UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

SISTEMAS DE INFORMAÇÃO

Gabreil Morais

Tales Félix

Padrão de Projeto - Model View Controller X Observer

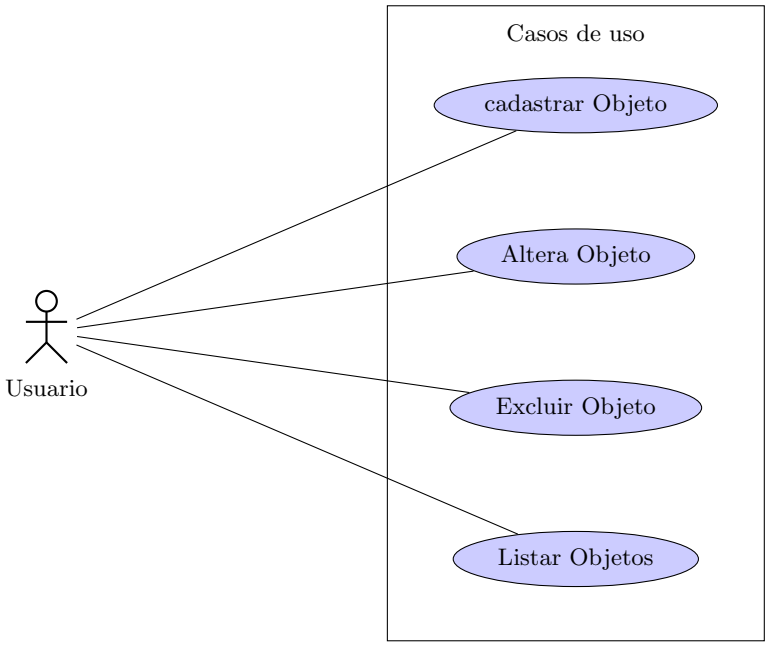
Docente: Eduardo Pelli.

Diamantina, 18 de Outubro de 2020

1 Projeto em MVC - Introdução

Um colecionador precisa de um sistema para guardar informações de seus objetos. O sistema deverá fazer registro do nome e descrição geral de cada pertence. Ao ser solicitado pelo usuário o sistema deverá retonar uma lista contendo todos objetos cadastrados.

1.1 Diagrama de Casos de Uso



1.2 Fluxo de Eventos

1.2.1 Cadastrar Objeto

O usuário Deverá preencher os campos, "Nome" e "Descrição" e sobmeter clicando no botão Enviar.
Aparecerá uma mensagem de sucesso, caso contrário mostrarar um erro e o usuário terá que repetir a operação.

1.2.2 Listar Objetos

Após clicar em "Listar" aparecerar uma lista de elementos.
Caso não tenha nenum matrial cadrastado mostrarar a lista vazia.

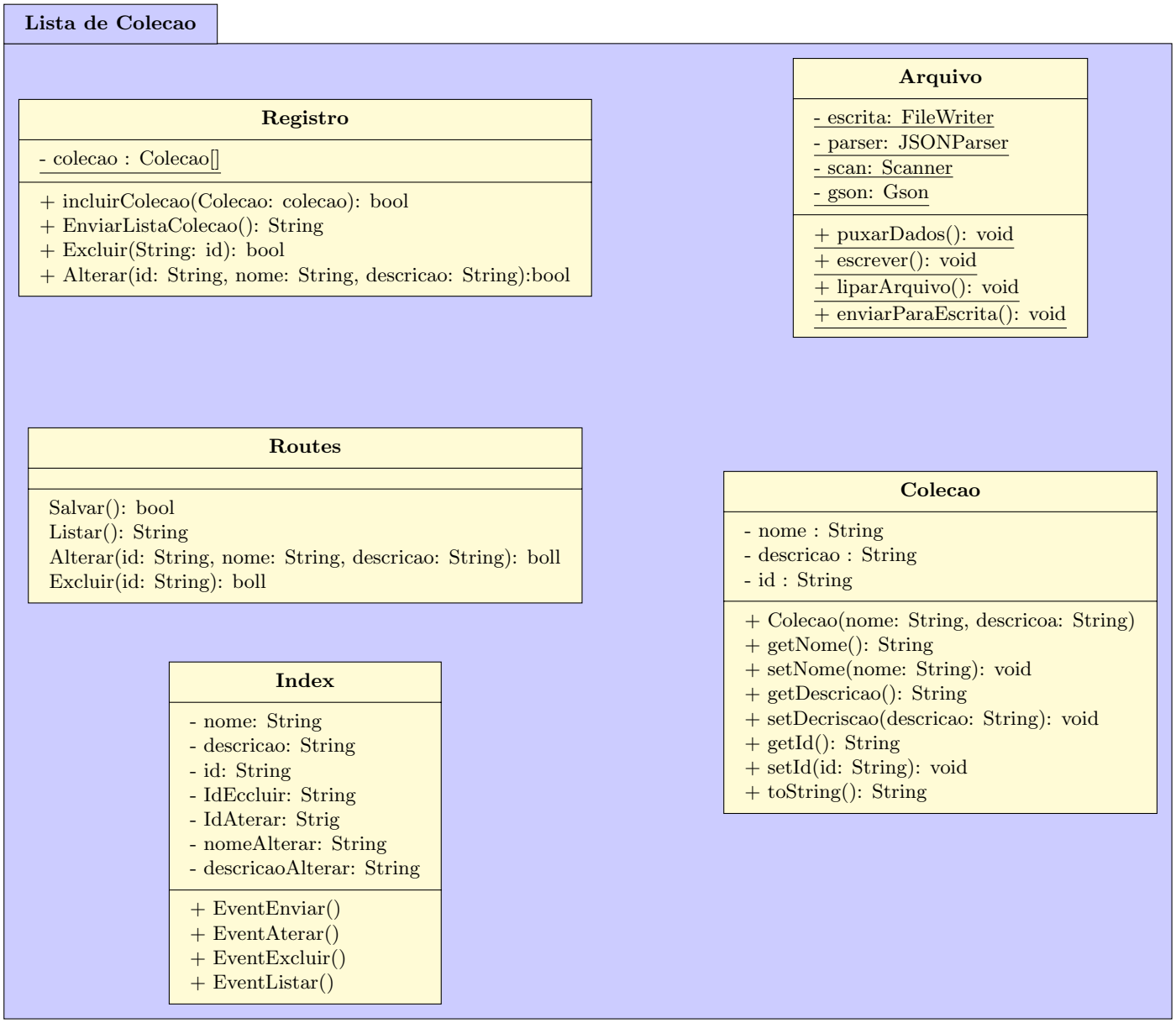
1.2.3 Alterar Objetos

O usuário Deverá preencher os campos, "Nome", "Descrição" e "ID" e submeter
Se os ID do objeto que deseja alterar não estiver devidadamente preenchido mostrarar uma mensagem de erro.
Caso contrário aparecerá uma mensagem de sucesso

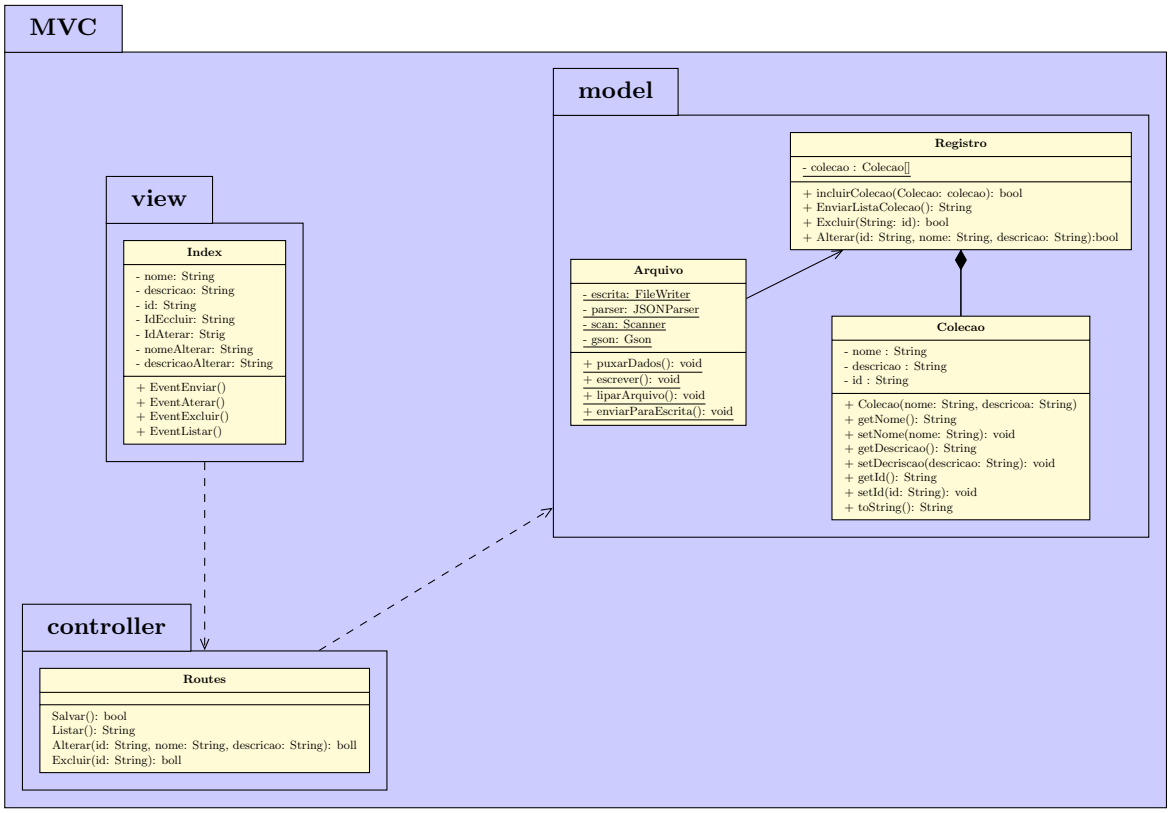
1.2.4 Excluir Objeto

O usuário Deverá preencher "ID" e submeter
Se os ID do objeto que deseja excluir não estiver devidadamente preenchido mostrarar uma mensagem de erro.
Caso contrário aparecerá uma mensagem de sucesso

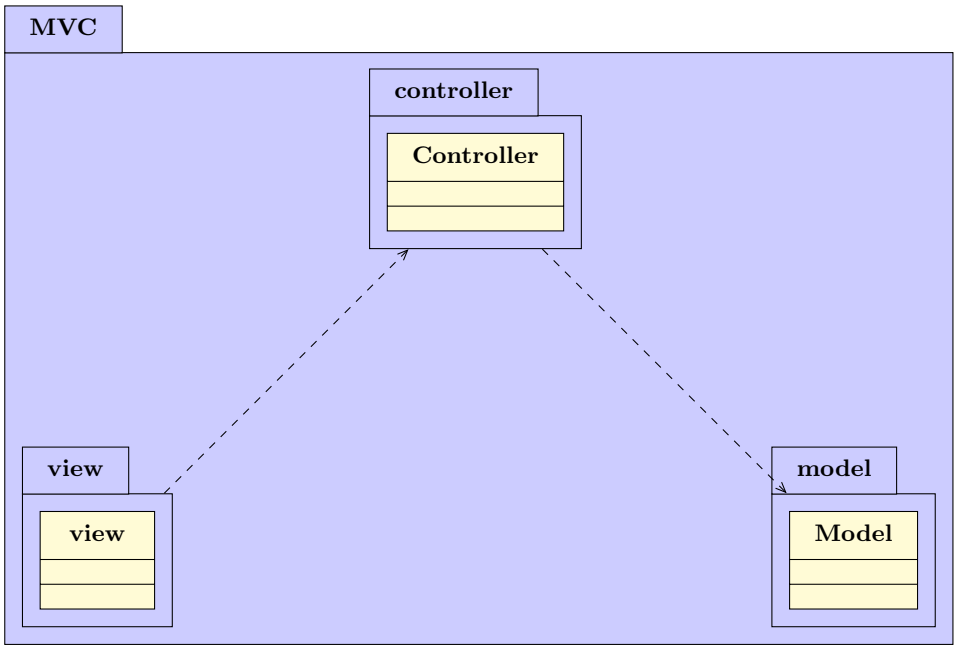
1.3 Classes


Lista de Colecao
Registro
- colecao : Colecao[]
+ incluirColecao(Colecao: colecao): bool
+ EnviarListaColecao(): String
+ Excluir(String: id): bool
+ Alterar(id: String, nome: String, descricao: String):bool
Arquivo
- escrita: FileWriter
- parser: JSONParser
- scan: Scanner
- gson: Gson
+ puxarDados(): void
+ escrever(): void
+ liparArquivo(): void
+ enviarParaEscrita(): void
Routes
Salvar(): bool
Listar(): String
Alterar(id: String, nome: String, descricao: String): boll
Excluir(id: String): boll
Colecao
- nome : String
- descricao : String
- id : String
+ Colecao(nome: String, descricoa: String)
+ getNome(): String
+ setNome(nome: String): void
+ getDescricao(): String
+ setDecriscao(descricao: String): void
+ getId(): String
+ setId(id: String): void
+ toString(): String
Index
- nome: String
- descricao: String
- id: String
- IdEccluir: String
- IdAterar: Strig
- nomeAlterar: String
- descricaoAlterar: String
+ EventEnviar()
+ EventAterar()
+ EventExcluir()
+ EventListar()

