Universidade Estadual de Maringá
Departamento de Informática
Curso: Informática
Disciplina: Programação de Sistemas Web – 5188
Professora: Juliana Keiko Yamaguchi

Programação de Sistemas WEB
Trabalho Prático: Parte II

Integrante	RA
Douglas Mezuraro	95676
Tatiane Lie Takeshima	87971
Victor Glauber L. Silva	68474

Sumário

Domínio do sistema

Este documento especifica os requisitos de um projeto de software para uma clínica médica, fornecendo as informações necessárias para o desenvolvimento do mesmo.

A necessidade de criação do sistema surgiu pois os médicos da clínica não terem controle do fluxo dos pacientes que atendem e consequentemente quando precisam recuperar informações sobre os últimos atendimentos despendem muito tempo procurando nos prontuários físicos de atendimento. Nem sempre a secretária consegue se lembrar de confirmar as consultas com os pacientes com um dia de antecedência, sendo que muitos deles esquecem e os médicos ficam com tempo não preenchido em seus consultório.

A clínica em questão, por ser de pequeno porte, não possui grande quantidade de equipamentos tecnológicos, apenas dois computadores pessoais, uma utilizado por um usuário que trabalha na recepção e um computador pessoal para cada médico da clínica, todos com acesso a internet.

Arquitetura e tecnologias

O sistema foi implementado seguindo a arquitetura MVC (*model-view-controller*) modelo *fat-client* onde toda a regra de negócio fica no lado do cliente. Não foi persistido os dados em banco de dados, ao invés, foi mantido os dados em memória utilizando listas.

Foi-se utilizado o java como linguagem de programação aplicando os conceitos de orientação à objetos.

Foram utilizados dois frameworks para facilitar o desenvolvimento e aumentar a produtividade, sendo eles:

- JSF (*Java Server Faces*): Framework para facilitar o desenvolvimento de interfaces para o usuário
- Bootstrap: Framework *front-end* para estilização de interfaces para o usuário melhorando sua experiência com a aplicação

Além dos frameworks foi utilizado a biblioteca *OmniFaces* que tem como objetivo facilitar o uso do *JSF*.

Melhorias em relação a primeira parte

- *Login:* Foi implementada funcionalidade de login que permite o usuário acessar o sistema
- *Correção de bugs na navegação entre páginas:* Foi corrigido a navegação entre as páginas seguindo o padrão do framework JSF
- *Aperfeiçoamento das views:* Foi aperfeiçoado o uso correto de tags HTML antes não usadas, além do aperfeiçoamento da estilização das views
- *Implementado validação de campos obrigatórios*: Foi feito a validação dos campos que o usuário obrigatoriamente deve informar para completar determinadas ações
- *Delegação de responsabilidades:* Na primeira parte os beans estavam sendo usados como controladores. Então os controladores foram refatorados para classes singletons e os beans para atuar na camada de visão
- *Aperfeiçoamento da barra de navegação*: Foi aperfeiçoada para ser uma página em comum para todas as demais páginas (ao invés de duplicar código em toda view como foi feito na primeira parte)

Classes do sistema

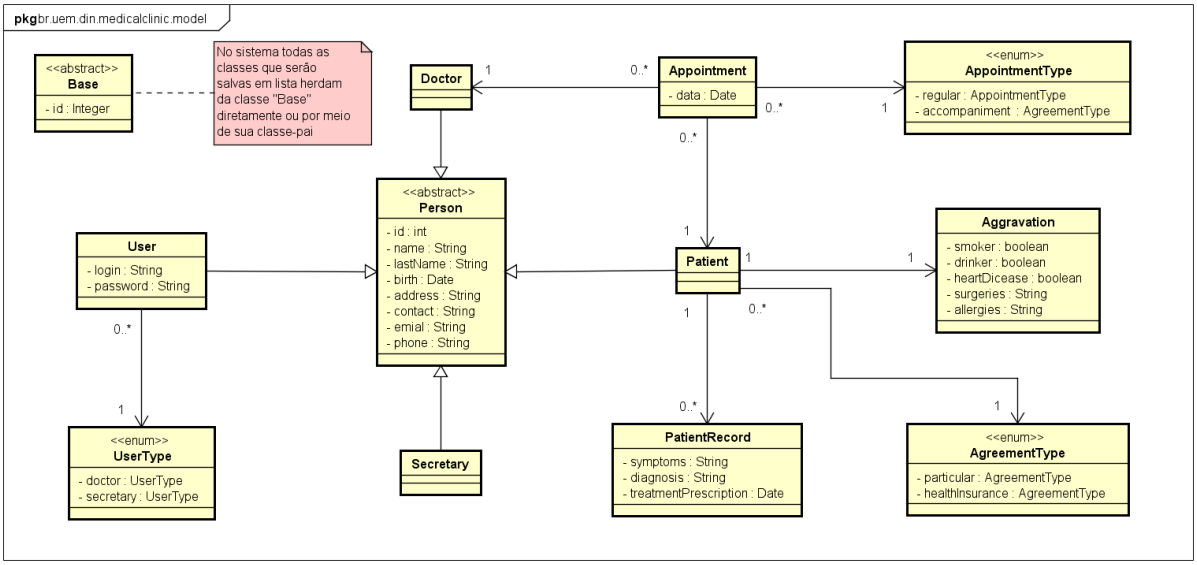
Tipo	Nome	Objetivo
br.uem.din.medicalclinic.model		
Entidade	Aggravation	Indica os agravantes de riscos que um paciente possui, como, se o paciente fuma, bebe, possui doenças cardíacas, alergias que possui e cirurgias já realizadas
Enumerado	AgreementType	Indica o tipos de convênio que um paciente pode ter, sendo eles, particular e plano de saúde
Entidade	Appointment	Indica as informações sobre a consulta, como, o médico que irá consultar, o paciente a ser consultado, a data da consulta e o tipo da consulta
Enumerado	AppointmentType	Indica o tipo que uma consulta pode ser, sendo elas, normal e de retorno
Classe	Base	Classe de qual toda entidade cadastrável do sistema herda, possui o atributo id que é o indicador único de cada registro pelo qual é usado para recuperar o mesmo
Entidade	Doctor	Indica as informações pessoais de um médico (para mais detalhes ver classe Person)
Entidade	Patient	Indica as informações pessoais de um paciente (para mais detalhes ver classe Person)
Entidade	PatientRecord	Indica o prontuário de uma consulta de um paciente, como os sintomas apresentados, o diagnóstico feito pelo médico e a prescrição do tratamento
Entidade	Person	Indica os dados pessoais em que todas pessoas do sistema tem em comum, como, nome, último nome, aniversário, endereço, nome da pessoa para contato, endereço de e-mail e número de telefone

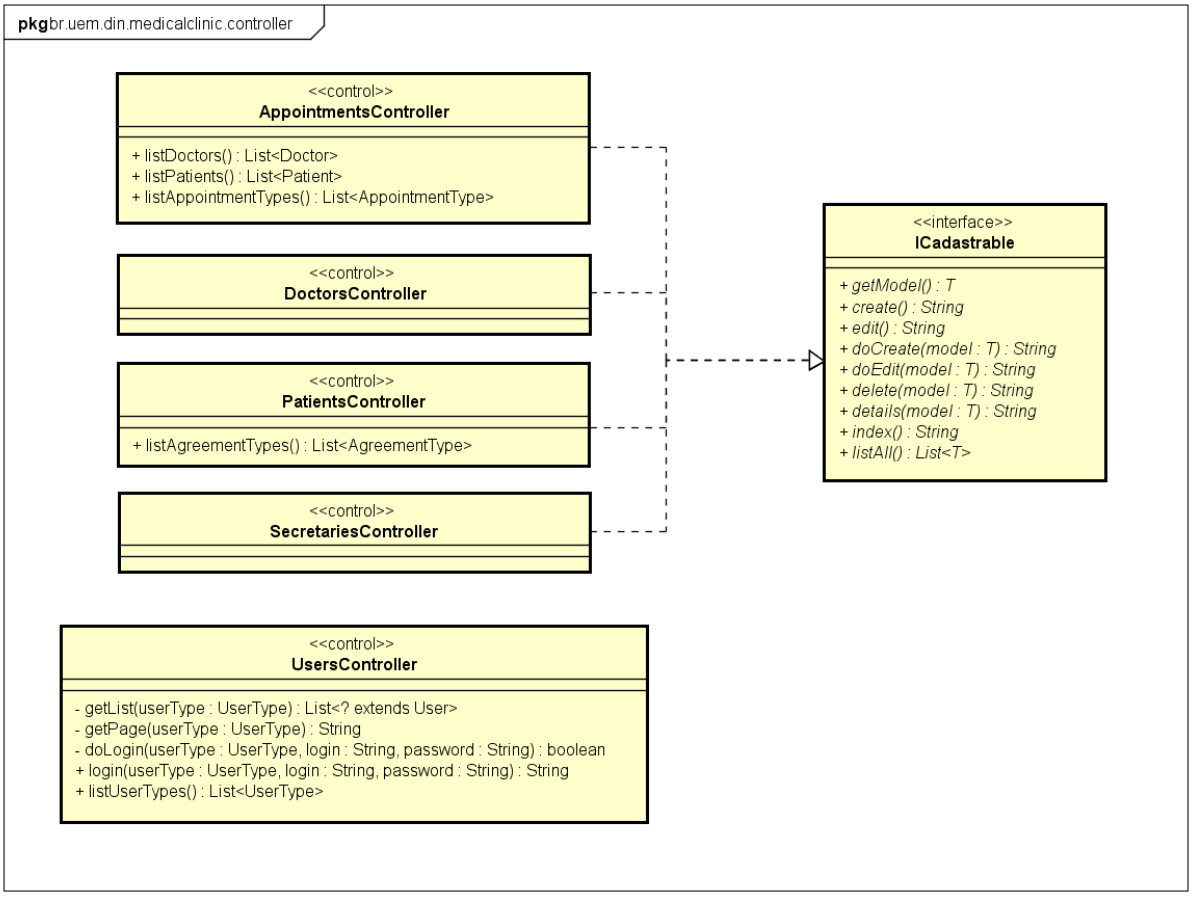
Entidade	Secretary*	Indica as informações pessoais de um secretário (para mais detalhes ver classe Person)
Entidade	User*	Indica as informações sobre o usuário do sistema, como, login e senha
Enumerado	UserType*	Indica os tipos de usuários que acessam o sistema, sendo eles, médico e secretário
br.uem.din.medicalclinic.controller		
Interface	ICadastrable	Interface que define os métodos os quais todos os controladores de cruds deverão implementar
Controlador	AppointmentsController	Controlador da entidade Appointment
Controlador	DoctorsController	Controlador da entidade Doctor
Controlador	PatientsController	Controlador da entidade Patient
Controlador	SecretariesController	Controlador da entidade Secretary
Controlador	UsersController*	Controlador de acesso ao sistema
br.uem.din.medicalclinic.bean.appointment		
ManagedBean	Create*	Bean que representa a view de cadastro de consultas
ManagedBean	Delete*	Bean que representa a view de exclusão de consultas
ManagedBean	Details*	Bean que representa a view de detalhes de consultas
ManagedBean	Edit*	Bean que representa a view de edição de consultas
ManagedBean	Index*	Bean que representa a view inicial de consultas
br.uem.din.medicalclinic.bean.doctor		

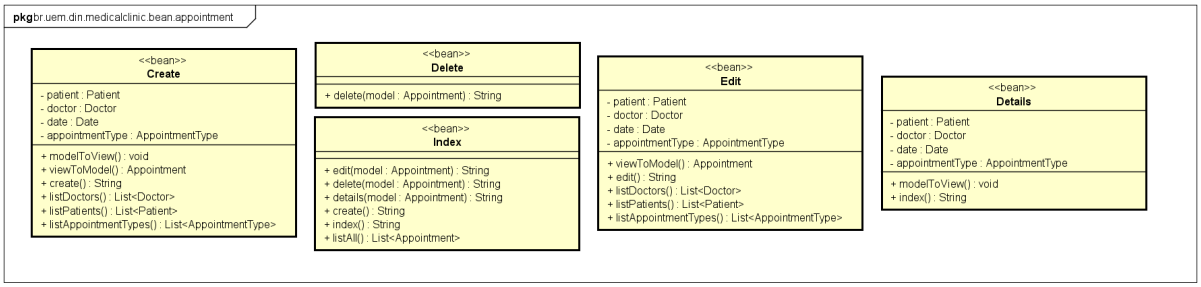
ManagedBean	Create*	Bean que representa a view de cadastro de médicos
ManagedBean	Delete*	Bean que representa a view de exclusão de médicos
ManagedBean	Details*	Bean que representa a view de detalhes de médicos
ManagedBean	Edit*	Bean que representa a view de edição de médicos
ManagedBean	Index*	Bean que representa a view inicial de médicos
br.uem.din.medicalclinic.bean.patient		
ManagedBean	Create*	Bean que representa a view de cadastro de pacientes
ManagedBean	Delete*	Bean que representa a view de exclusão de pacientes
ManagedBean	Details*	Bean que representa a view de detalhes de pacientes
ManagedBean	Edit*	Bean que representa a view de edição de pacientes
ManagedBean	Index*	Bean que representa a view inicial de pacientes
br.uem.din.medicalclinic.bean.secretary		
ManagedBean	Create*	Bean que representa a view de cadastro de secretários
ManagedBean	Delete*	Bean que representa a view de exclusão de secretários
ManagedBean	Details*	Bean que representa a view de detalhes de secretários
ManagedBean	Edit*	Bean que representa a view de edição de secretários
ManagedBean	Index*	Bean que representa a view inicial de secretários
br.uem.din.medicalclinic.bean.user		

