***Transferência do Cuidado de Pacientes 3.0***

**Documento de Arquitetura de Software**

**Alunos**:

Amanda Lobo Gomes

Luca Baccheschi Benetti

Lucas Oliveira de Souza

Vitor Henrique Ferreira de Brito

1. **Introdução**
   1. **Finalidade**

Este documento possui como objetivo definir os aspectos da Arquitetura do software Transferência do Cuidado de Pacientes 3.0, sendo direcionado a possíveis stakeholders, gerentes de projeto, clientes, desenvolvedores, implantadores e demais equipes técnicas responsáveis.

* 1. **Escopo**

Este documento se baseia no documento "Especificação Trabalho Final - PAS 2022/2” para definir os atributos de qualidade a serem priorizados, bem como padrões de projeto de estilos arquiteturais que os favoreçam, assim como as representações das visões arquiteturais e seus subprodutos.

* 1. **Definições, Acrônimos e Abreviações**

**AAS:** Artefato de Arquitetura de Software.

**RAS:** Requisito de Arquitetura de Software.

**Software:** Conjunto de documentações, guias, metodologias, processos, códigos e ferramentas para a solução de um problema.

**Sistema:** Conjunto de pessoas, softwares, hardwares e outros sistemas para a solução de um problema.

**Stakeholder:** Indivíduo, grupo ou organização que possua interesse no Sistema.

**Visão Arquitetural:** Produto resultante da interpretação de um Stakeholder do sistema.

**Arquitetura de Software:** Forma como os componentes são agrupados com o objetivo de construir um software ou sistema.

**Atributos de qualidade**: São atributos que impactam diretamente na concepção de um software, são definidos conforme a ISO-IEEE 9126.

**Sistema Operacional**: Software responsável por gerenciar e abstrair a interação entre o usuário e o hardware ou aplicações externas e o hardware.

**UML**: Sigla para Linguagem de Modelagem Unificada.

* 1. **Referências**

| **Id** | **Nome do artefato** |
| --- | --- |
| AAS\_1 | Documento Especificação Trabalho Final - PAS 2022/2 |
| AAS\_2 | ISO-IEEE 9126 |
| AAS\_3 | Slides Ministrados em Sala |
| AAS\_4 | 4+1 View |

* 1. **Visão Geral**

Os próximos tópicos irão descrever os requisitos e restrições utilizados para definição da arquitetura a ser implementada e também quais atributos de qualidade serão priorizados e o motivo de cada escolha.

Quais os padrões de arquitetura serão utilizados, bem como o porquê da escolha de tais padrões arquiteturais. Além do detalhamento das visões arquiteturais.

1. **Contexto da Arquitetura** 
   1. **Funcionalidades e Restrições Arquiteturais**

| **Id.** | **Tipo** | **Requisito no documento** |
| --- | --- | --- |
| RAS 1 | Requisitos Não-Funcionais | Disponibilidade de dados em tempo real |
| RAS 2 | Requisitos Não-Funcionais | Alta tolerância a falhas |
| RAS 3 | Requisitos Não-Funcionais | Balanceamento de cargas |
| RAS 4 | Requisitos Não-Funcionais | Permitir grande volumes de processamento de dados |

Os requisitos de arquitetura de software direcionam para qual estilo de arquitetura de software será adotada. Dessa forma, esses RAS direcionam para escolha da arquitetura de microsserviços.

Os RAS levaram a escolha do microsserviços, uma vez que essa arquitetura prioriza a grande disponibilidade, eficiência e confiabilidade. Além disso, por ser dividido em serviços independentes, é possível tornar a manutenção deste software de forma rápida.

* 1. **Atributos de Qualidades Prioritários**

Conforme definido na seção anterior, as informações do paciente devem ser disponibilizadas em tempo real, quando o paciente for identificado pelos AV/RV. Sendo assim, o principal atributo de qualidade a ser priorizado é o de **eficiência,** uma vez que esse estilo (**microsserviços**), garante que o tempo de resposta será o mais rápido possíve**l**, mesmo quando processando grandes volumes de dados.

Além disso, havendo três servidores para cada microsservico, para a atender as requisições, em caso de falha de um desses servidores, a requisição deve ser direcionada a outra instância. Com isso, outro atributo de qualidade pode ser priorizado, que é a **confiabilidade,** pois o sistema se torna **tolerante às falhas**.

