Relatório de Especificação Lógica

Sumário:	Este relatório lógico documenta o desenvolvimento e a estrutura do sistema de gestão clínica, abordando os principais modelos e componentes necessários para o funcionamento da aplicação. O sistema foi projetado para simplificar o gerenciamento de consultas, registos clínicos e prescrições, promovendo um atendimento mais eficiente e seguro na clínica. O sistema garante que diferentes tipos de utilizadores – médicos, funcionários e gestores – possam realizar tarefas específicas dentro das suas permissões.

Data de preparação:	23/09/2024
Grupo:	João Gabriel (Nº 117589)
	Luís Assis (Nº 112763)
	Luís Nantes (Nº 120401)
	Pedro Sampaio (Nº 119213)
	Rodrigo Ferreira (Nº 120099)
Professor orientador	João Gonçalo Gomes de Paiva Dias
Versão:	v1.0
Circulação:	ESTGA-UA

Índice

1 Introdução

Este relatório lógico descreve a estrutura e funcionamento do sistema de gestão da clínica médica, detalhando o modelo de dados, a lógica de funcionamento e as interações essenciais para a operação integrada e segura das funcionalidades. Este sistema visa solucionar as dificuldades previamente enfrentadas na administração de consultas, prescrições e no acesso seguro aos dados clínicos dos pacientes, centralizando todas essas operações numa única plataforma digital.

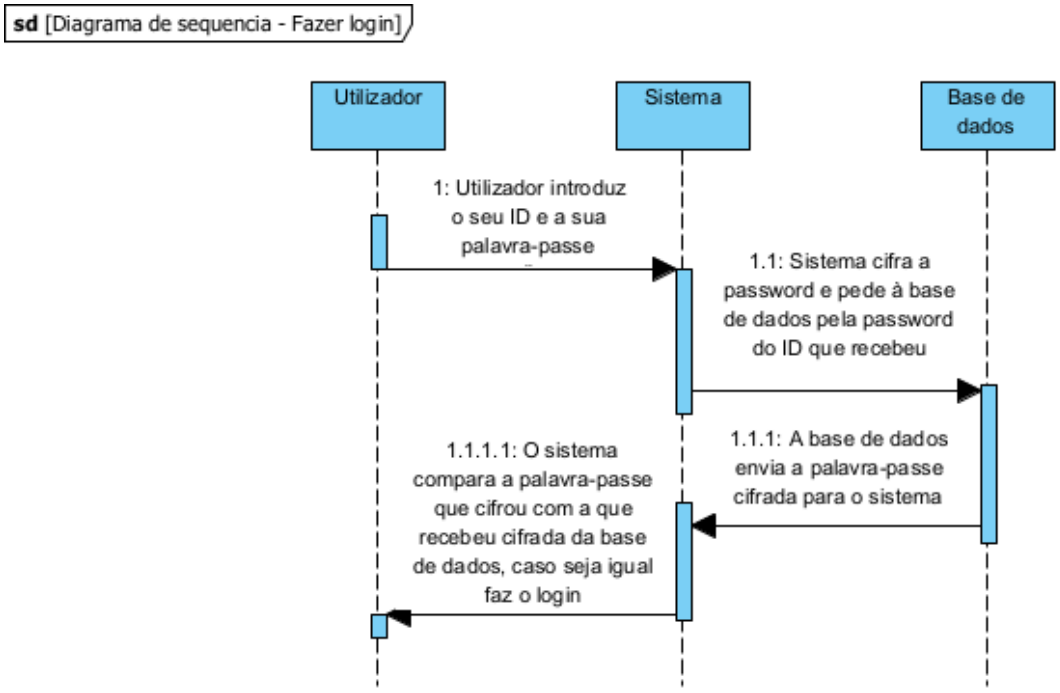
O desenvolvimento do sistema seguiu um ciclo de análise e design cuidadoso, baseado nos requisitos previamente especificados. Durante este ciclo, foram considerados padrões de segurança, integridade dos dados e usabilidade, especialmente importantes no contexto clínico. Além disso, houve uma preocupação constante em garantir a conformidade com as normativas de proteção de dados. A estrutura lógica descrita neste relatório reflete o alinhamento entre as funcionalidades propostas e as necessidades de segurança e eficiência identificadas.

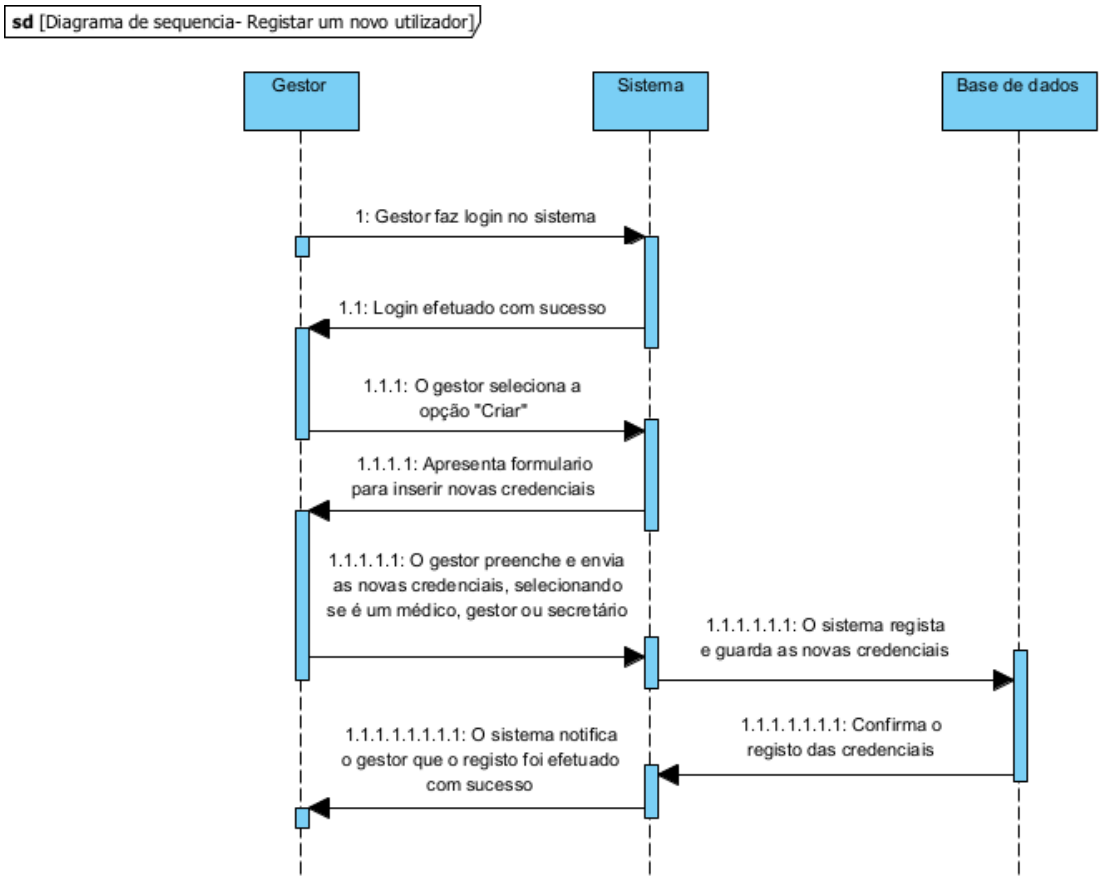
No contexto da clínica, o sistema permite que:

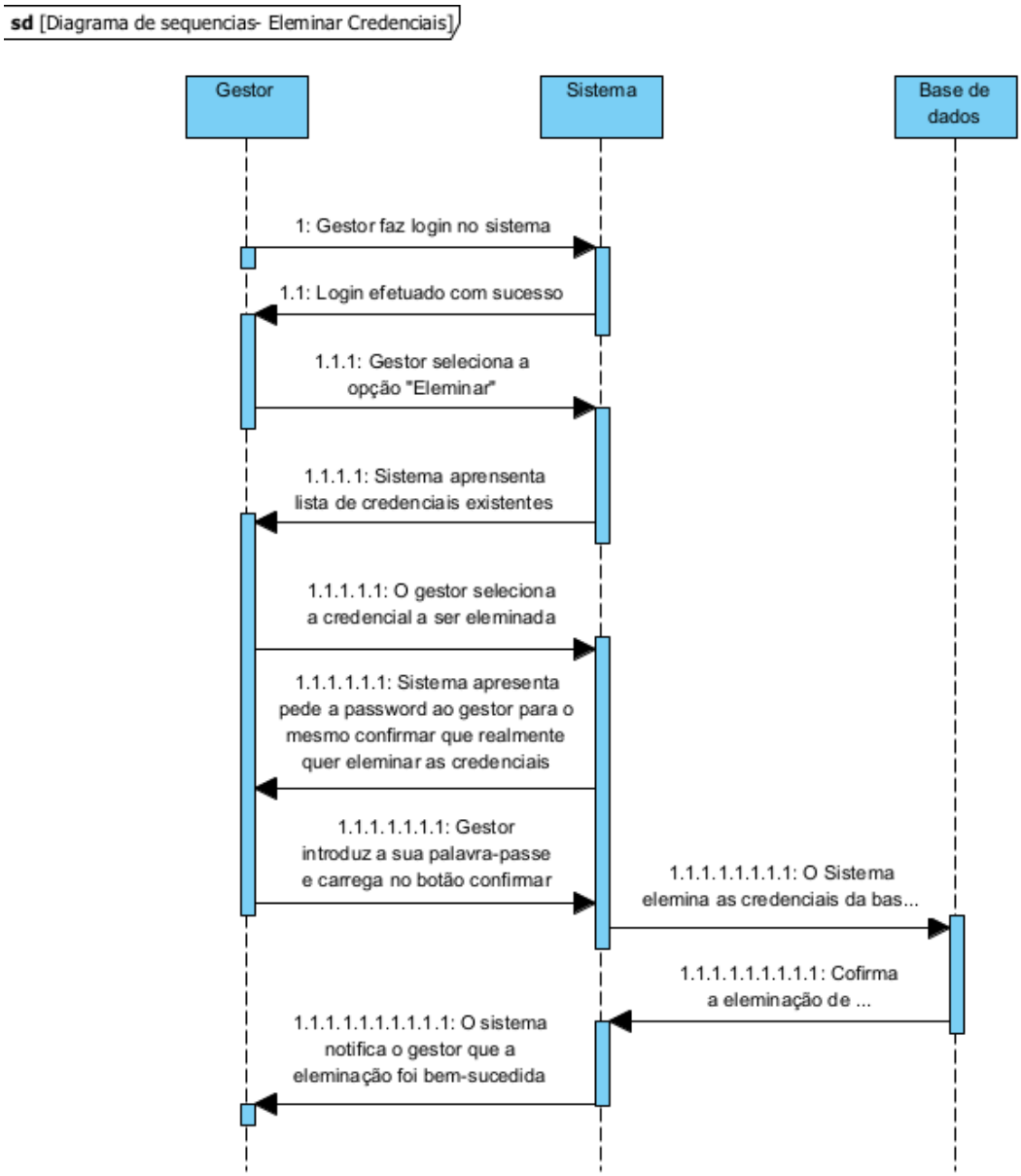
- **Funcionários administrativos** realizam a gestão de consultas, incluindo marcação, desmarcação e consulta de horários.
- **Médicos** acedam e atualizem registos clínicos dos pacientes e façam prescrições.
- **Gestor**, responsável pela criação e remoção de credenciais de acesso ao sistema, garantindo que os diferentes usuários disponham das permissões apropriadas e que o acesso seja controlado de acordo com o papel de cada utilizador.

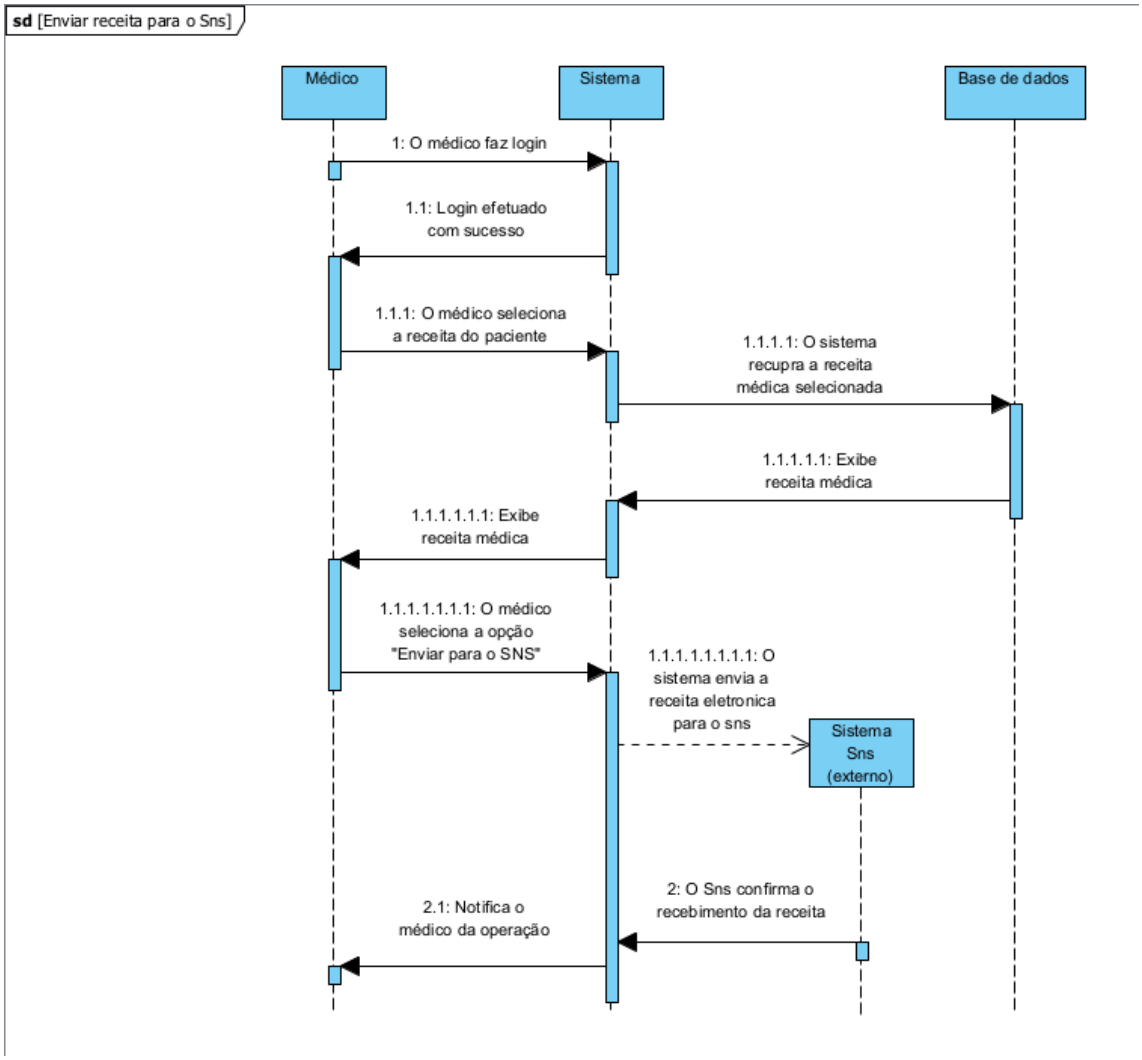
2 Modelo dinâmico

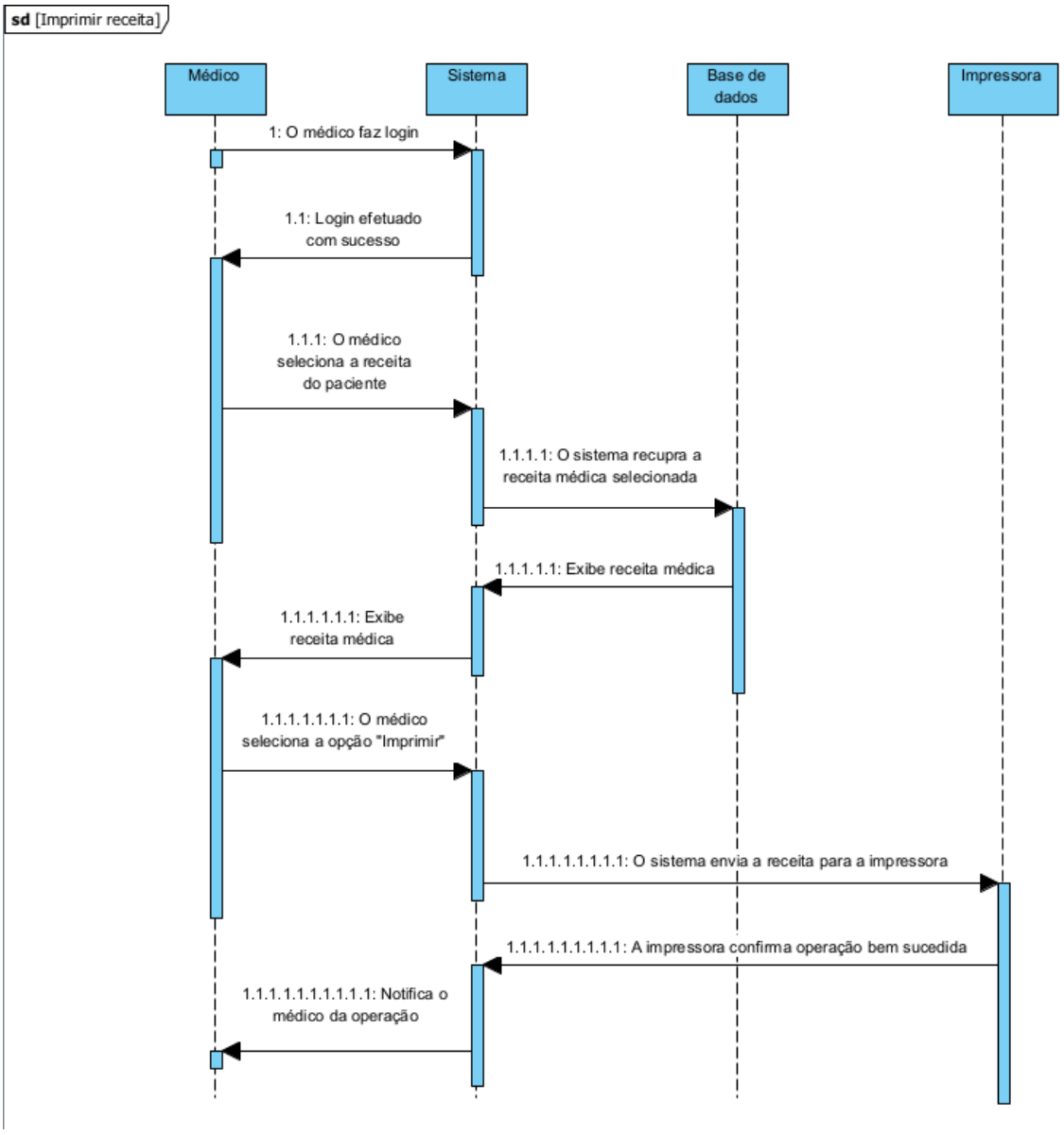
2.1 Diagramas de Sequências

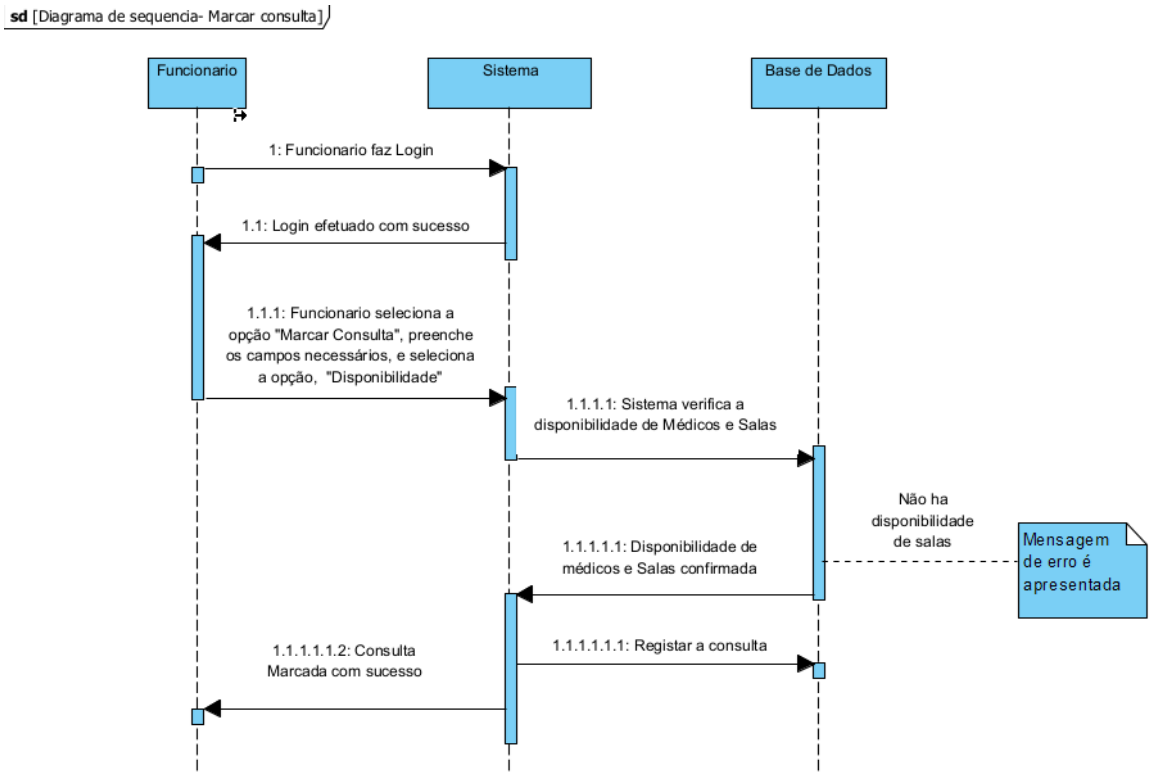