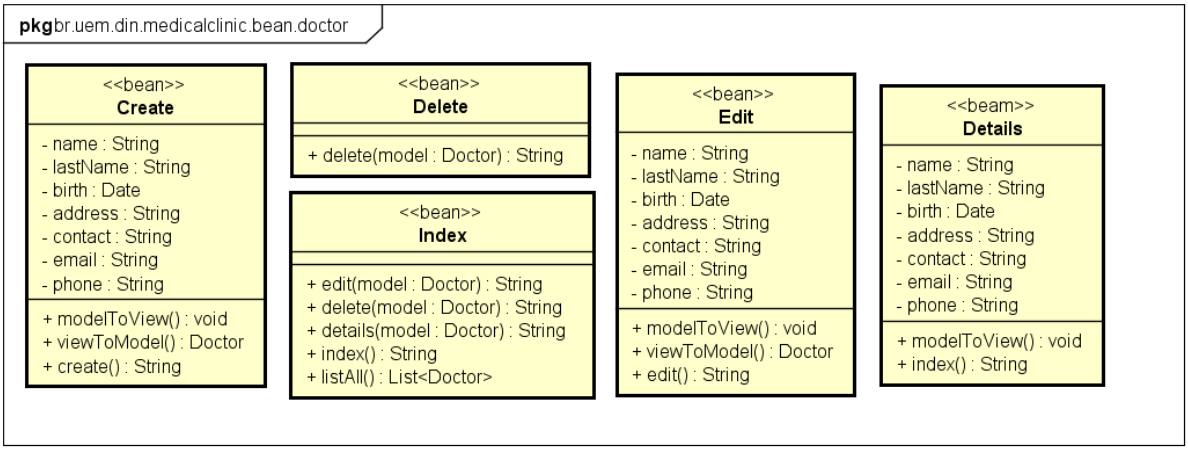
ManagedBean	Login*	Bean que representa a view de login do sistema
br.uem.din.medicalclinic.utils		
Classe	BaseList	Classe que implementa representa uma lista de objetos que herdam da classe Base onde é possível fazer as operações de CRUD
Classe	Populate*	Classe que retorna objetos exemplos das entidades de médico, paciente e secretário
* Classe nova no sistema		

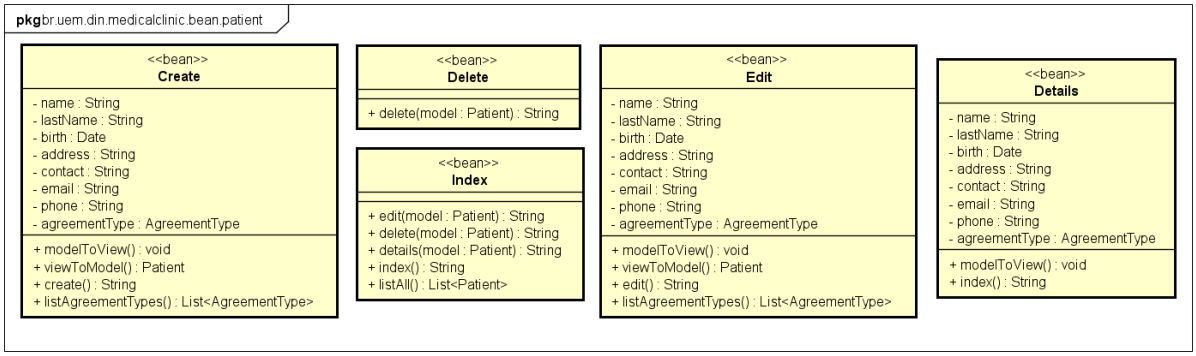
Diagrama de classes

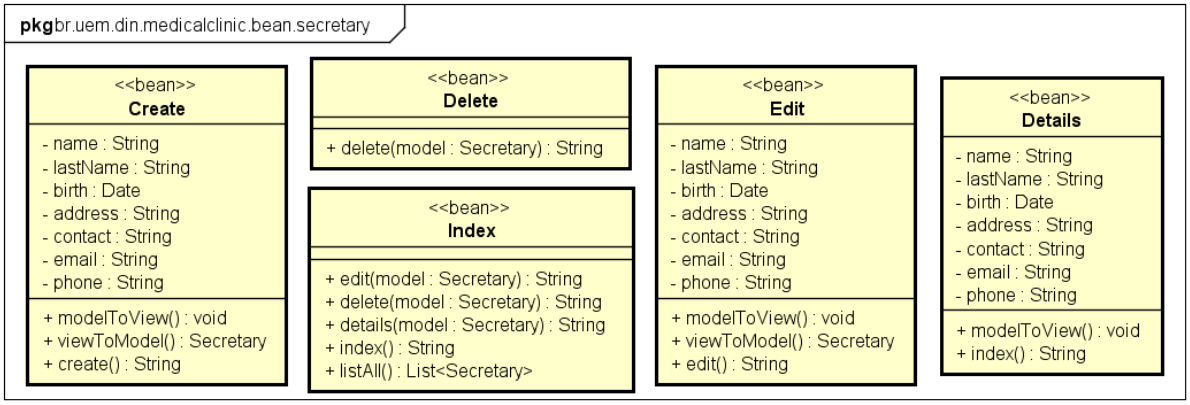


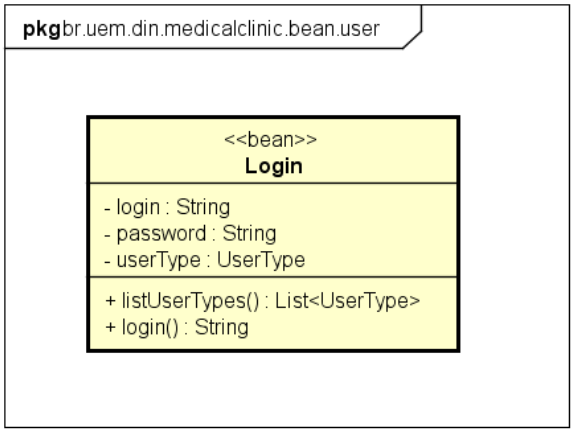


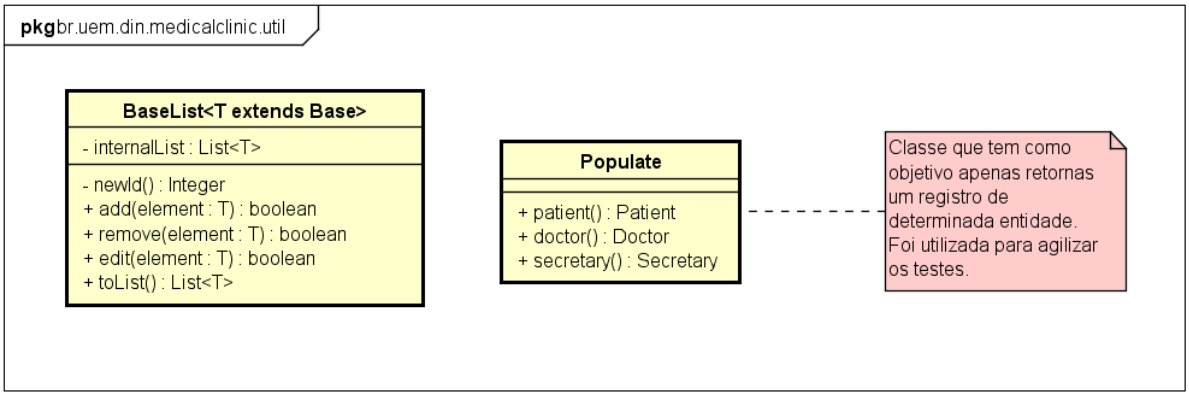

pkgbr.uem.din.medicalclinic.bean.appointment
<<bean>>
Create
- patient : Patient
- doctor : Doctor
- date : Date
- appointmentType : AppointmentType
+ modelToView() : void
+ viewToModel() : Appointment
+ create() : String
+ listDoctors() : List<Doctor>
+ listPatients() : List<Patient>
+ listAppointmentTypes() : List<AppointmentType>
<<bean>>
Delete
+ delete(model : Appointment) : String
<<bean>>
Index
+ edit(model : Appointment) : String
+ delete(model : Appointment) : String
+ details(model : Appointment) : String
+ create() : String
+ index() : String
+ listAll() : List<Appointment>
<<bean>>
Edit
- patient : Patient
- doctor : Doctor
- date : Date
- appointmentType : AppointmentType
+ viewToModel() : Appointment
+ edit() : String
+ listDoctors() : List<Doctor>
+ listPatients() : List<Patient>
+ listAppointmentTypes() : List<AppointmentType>
<<bean>>
Details
- patient : Patient
- doctor : Doctor
- date : Date
- appointmentType : AppointmentType
+ modelToView() : void
+ index() : String


pkgbr.uem.din.medicalclinic.bean.doctor
<<bean>>
Create
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ viewToModel() : Doctor
+ create() : String
<<bean>>
Delete
+ delete(model : Doctor) : String
<<bean>>
Index
+ edit(model : Doctor) : String
+ delete(model : Doctor) : String
+ details(model : Doctor) : String
+ index() : String
+ listAll() : List<Doctor>
<<bean>>
Edit
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ viewToModel() : Doctor
+ edit() : String
<<beam>>
Details
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ index() : String


pkgbr.uem.din.medicalclinic.bean.patient
<<bean>>
Create
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
- agreementType : AgreementType
+ modelToView() : void
+ viewToModel() : Patient
+ create() : String
+ listAgreementTypes() : List<AgreementType>
<<bean>>
Delete
+ delete(model : Patient) : String
<<bean>>
Index
+ edit(model : Patient) : String
+ delete(model : Patient) : String
+ details(model : Patient) : String
+ index() : String
+ listAll() : List<Patient>
<<bean>>
Edit
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
- agreementType : AgreementType
+ modelToView() : void
+ viewToModel() : Patient
+ edit() : String
+ listAgreementTypes() : List<AgreementType>
<<bean>>
Details
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
- agreementType : AgreementType
+ modelToView() : void
+ index() : String


pkgbr.uem.din.medicalclinic.bean.secretary
<<bean>>
Create
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ viewToModel() : Secretary
+ create() : String
<<bean>>
Delete
+ delete(model : Secretary) : String
<<bean>>
Index
+ edit(model : Secretary) : String
+ delete(model : Secretary) : String
+ details(model : Secretary) : String
+ index() : String
+ listAll() : List<Secretary>
<<bean>>
Edit
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ viewToModel() : Secretary
+ edit() : String
<<bean>>
Details
- name : String
- lastName : String
- birth : Date
- address : String
- contact : String
- email : String
- phone : String
+ modelToView() : void
+ index() : String

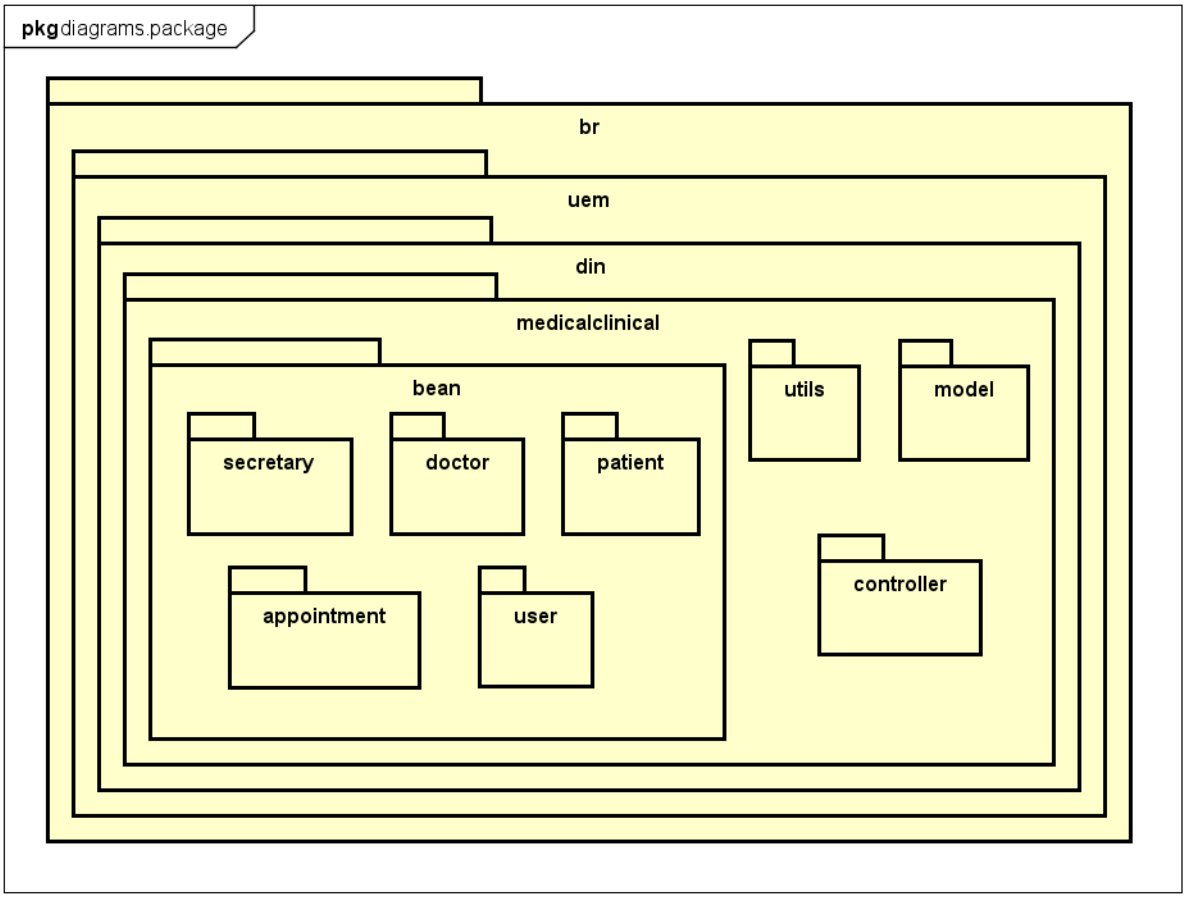




Dentre as classes descritas acima temos a tabela que indica quais entidades foram implementadas o CRUD:

Entidade	Cadastrar	Editar	Excluir	Detalhar
Aggravation	Não	Não	Não	Não
Appointment	Sim	Sim	Sim	Sim
Doctor	Sim	Sim	Sim	Sim
Patient	Sim	Sim	Sim	Sim
PatientRecord	Não	Não	Não	Não
Secretary	Sim	Sim	Sim	Sim

Diagrama de pacotes



MVC	Pacote	
View	br.uem.din.medicalclinc.bean	appointment
		doctor
		patient
		secretary
		user
Controller	br.uem.din.medicalclinc.controller	
Model	br.uem.din.medicalclinc.model	
N/A	br.uem.din.medicalclinc.utils	

Camada	Descrição

Model	São as classes de entidades levantadas com a análise de requisitos. As entidades são classes são abstrações de algo real na clínica médica. É também a classe que possuem as informações que irão ser persistidas
View	Interfaces de usuário (páginas HTMLs). As visões são responsáveis pela exibição de dados para o usuário. Atualmente no sistema existe uma view para cada operação de CRUD de cada entidade
Controller	O controlador na aplicação tem o papel de receber a informação e ação da view e a partir disto controlar e atualizar o modelo. O controlador também devolve para o browser qual visão irá ser exibida de acordo com a ação executada pelo usuário.