Além disso, a arquitetura a ser adotada favorece a manutenção, uma vez que cada serviço funciona de forma independente e consequentemente, facilita para que partes do sistema sejam modificadas sem que afetem o desempenho total da aplicação. Sendo assim, outro atributo de qualidade priorizado será **manutenibilidade**.

1. **Representação da Arquitetura**

Com base nos tópicos anteriores, a arquitetura a ser desenvolvida será a de microsserviços, priorizando os atributos de qualidade: **eficiência, confiabilidade** e **manutenibilidade**.

Para representar as decisões arquiteturais, serão utilizados as seguintes visões:

| **Visão** | **Diagrama(s)** |
| --- | --- |
| Casos de Uso | Casos de Uso |
| Desenvolvimento | Componentes |
| Lógica | Classes |
| Física | Implantação |
| Processo | Atividades |

1. **Visão de casos de uso**

Os casos de uso representam uma unidade discreta referente a interação entre um usuário e o sistema, ele define uma sequência de ações executadas pelo sistema que geram um resultado de valor para um ator em particular.

Para se garantir que todos os requisitos sejam atendidos no desenvolvimento de um projeto, em relação às questões arquiteturais é necessário ser disponibilizado uma análise de requisitos, está visão é denominada de “Visão de casos de uso”, responsável por apresentar os determinados casos de uso e cenários que em conjunto representam os comportamentos esperados para o sistema em si.

O diagrama de casos de uso está representado abaixo:

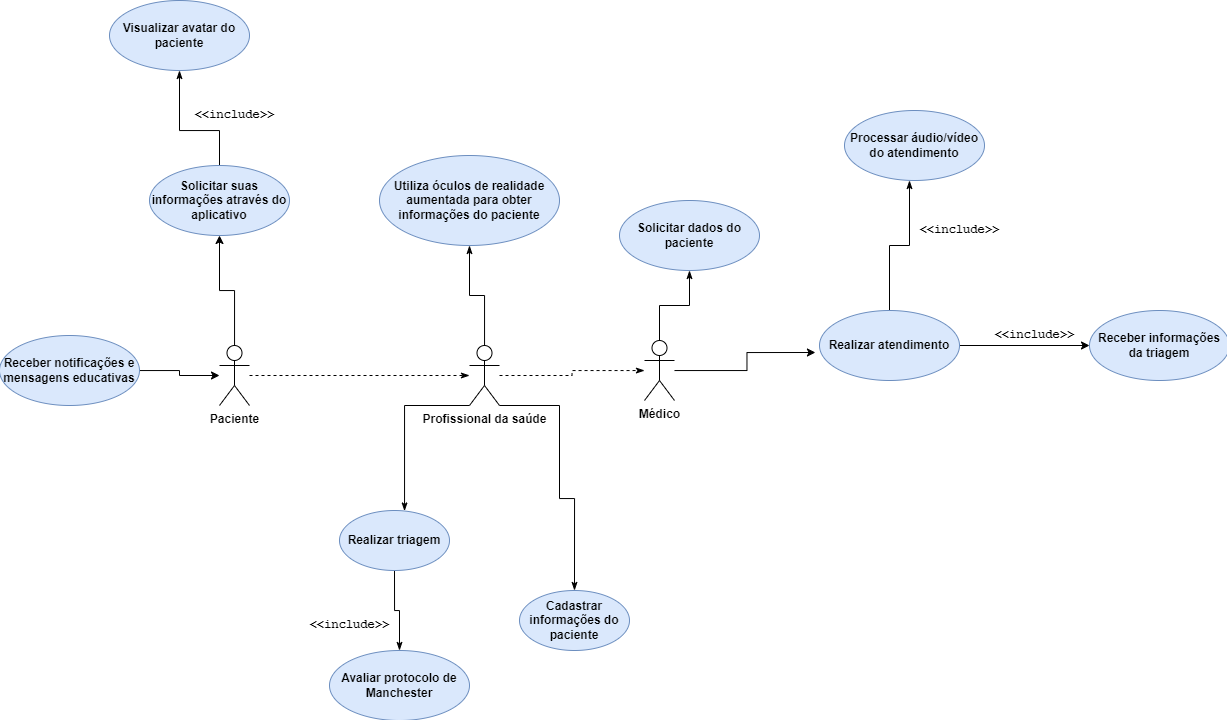
****

Diagrama de Casos de Uso

1. **Visão de desenvolvimento**

A visão de desenvolvimento é responsável por definir as principais partes que compõem o sistema, como por exemplo,os componentes, assim como suas responsabilidades.

Para entender a visão de desenvolvimento do projeto, foram definidas as principais partes, juntamente com suas funcionalidades, como pode ser observado no diagrama a seguir:

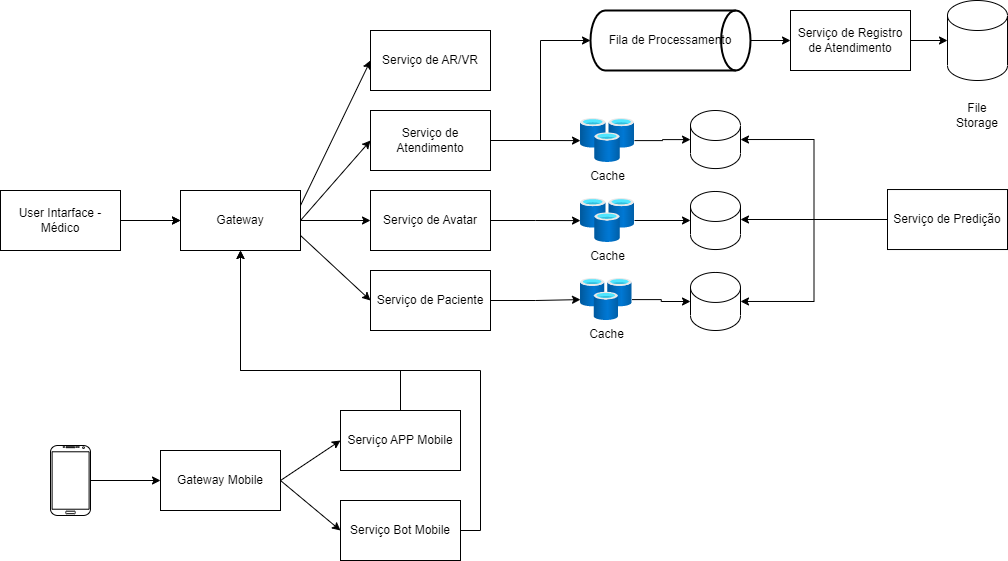


Diagrama de componentes

Por se tratar de uma arquitetura de microsserviços, o projeto foi dividido em diversos serviços com suas respectivas funções, além de outros componentes, que serão explicados a seguir:

* **Serviço APP Mobile e Serviço Bot Mobile** - backend do paciente que notifica o paciente e envia mensagens educativas.
* **Serviço de Paciente** - contém dados do paciente, sendo alimentado por médicos e sensores.
* **Serviço de Avatar** - contém dados e especificações de avatares.
* **Serviço de Atendimento** - contém os dados de atendimento de um paciente, além de receber requisição de triagem e avaliar o protocolo Manchester.
* **Serviço de AR/VR** - adapter para os óculos e demais dispositivos e drivers necessários.
* **Fila de Processamento** - processa o registro de áudio e vídeo de atendimentos (por serem dados de grande volume).
* **Serviço de Predição** - consome os dados existentes nos vários banco de dados para realizar a predição de atendimentos e situações de saúde dos pacientes.
* **Cache** - disponibiliza de forma rápida os dados do paciente quando acessa a unidade de saúde.

Além disso, os serviços podem ser divididos e separados em responsabilidades. Sendo assim, cada serviço citado anteriormente, seguirá a estrutura básica a seguir:

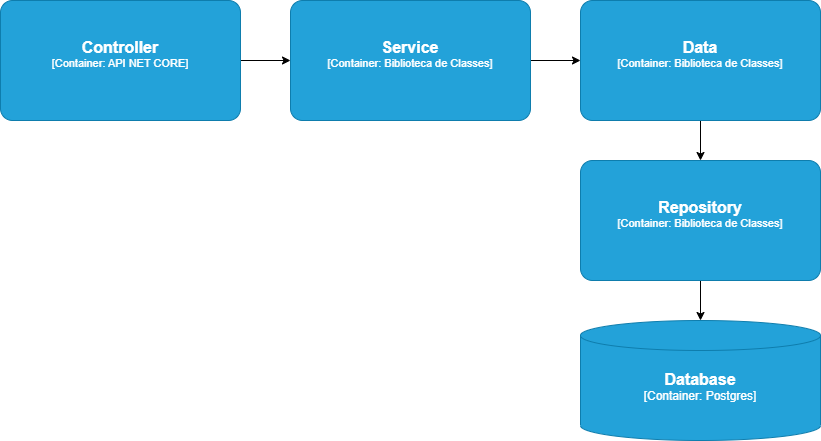
****

Diagrama de componentes expandido

Com isso, os serviços foram divididos em camadas que tem as seguintes responsabilidades:

* **Controller** - responsável por intermediar as requisições de outros serviços ou componentes, processando as informações e repassando para a camada seguinte.
* **Service** - responsável pela lógica de negócio e manipulação dos dados.
* **Data** - responsável pela estruturação do banco de dados em forma de modelo e classes de negócio.
* **Repository** - responsável pela manipulação referente às entidades dentro do banco de dados.

1. **Visão lógica**

A visão lógica é responsável por descrever como o sistema está estruturado em termos de unidades de implementação, os elementos são pacotes, classes e interfaces, é possível estabelecer os relacionamentos entre esses elementos das quais é apresentado suas dependências.

Esta visão tem por objetivo definir como as estruturas dos componentes estão representadas, auxiliando no desenvolvimento da proposta de arquitetura.

Para melhor entendimento, o diagrama de classes foi separado de acordo com cada serviço descrito, sendo eles:

* **Serviço de paciente**

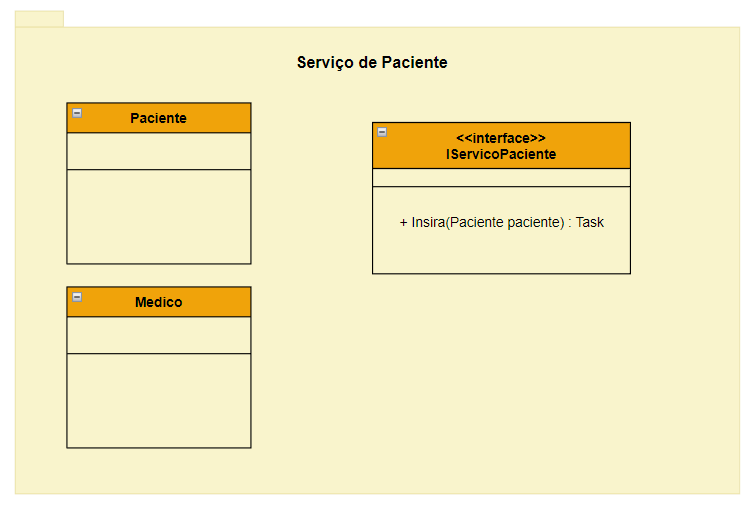
****

Diagrama representando o serviço de paciente

Este serviço será responsável por disponibilizar e realizar a manutenção referente aos dados do paciente, dos quais serão preenchidos pelos médicos e sensores.

* **Serviço de avatar**

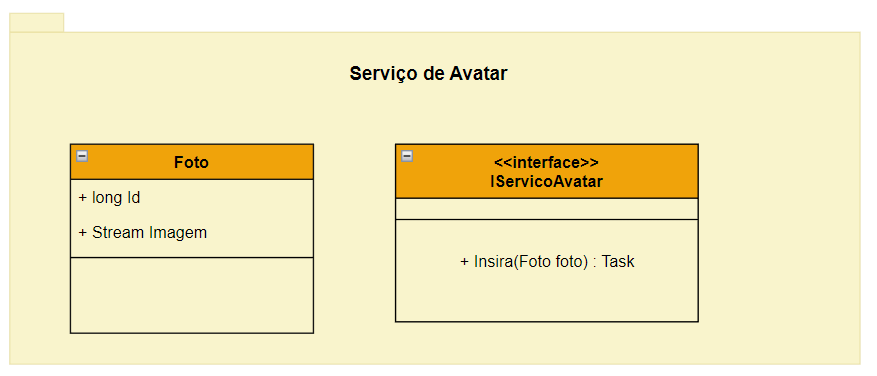
****

Diagrama representando o serviço de avatares

Este serviço será responsável por manipular e especificar os avatares utilizados no cadastro dos pacientes, por exemplo.

* **Serviço de notificação**

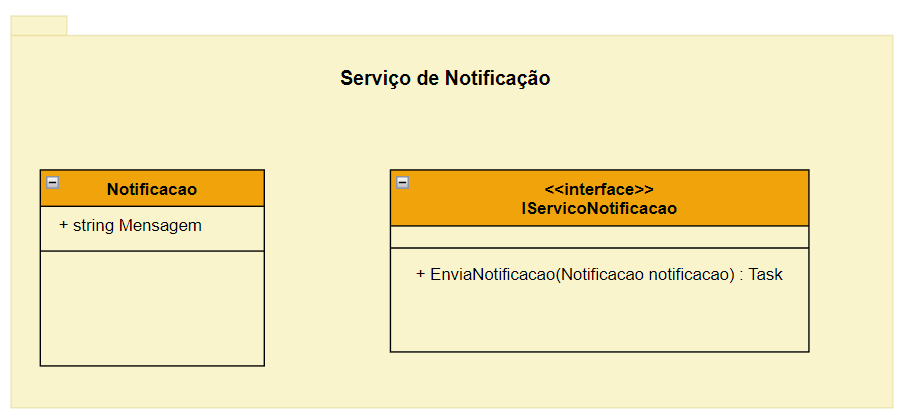


Diagrama representando o serviço de notificação

Este serviço será responsável por manipular as notificações que serão enviadas para os respectivos pacientes, das quais essas notificações também poderão ser de caráter educativo.

* **Serviço de AR/VR**

****

Diagrama representando o serviço de notificação

Este serviço será responsável por ser o interceptador/adaptador entre os óculos, dispositivos e drivers necessários para a execução do requisito.

* **Serviço de atendimento**

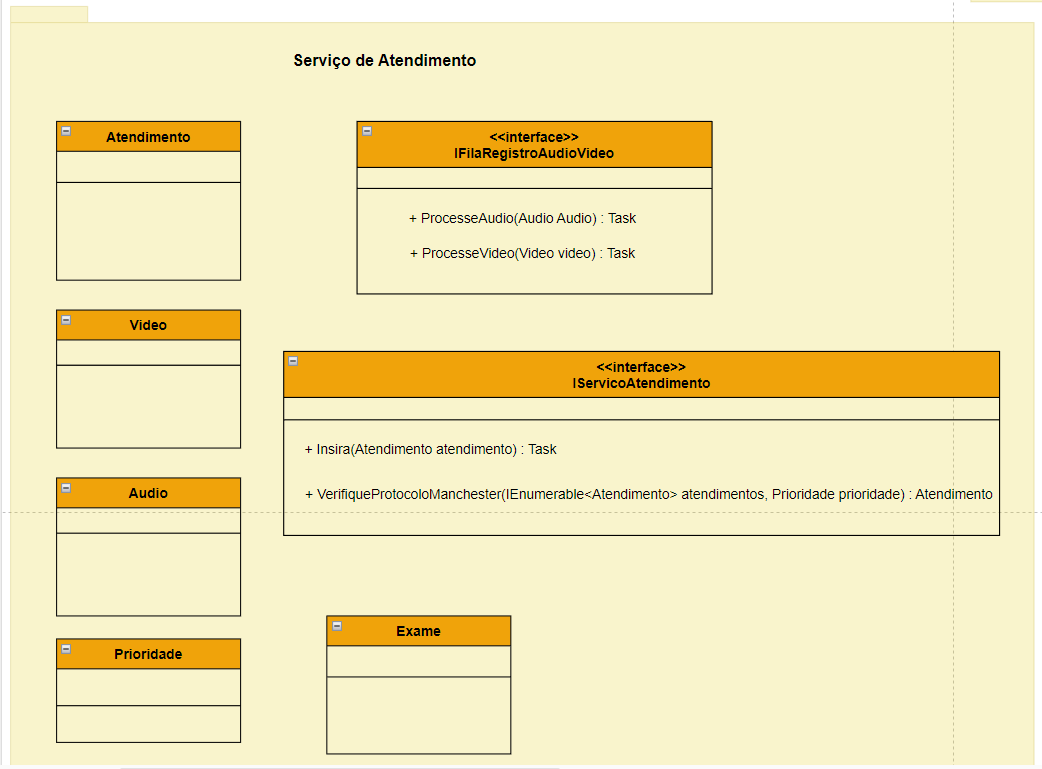
****

Diagrama representando o serviço de atendimento

Este serviço será responsável por manipular e gerenciar os respectivos atendimentos que estão sendo realizados, dos quais caso tenha sido feito algum registro por áudio ou vídeo estes podem ser auditados para possíveis análises futuras.

1. **Visão física**

A visão física é a responsável por definir as ferramentas e os ambientes necessários para o funcionamento do software. Portanto, o **diagrama de implantação** para esse sistema, será o seguinte:

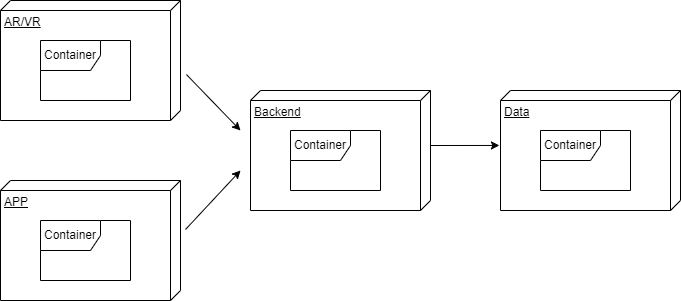
****

Diagrama de Implantação

De forma simplificada, há uma divisão de camadas na visão física:

* **AR/VR:** camada responsável pela interação dos óculos de realidade aumentada (AR) e virtual (VR).
* **APP:** camada responsável pela interação com os usuários pelo aplicativo.
* **Backend**: camada responsável por receber a requisição do APP e do AR/VR, processar, validar e comunicar com a camada de dados.
* **Data:** camada de dados. Para os serviços de áudio e vídeo, é preferível que utilize algum tipo de serviço de storage, como AWS S3, por se tratar de arquivo. Para os demais dados, utilizar um banco de dados.

1. **Visão de processo**

A visão de processos, é responsável por estruturar o sistema em linhas de execução de processos, assim como as interações e comportamentos.

Sendo assim, a sequência de execução das atividades é crucial para atender os atributos de qualidade do sistema. Essas atividades representam as ações que os usuários podem desempenhar no sistema, e as responsabilidades que os componentes do sistema têm para executá-las.

Foi definido os diagramas contendo as atividades, que foram divididos por tipo de usuário, para facilitar a visualização, como pode se observar a seguir:

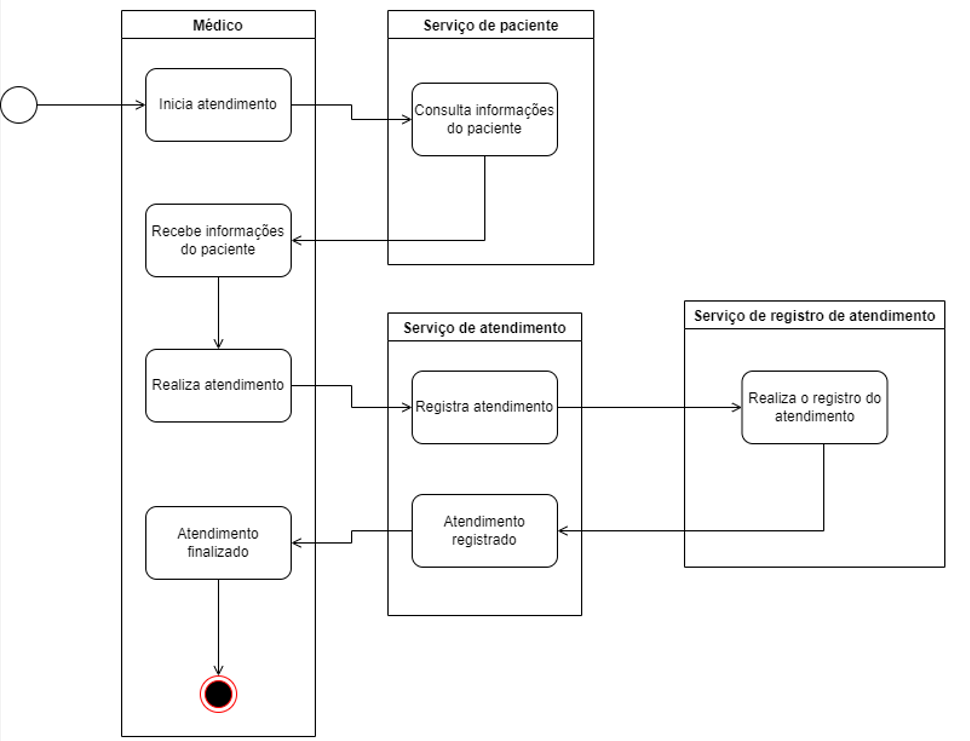


Diagrama de processo - Médico

O médico é responsável pela realização do atendimento, no qual pode-se consultar as informações do paciente em questão. Recebendo essas informações do serviço de paciente o atendimento é iniciado.

Logo após o atendimento ser finalizado, ele é registrado através do serviço de atendimento , que armazena os vídeos e fotos referente a esse atendimento.

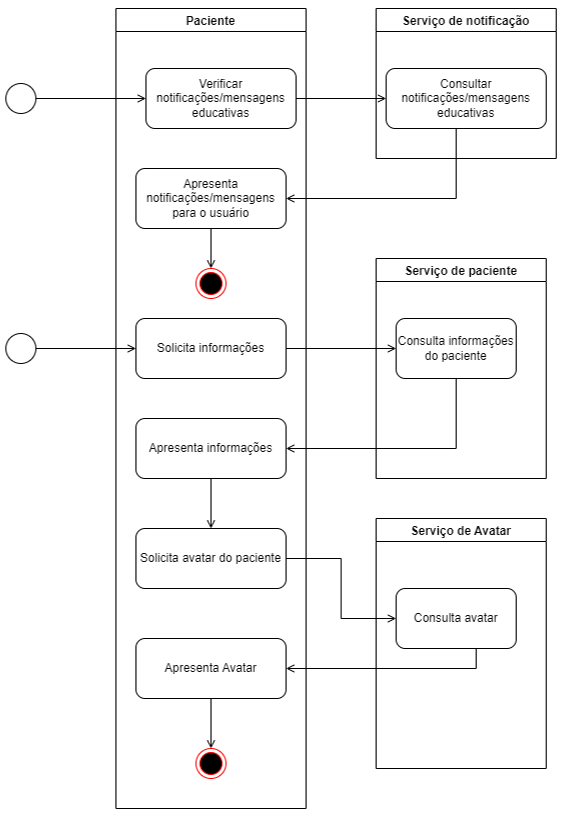


Diagrama de processo - Paciente

O paciente consegue acessar suas informações cadastradas, além de consultar seu avatar para visualizar. Somando-se a isso, através do serviço de notificação, o paciente é notificado dos atendimentos, além de receber mensagens educacionais dos estabelecimentos hospitalares.