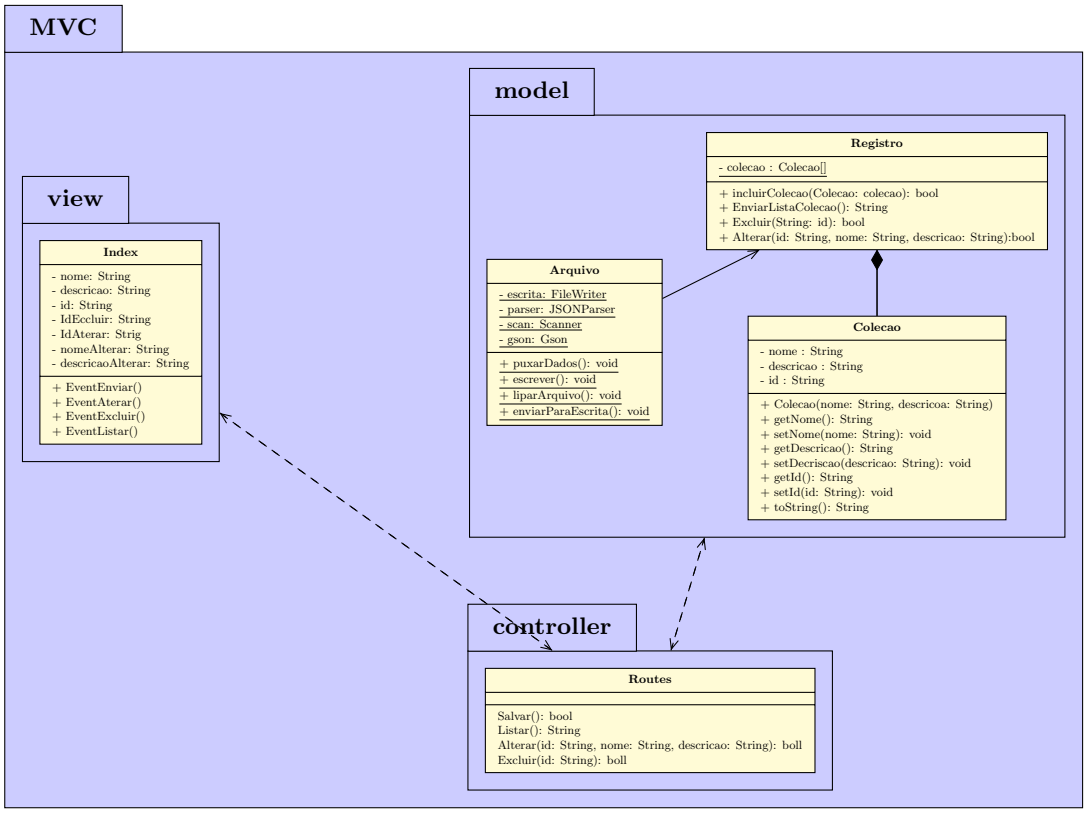
1.4 Diagrama de Classes Lista de Coleções



1.5 Diagrama de classe MVC



1.6 Diagrama de classe MVC

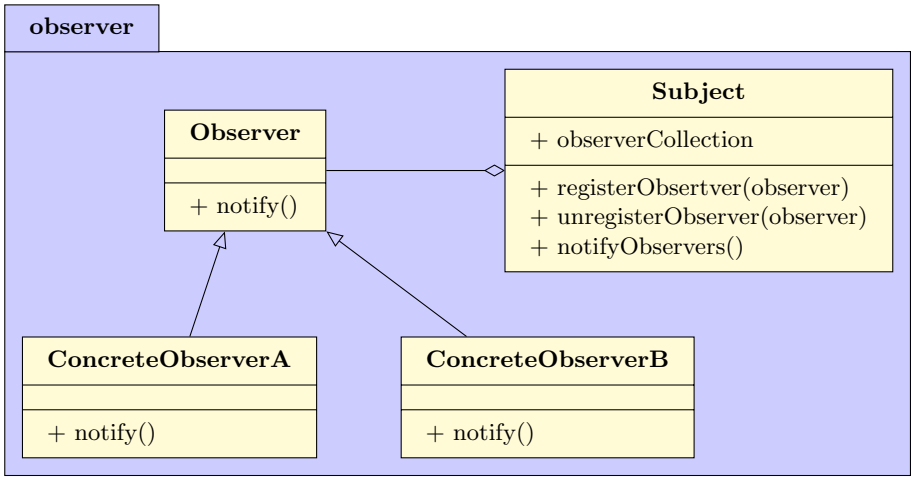


2 Observer

2.1 Objetivo

Observer é um padrão de projeto comportamental que permite que um objeto notifique outros objetos sobre a mudança em seu estado. O padrão Observer fornece uma maneira de assinar e cancelar a assinatura desses eventos para qualquer objeto que implemente uma interface de assinante. Sendo assim o observar é um padrão de projeto que tem como princípio a ligação leve. Sendo assim, o princípio da ligação leve procurar objetos levemente ligados um com o outro, para que eles saibam muito pouco de cada um, sendo assim interagindo normalmente.

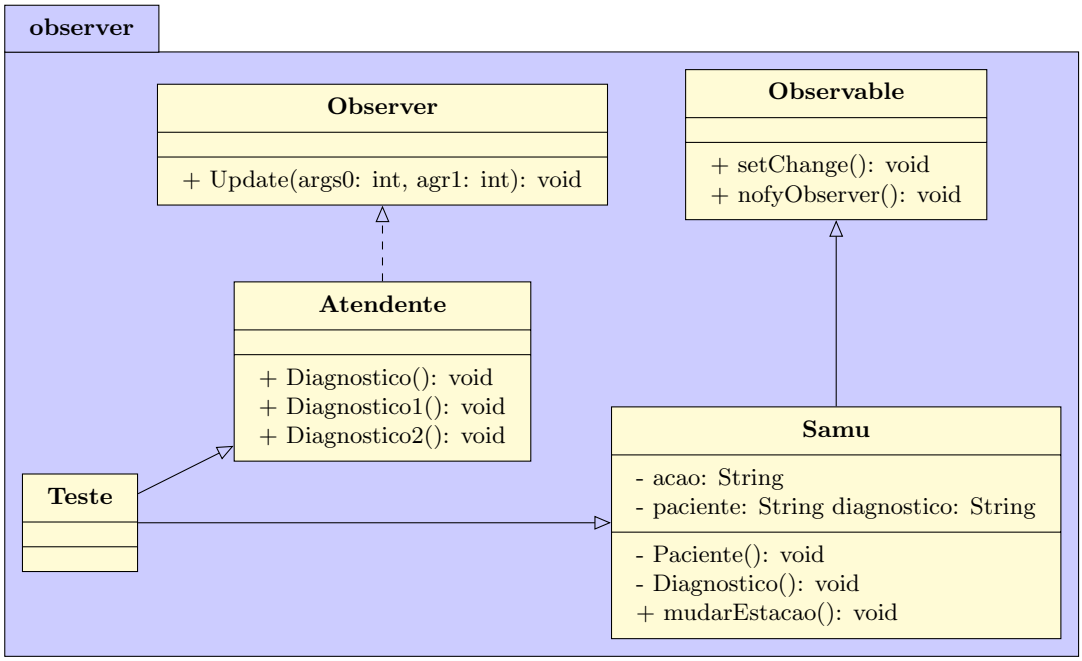
2.2 Diagrama Observer



2.3 Diagrama de classes

Este projeto é uma implementação do padrão de projeto observer, em um cenário de um resgate do samu, e a chegada no hospital, sendo assim ele designa depois de um leve diagnóstico a que local do hospital, o paciente s erá encaminhado.
Devido a esse cenário temos 2 classes , a Atendente e o Samu. Na qual a atendente e uma classe observadora da classe samu, contendo nela três métodos diaguinosticar1():void, diaguinosticar2():void, diaguinosticar3():void e update que é um método que foi declarado na interface observer.
Tendo também a classe Samu na qual tem os atributos acao String , diagnóstico String, e paciente String e sendo os métodos paciente():void , diagnostico():void e mudaEstado():void . Devido a esses métodos, o diagnóstico passa no método, faz uma comparação passando para a variável ação que por sua vez se chama this.mudaEstado, sendo assim notificando seu observador, que por sua vez na sua função update compara a ação e chama um método do atendente . Prós x contras.
Princípio do aberto ou fechado pode-se introduzir novas classes assinantes a qualquer momento. Devido que um objeto observado pode receber novos observadores a qualquer momento. Sendo assim, gerando conexões com objetos a qualquer momento.
O ponto negativo, assinantes são notificados de forma. Se é emitido um sinal que muda o estado do objeto observado, então todos os observadores vão reagir a essa mudança de estado.

2.4 Diagrama de Classes Observer



3 Conclusão

O Padrão de Projeto Model View Controller é um padrão de arquitetura de Software

References