidas todas informações necessárias, caso algum tipo esteja incorreto ou informação esteja faltando. O Autenticador será responsável por validar as credenciais e as informações da comunicação para poder determinar se a requisição é de um remetente válido ou não.

No caso de uma falha em qualquer uma das etapas citadas acima, uma exceção é gerada, ela é coletada pela classe ExceptionCollector e tratada em seguida.



8. Ponto de vista do Implantador

8.1. Visão geral

O ponto de vista do implantador é direcionado para equipe de implantação e define como será o ambiente de execução do sistema, definindo as ferramentas e processos necessários para seu funcionamento. Definindo como será feita a relação entre os diversos componentes do sistema.

8.2. Visão física

A visão física descreve o sistema do ponto de vista de um engenheiro de sistemas. Ele se preocupa com a topologia dos componentes de software na camada física, bem como com as conexões físicas entre esses componentes.

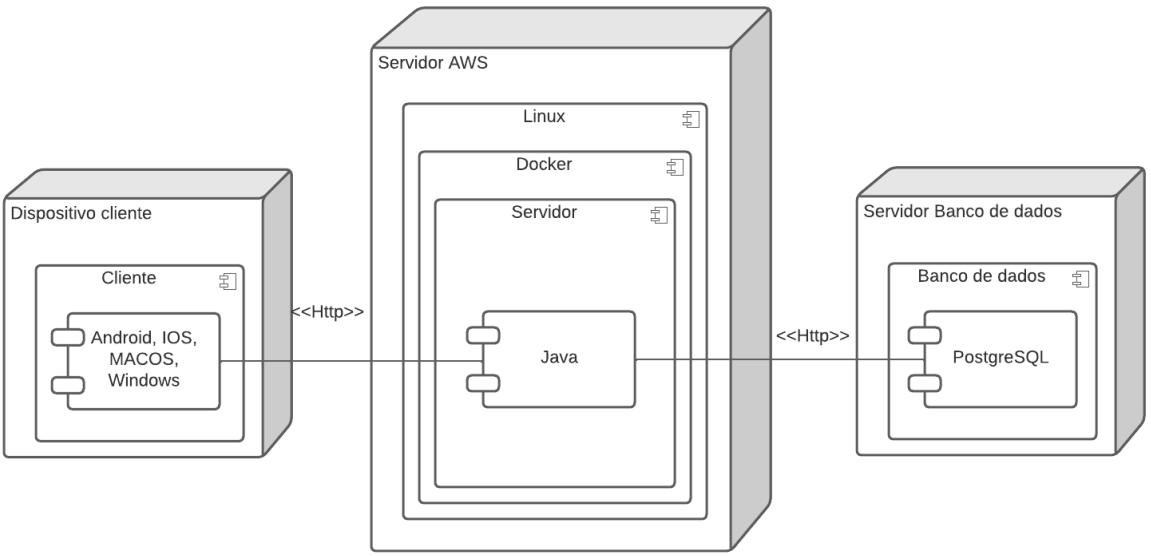
O projeto será dividido em 3 componentes: o componente cliente, o componente servidor e o componente de gerenciamento de dados. Cada componente terá sua representação física e as ferramentas necessárias para sua execução

8.2.1. Detalhamento dos nós-físicos

O componente do cliente será representado pelo computador ou aparelho móvel do usuário. Para seu funcionamento será necessário a instalação do aplicativo no dispositivo, a aplicação será fornecida para os sistemas operacionais: Windows 10, 11, iOS, Mac OSx e Android.

O componente de servidor será desenvolvido em várias máquinas virtuais AWS, com sistema operacional Amazon Linux 2. Será necessário que ela tenha a versão 19 do java. O componente servidor será executado através do Docker versão 20.10, para permitir um ambiente de execução controlável.

O componente de gerenciamento de dados é responsável por armazenar e fornecer acesso aos dados necessários para o funcionamento correto dos outros componentes do software. Será composto de um banco de dados desenvolvido na tecnologia PostgreSQL versão 15.1 e hospedado no serviço de banco de dados gerenciado Supabase.



Recurso: Lucidchart

8.2.2. Detalhamento do CI/CD

Sempre que um novo codigo for aceito via PR para a branch main, algumas actions do GitHub Actions começarão a rodar. Essas actions devem:

- Rodar os testes
- Buildar o container

- Subir o container para o Docker Hub
- Rodar o container na máquina virtual
- Publicar no servidor de produção

9. Framework WebXR: Unity

Para a implementação da realidade aumentada permitindo a visualização 3D dos pacientes que estão sob cuidado no hospital, e do avatar do paciente durante a comunicação entre os médicos será utilizado o Unity e exportado para WEB XR.

Esse ambiente de realidade aumentada permitirá uma visualização mais dinâmica de exames de imagem e de uma avatar do paciente que terá uma representação da condição do paciente de forma visual de forma a facilitar a ambientação do profissional com o paciente.

Screenshot: