Vacinei

Vacinei
Documento de Arquitetura de Software

2.1

Histórico da Revisão

Data	Versão	Descrição	Autor
24/09/2020	1.0	Criação do Documento	Dominic Rocha de Paulo
20/10/2020	1.1	Revisão do Documento	Felipe Gonçalves Ferreira
29/09/2020	2.0	Revisão do Documento	Dominic Rocha de Paulo
04/10/2020	2.1	Revisão do Documento	Felipe Gonçalves Ferreira

Índice

Documento de Arquitetura de Software

1. Introdução

1.1 Este documento tem como objetivo fornecer uma visão geral da arquitetura que será usada no desenvolvimento do projeto e permitir um maior entendimento do módulo IndicaAi, para o aplicativo Integra, e de como ele irá se comportar e se comunicar com as outras aplicações que compõem o projeto. Ele deve mostrar de forma clara e objetiva as decisões arquiteturais que foram tomadas em relação ao projeto.

1.2 Objetivo

Este documento tem por objetivo documentar a arquitetura de software, apresentando visõesarquiteturais do sistema de gerenciamento de vacina, levando em consideração os atributos de qualidade escolhidos para o projeto.

1.3 Escopo

O escopo deste documento é apresentar várias visões diferentes para auxiliar no desenvolvimento do projeto orientando todos os envolvidos técnicos sobre o design de arquitetura escolhida para o sistema.

1.4 Definições, Acrônimos e Abreviações

MVC (Model View Controller) : Arquitetura de software utilizada em sistemas que desejam separar a modelagem de dados, interface e processamento de requisições em camadas independentes.Visão Geral

2. Representação Arquitetural

A arquitetura definida para o sistema é o padrão MVC na plataforma Web. A arquitetura foi definida com base no conhecimento dos envolvidos técnicos do projeto e segundo os atributos de qualidade escolhidos, como testabilidade, modularidade, modificabilidade e usabilidade.

Visão	Público	Área	Modelo da MDS
Lógica	Analistas	Realização dos Casos de Uso	
Processo	Integradores	Performance, Escalabilidade, Concorrência	
Implementação	Programadores	Componentes de Software	
Implantação	Gerência de Configuração	Nodos físicos	
Caso de Uso	Todos	Requisitos	

		funcionais	
Dados	Especialistas em dados Administradores de dados	Persistência de dados	

Visão de Caso de Uso

Apresenta funcionalidades importantes

Visão de Processos

Visão Lógica

Descreve as classes do projeto e seus relacionamentos apresentando a organização

Visão Componentes/Pacotes

3. Restrições e Metas Arquiteturais

- Estrutura MVC;
- Linguagem de programação PHP e JavaScrip;
- Bancos de dados MySql;

Requisito	Solução
Linguagem	PHP e JavaScript
Plataforma	Web
Segurança	O software garantirá a segurança dos dados informados pelo usuário
Persistência	MySql

4. Visão de Casos de Uso

Nesta sessão está presente os casos de usos significativos para a arquitetura do sistema.

Esta seção lista as especificações centrais e significantes para a arquitetura do sistema, os itens em negritos desta seção são os casos de uso central.

Lista de casos de uso do sistema:

- Criar Conta
- Recuperar Conta
- Realizar Login
- Deletar Conta

- **Exportar Registro das Vacinas**
- **Agendar Aplicação Vacina**
- Editar Registro de Aplicação de Vacina
- **Registrar Aplicação de Vacina**
- **Cadastrar Tipo de Vacina**
- **Visualizar Cartão de Vacina Virtual**
- Visualizar Vacinas Aplicadas
- Cadastrar local de aplicação de vacina
- Cadastrar tipo de vacina

Nome do Caso de Uso	**AgendarAplicaçãoVacina**
ID Caso de Uso	**UC06**
Descrição	1. Ousuáriodevesercapazdeagendaraaplicaçãodeumavacina.
Atores	1. Usuário
Pré-Condições	1. Oator deverá estar logado. 2. Deve haver ao menos uma vacina cadastrada.
Pós-Condições	1. Vacina agendada.
FluxoPrincipal	1. Oatorseleciona aopçãodeagendamento. 2. O sistema direciona o ator para a página solicitada. 3. O ator seleciona a data e hora. 4. O sistema exibe tela de cadastro do paciente. 5. O ator informa os dados do paciente. 6. O sistema exibe o localizador de vacinas. 7. O ator seleciona o local de administração da vacina. 8. O ator seleciona os dados da vacina. 9. O sistema emite um alerta de agendamento realizado e muda a cor da data marcada.
FluxoAlternativo	**A1 – Horário não disponível**

	1. No passo 3 do fluxo principal o ator tenta marcar em um horário do dia que já está agendado. 2. O sistema alerta o ator que já existe um paciente agendado para aquele horário e aborta a ação. 3. Sistema retorna para o passo 3 do fluxo principal. **A2 – Vacina não cadastrada** 1. No passo 3 do fluxo principal o ator não consegue localizar a vacina. 2. O sistema alerta não encontrou nenhum resultado com o filtro selecionado. 3. O sistema solicita se o ator deseja realizar o cadastro da vacina e direciona para o caso de uso UC09.
Exceções	