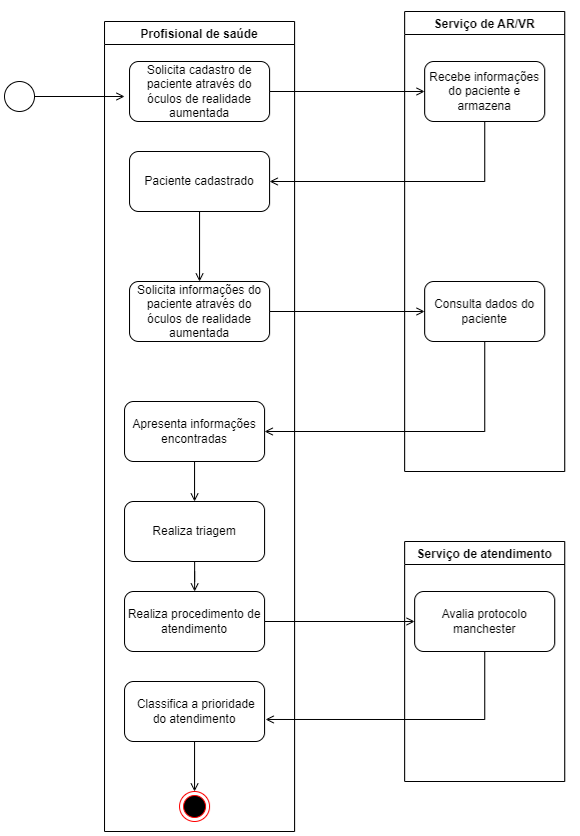


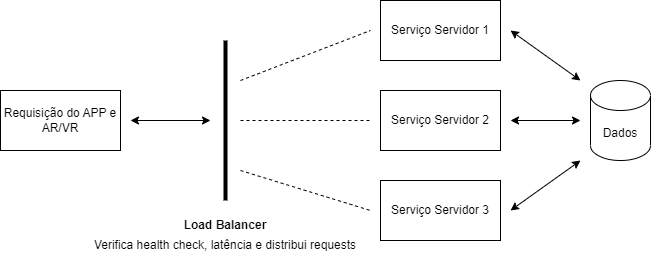
Diagrama de processo - Profissional de Saúde

O profissional de saúde é o responsável no primeiro momento pelo atendimento ao paciente, através do óculos de realidade aumentada é possível visualizar as informações de um paciente desde que o mesmo já possua cadastro, caso não tenha, o mesmo pode ter suas informações cadastradas no sistema..

Ao realizar o atendimento em primeiro momento, o profissional faz a avaliação através do protocolo de Manchester, verificando a prioridade e estado do paciente naquele momento.

1. **Desenho arquitetural**

Com base no que foi discutido nos tópicos anteriores e levando em consideração os atributos de qualidade priorizados, ficou definido que cada serviço da aplicação, terá a seguinte estrutura básica:

****

Desenho arquitetural

Todos os serviços da aplicação, serão rodados em três servidores e seguirão o seguinte fluxo:

1. O usuário realiza ação no APP ou o AR/VR é acionado, gerando uma requisição.
2. Requisição chega ao load balancer, onde são empregadas algumas técnicas:

**2.1.** Há um health check para saber o estado dos 3 servidores que rodam a mesma aplicação.

**2.2.** Existe uma escolha do servidor a ser utilizado de acordo com a latência (menor, mais prioridade).

1. Servidor recebe a requisição do load balancer, busca e retorna o dado solicitado.
2. **Decisões arquiteturais**

De forma geral, para atender os requisitos e garantir os atributos de qualidade "**Confiabilidade**" e “**Eficiência**”, existe a necessidade de replicações das instâncias da aplicação, de forma a garantir que em mal funcionamento de uma, outra que esteja disponível e processe a requisição.

Sendo assim, levando em consideração que a disponibilidade é um dos grandes pilares da computação em nuvem, seria interessante aproveitar a computação em nuvem para desenvolver de forma completa nossa aplicação, com os exemplos a seguir, utilizando Amazon Web Services:

* **Backend**: aqui possuímos várias possibilidades, poderíamos provisionar os 3 servidores, de maneira que fiquem sempre ativos em máquinas AWS EC2, possibilitando uma escala automática de acordo com requests, de forma que caso haja uma carga maior, novas instâncias são criadas automaticamente, e desligadas em caso de menor carga. Outra possibilidade, é utilizarmos de serviços serverless (como AWS Lambda), sem que haja a necessidade de provisionar os recursos, a cada requisição, o processamento é realizado e é criada uma instância, que ao fim do processamento é deletada.
* **Cache:** poderia utilizar o Elastic Cache - Redis
* **Dados**: poderia ser hospedado os arquivos no serviço AWS S3.

Para que a aplicação da arquitetura não fique dependente de uma solução de computação em nuvem específica, temos de nos ater aos conceitos: cache, servidor e banco de dados são possíveis de serem provisionados em vários serviços de computação em nuvem existentes.

Assim, os padrões identificáveis são: camadas (separação entre front, back, camada de acesso a dados), SOA (o serviço de obter vídeo tem apenas uma responsabilidade), e o padrão de proxy reverso (sendo utilizado na forma de load balancer na nossa aplicação).