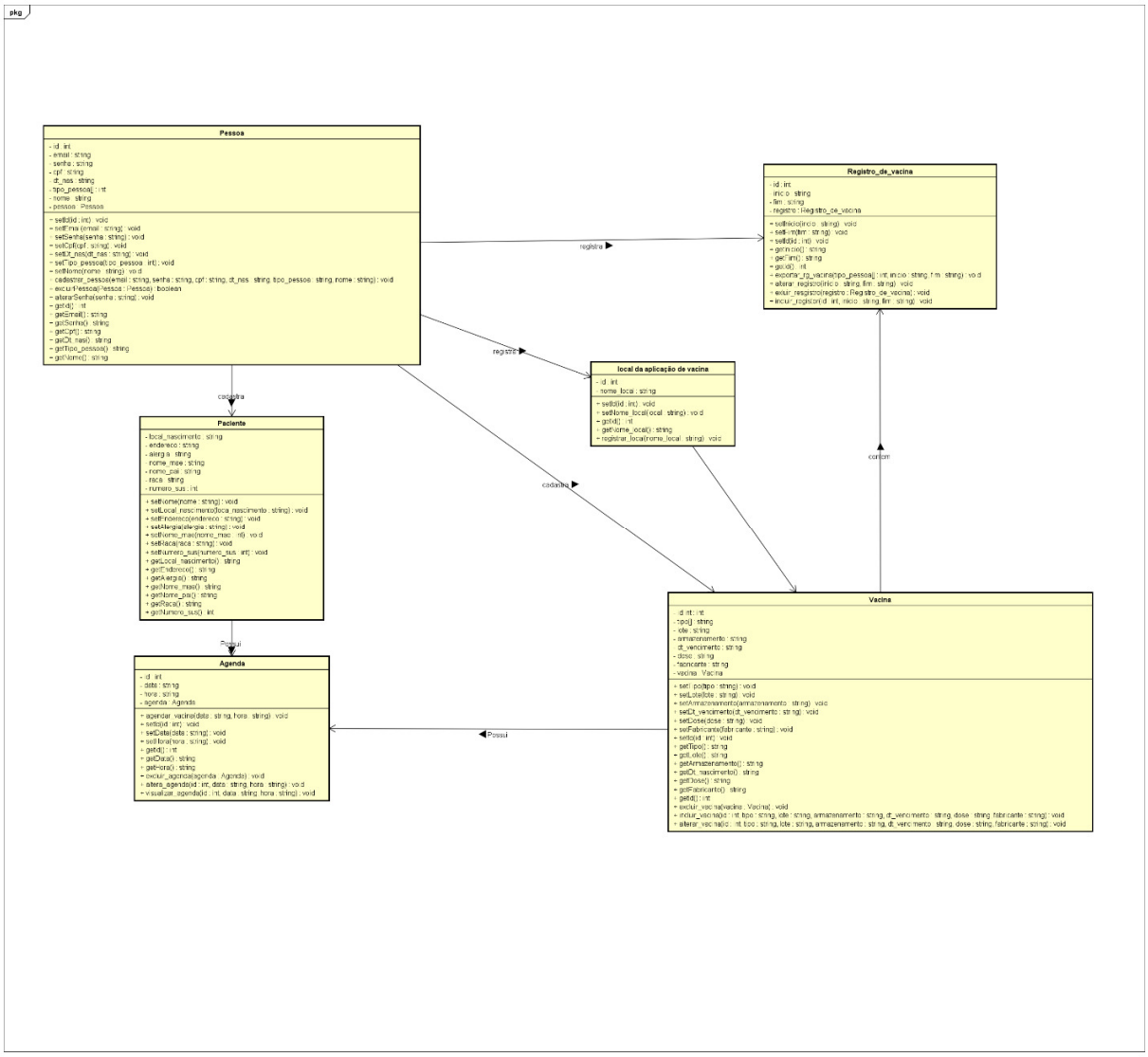
5. Visão Lógica

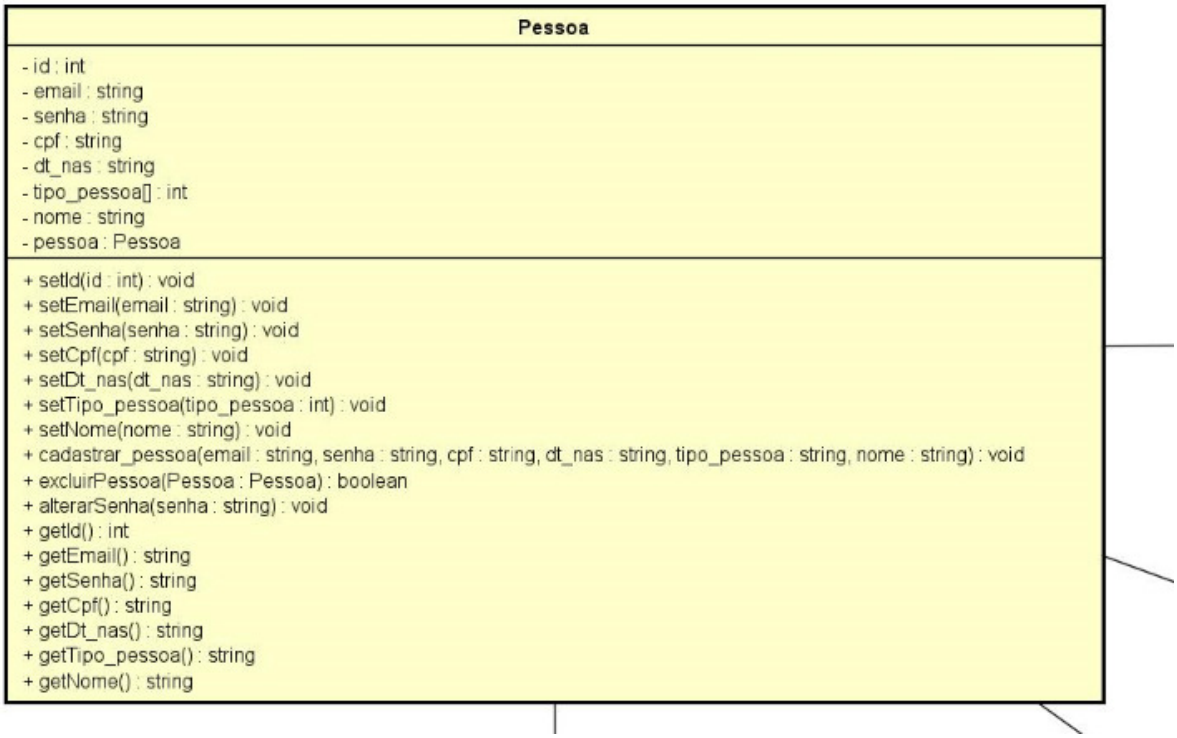
5.1 Visão Geral

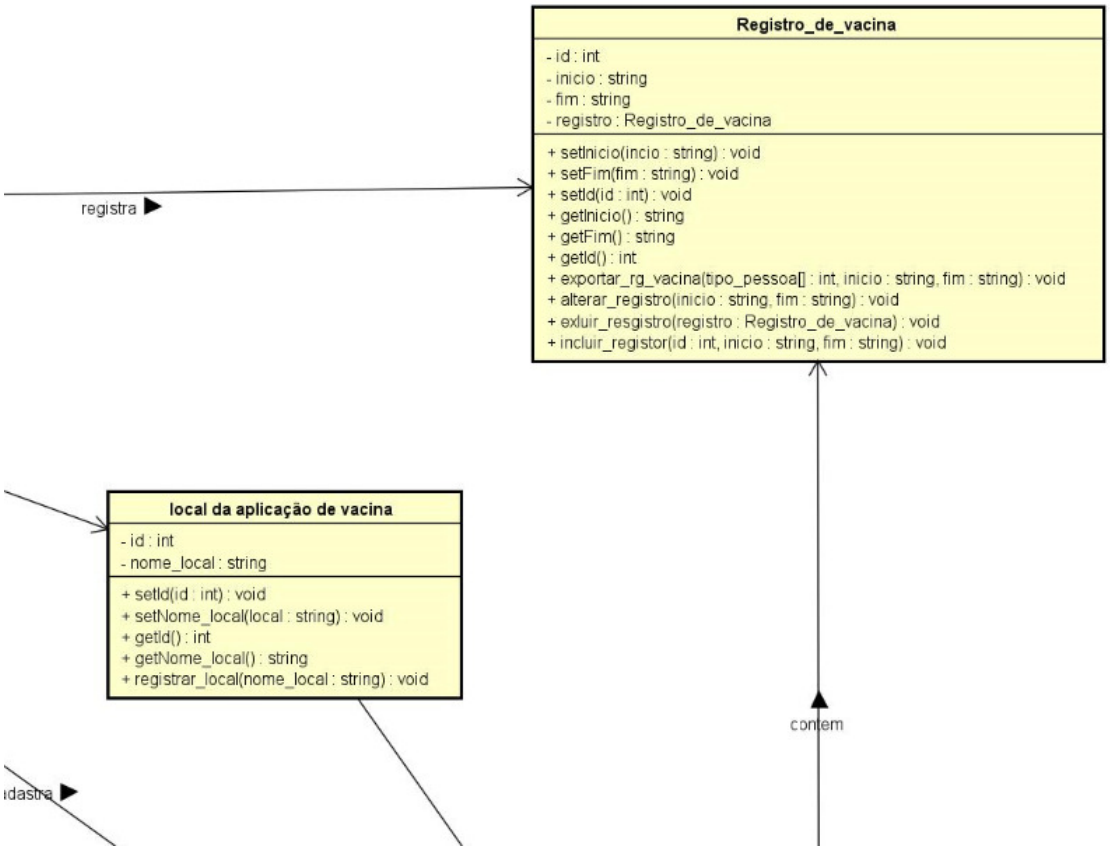
A visão lógica define a estrutura da arquitetura:

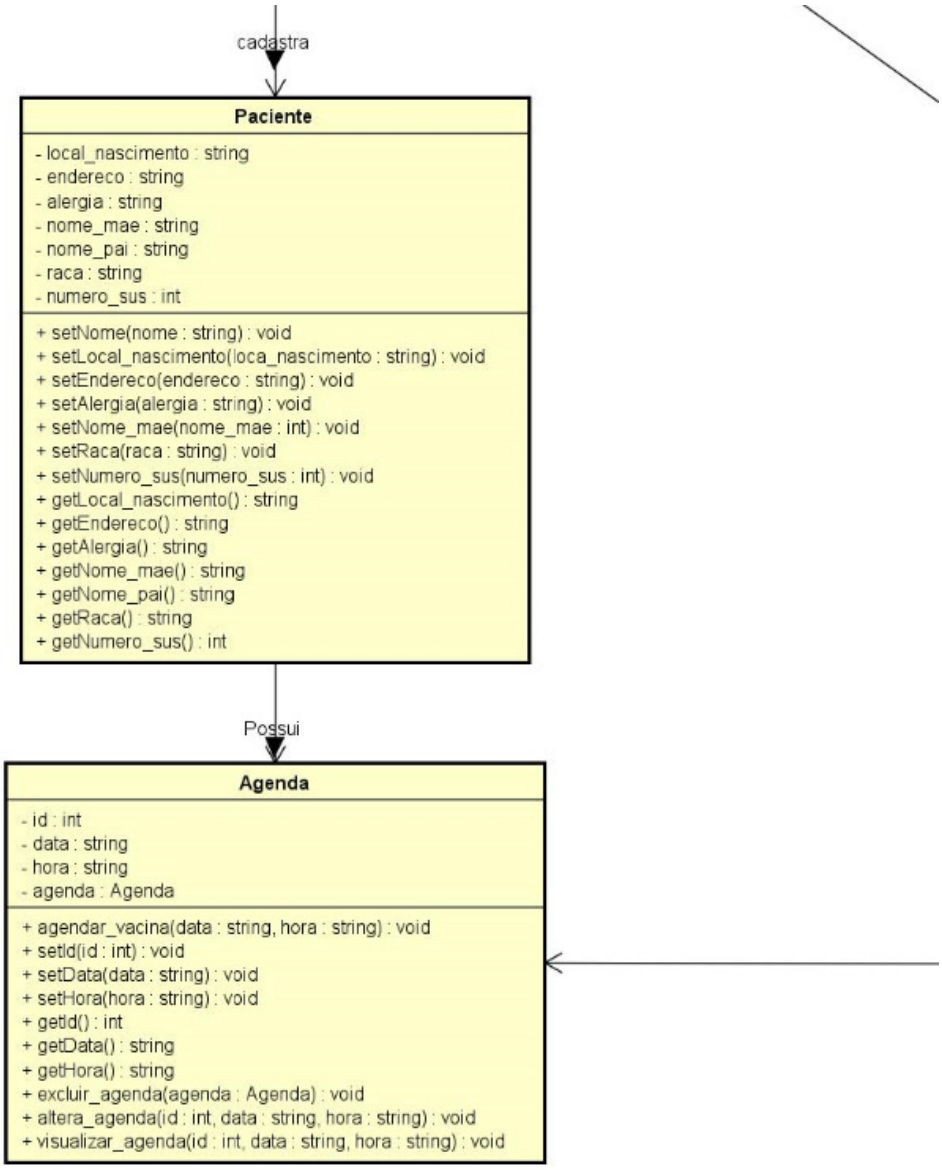
- View: Componente que contém as visões do projeto, ou seja, as interfaces, formulários e etc.;
- Controller: Componente que recebe as informações e requisições do pacote view e os despacha para devida classe de controle, o controller se comunica com classes BusinessLogic que contém as regras de negócio do sistema;
- Model: pacote que recebe as informações e requisições do pacote controller e os atribui às respectivas classes do pacote bean e do pacotepersistence,utilizando dados de seus modelos;
- Persistencia: pacote que recebe as requisições e realiza as operações relacionadas ao Banco de Dados, utilizando dados provindos do sistema. O pacote persistencia também pode enviar dados para o pacote view.
- Cliente: Inicia pedidos para servidores, Espera por respostas, Recebe respostas, Conecta-se a um pequeno número de servidores de uma só vez, Normalmente interage diretamente com os servidores através de seu software aplicação especifico, que lhe possibilita a comunicação com o servidor e Utiliza recursos da rede.

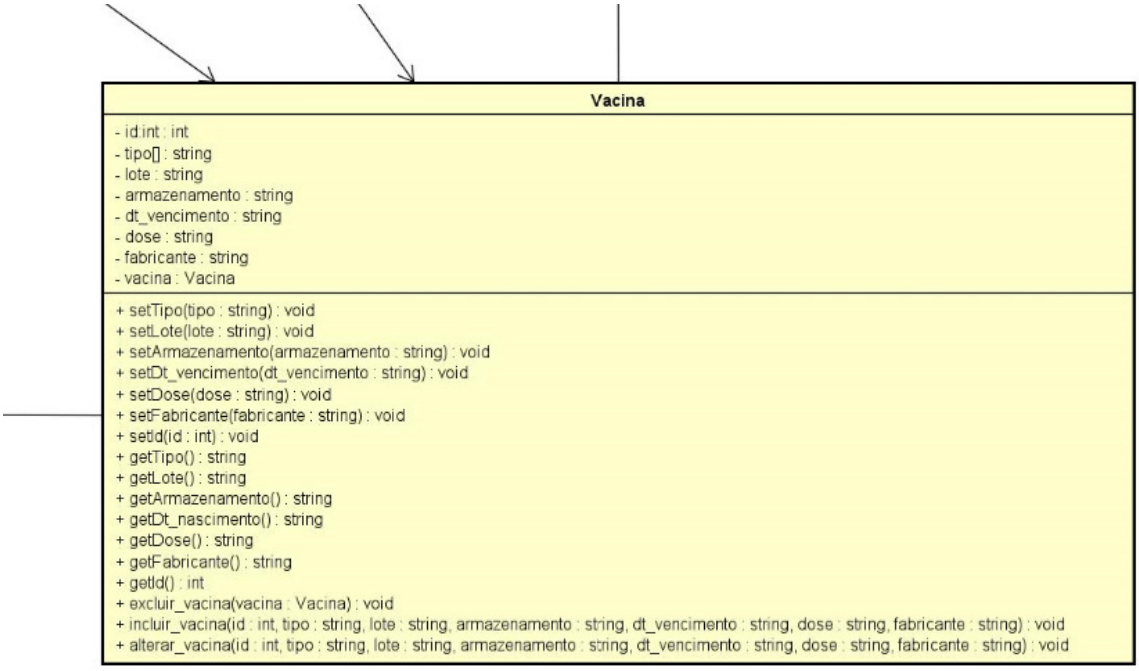
5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura


pkg
Pessoa
- id : int
- email : string
- senha : string
- cpf : string
- dt_nas : string
- tipo_pessoa[] : int
- nome : string
- pessoa : Pessoa
+ setId(id : int) : void
+ setEmail(email : string) : void
- setSenha(senha : string) : void
+ setCpf(cpf : string) : void
- setDt_nas(dt_nas : string) : void
+ setTipo_pessoa(tipo_pessoa : int) : void
+ setNome(nome : string) : void
- cadastrar_pessoa(email : string, senha : string, cpf : string, dt_nas : string, tipo_pessoa : string, nome : string) : void
- excluirPessoa(Pessoa : Pessoa) : boolean
- alterarSenha(senha : string) : void
- getId() : int
getEmail() : string
+ getSenha() : string
- getCpf() : string
- getDt_nas() : string
+ getTipo_pessoa() : string
+ getNome() : string
registra
registra
cadastra
cadastra
Registro_de_vacina
- id :int
- inicio : string
- fim : string
- registro : Registro_de_vacina
+ setInicio(inicio : string) : void
+ setFim(fim : string) : void
+ setId(id : int) : void
+ getInicio() : string
- getFim() : string
+ getId() : int
+ exporta_rg_vacina(tipo_pessoa[] : int, inicio : string, fim : string) : void
+ alterar_registro(inicio : string,fim : string) : void
+ excluir_registro(registro : Registro_de_vacina) : void
+ incluir_registro(id : int, inicio : string,fim : string) : void
local da aplicação de vacina
- id :int
- nome_local : string
+ setId(id : int) : void
+ setNome_local(local : string) : void
+ getId() : int
- getNome_local() : string
+ registrar_local(nome_local : string) : void
contem
Paciente
- local_nascimento : string
- endereco : string
- alergia : string
- nome_mae : string
- nome_pai : string
- raca : string
- numero_sus : int
+ setNome(nome : string) : void
+ setLocal_nascimento(local_nascimento : string) : void
+ setEndereco(endereco : string) : void
+ setAlergia(alergia : string) : void
+ setNome_mae(nome_mae : str) : void
+ setRaca(raca : string) : void
+ setNumero_sus(numero_sus : int) : void
+ getLocal_nascimento() : string
+ getEndereco() : string
+ getAlergia() : string
+ getNome_mae() : string
+ getNome_pai() : string
+ getRaca() : string
+ getNumero_sus() : int
Possui
Agenda
- id :int
- data : string
- hora : string
- agenda : Agenda
+ agendar_vacina(data : string,hora : string) : void
+ setId(id : int) : void
+ setData(data : string) : void
+ setHora(hora : string) : void
- getId() : int
+ getData() : string
+ getHora() : string
+ excluir_agenda(agenda :Agenda) : void
+ altera_agenda(id : int, data : string,hora : string) : void
+ visualizar_agenda(id : int, data :string,hora : string) : void
Possui
Vacina
- id :int
- tipo[] : string
- ide : string
- armazenamento : string
- dt_vencimento : string
- dose : string
- fabricante : string
- vacina : Vacina
+ setTipo(tipo : string) : void
+ setLote(lote : string) : void
+ setArmazenamento(armazenamento : string) : void
+ setDt_vencimento(dt_vencimento : string) : void
+ setDose(dose : string) : void
+ setFabricante(fabricante : string) : void
+ setId(id : int) : void
- getTipo() : string
- getLote() : string
- getArmazenamento() : string
+ getDt_nascimento() : string
+ getDose() : string
+ getFabricante() : string
+ getId() : int
+ excluir_vacina(vacina : Vacina) : void
+ incluir_vacina(id : int,tipo : string, lote : string, armazenamento : string, dt_vencimento : string, dose : string, fabricante : string) : void
+ alterar_vacina(id : int, tipo : string, lote : string, armazenamento : string, dt_vencimento : string, dose : string, fabricante : string) : void


Pessoa
- id : int
- email : string
- senha : string
- cpf : string
- dt_nas : string
- tipo_pessoa[] : int
- nome : string
- pessoa : Pessoa
+ setId(id : int) : void
+ setEmail(email : string) : void
+ setSenha(senha : string) : void
+ setCpf(cpf : string) : void
+ setDt_nas(dt_nas : string) : void
+ setTipo_pessoa(tipo_pessoa : int) : void
+ setNome(nome : string) : void
+ cadastrar_pessoa(email : string, senha : string, cpf : string, dt_nas : string, tipo_pessoa : string, nome : string) : void
+ excluirPessoa(Pessoa : Pessoa) : boolean
+ alterarSenha(senha : string) : void
+ getId() : int
+ getEmail() : string
+ getSenha() : string
+ getCpf() : string
+ getDt_nas() : string
+ getTipo_pessoa() : string
+ getNome() : string


Registro_de_vacina
- id : int
- inicio : string
- fim : string
- registro : Registro_de_vacina
+ setInicio(incio : string) : void
+ setFim(fim : string) : void
+ setId(id : int) : void
+ getInicio() : string
+ getFim() : string
+ getId() : int
+ exportar_rg_vacina(tipo_pessoa[] : int, inicio : string, fim : string) : void
+ alterar_registro(inicio : string, fim : string) : void
+ exluir_resgistro(registro : Registro_de_vacina) : void
+ incluir_registor(id : int, inicio : string, fim : string) : void
registra
local da aplicação de vacina
- id : int
- nome_local : string
+ setId(id : int) : void
+ setNome_local(local : string) : void
+ getId() : int
+ getNome_local() : string
+ registrar_local(nome_local : string) : void
contem
idastra


cadastra
Paciente
- local_nascimento : string
- endereco : string
- alergia : string
- nome_mae : string
- nome_pai : string
- raca : string
- numero_sus : int
+ setNome(nome : string) : void
+ setLocal_nascimento(loca_nascimento : string) : void
+ setEndereco(endereco : string) : void
+ setAlergia(alergia : string) : void
+ setNome_mae(nome_mae : int) : void
+ setRaca(raca : string) : void
+ setNumero_sus(numero_sus : int) : void
+ getLocal_nascimento() : string
+ getEndereco() : string
+ getAlergia() : string
+ getNome_mae() : string
+ getNome_pai() : string
+ getRaca() : string
+ getNumero_sus() : int
Possui
Agenda
- id : int
- data : string
- hora : string
- agenda : Agenda
+ agendar_vacina(data : string, hora : string) : void
+ setId(id : int) : void
+ setData(data : string) : void
+ setHora(hora : string) : void
+ getId() : int
+ getData() : string
+ getHora() : string
+ excluir_agenda(agenda : Agenda) : void
+ altera_agenda(id : int, data : string, hora : string) : void
+ visualizar_agenda(id : int, data : string, hora : string) : void



6. Visão de Processos

Para fornecer uma base para compreender a organização do processo do sistema, uma visualização arquitetural denominada visualização do processo é utilizada na disciplina Análise e Design. Existe apenas uma visualização do processo do sistema, que ilustra a decomposição do processo do sistema, incluindo o mapeamento de classes e subsistemas para processos e encadeamentos. A visão de processos é refinada durante cada iteração. Com a UML, os aspectos estáticos e dinâmicos desta visualização são capturados nos mesmos tipos de diagramas que a visualização de design, ou seja, diagramas de classes, diagramas de interação, diagramas de atividades e diagramas de estados, mas com um foco nas classes ativas que representam esses encadeamentos e processos". Simultaneidade, tempo de resposta, conflito, rendimento do processamento, tolerância a falhas e escalabilidade são preocupações ao construir e utilizar a visualização do processo.

É possível projetar a simultaneidade sem utilizar o suporte direto ao sistema operacional básico usando, por exemplo, um programador especialmente escrito ou outro suporte em tempo de execução. Nesses casos, a simultaneidade é simulada no nível da infra-estrutura do aplicativo, e não no sistema operacional. Se necessário, outros estereótipos (além dos threads e processos padrão) podem ser usados para fazer essa distinção (a fim de conduzir a implementação).

7. Visão da Implementação

Para fornecer uma base que permitirá compreender a distribuição física do sistema em um conjunto de nós de processamento, uma visualização arquitetural chamada **visualização de implementação** é utilizada no fluxo de trabalho Análise & Design. A visão de implementação (uma das cinco visões) ilustra a distribuição do processamento em um conjunto de nós do sistema, incluindo a distribuição física dos processos e threads. Ela é refinada durante cada iteração.

8. Visão de Dados

Há armazenagem é através do banco de dados MySql, que recebe as requisições e realiza as operações relacionadas ao Banco de Dados, utilizando dados provindos do sistema.

9. Tamanho e Desempenho

O produto deve fazer uso criterioso do armazenamento em cache para que mesmo com uma conexão ruim, ou inconstante, o usuário consiga utilizar o sistema.

Apesar de precisar de requisições externas para a comunicação, essa aplicação não tende a sofrer muitas quedas de desempenho, inclusive pode ser usado em sistemas com menor poder de processamento e memória.

10. Qualidade

O padrão de arquitetura adotado no projeto tem como finalidade garantir uma melhor organização do código-fonte, o que auxilia na manutenibilidade do software, bem como a portabilidade do mesmo.

O software garantirá a segurança dos dados informados pelo usuário, além de disponibilizar ferramentas simples, funcionais e intuitivas.