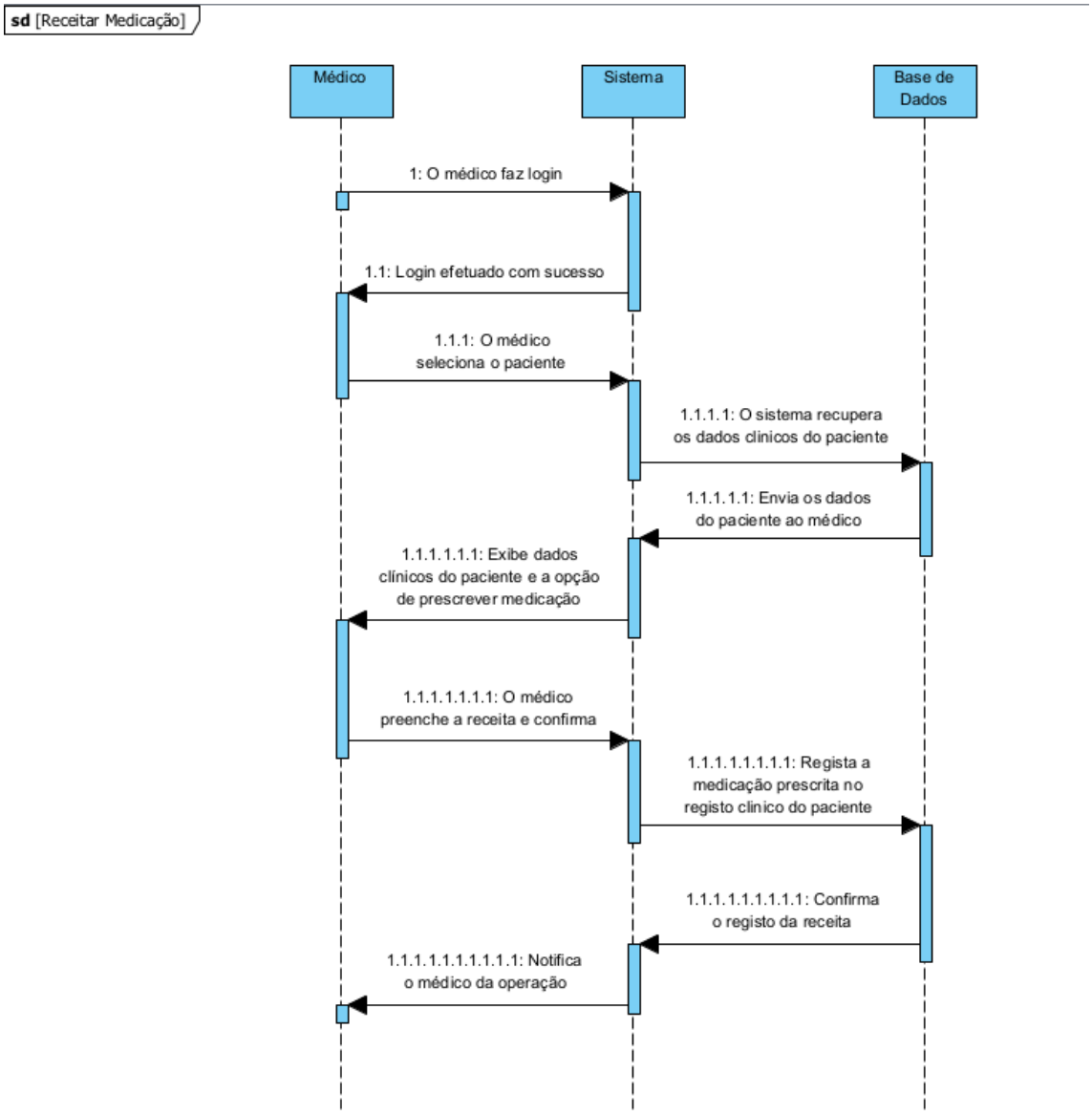

sd [Diagrama de sequencia- Registar um novo utilizador]
Gestor
Sistema
Base de dados
1: Gestor faz login no sistema
1.1: Login efetuado com sucesso
1.1.1: O gestor seleciona a opção "Criar"
1.1.1.1: Apresenta formulario para inserir novas credenciais
1.1.1.1.1: O gestor preenche e envia as novas credenciais, selecionando se é um médico, gestor ou secretário
1.1.1.1.1.1: O sistema regista e guarda as novas credenciais
1.1.1.1.1.1.1: Confirma o registo das credenciais
1.1.1.1.1.1.1.1: O sistema notifica o gestor que o registo foi efetuado com sucesso

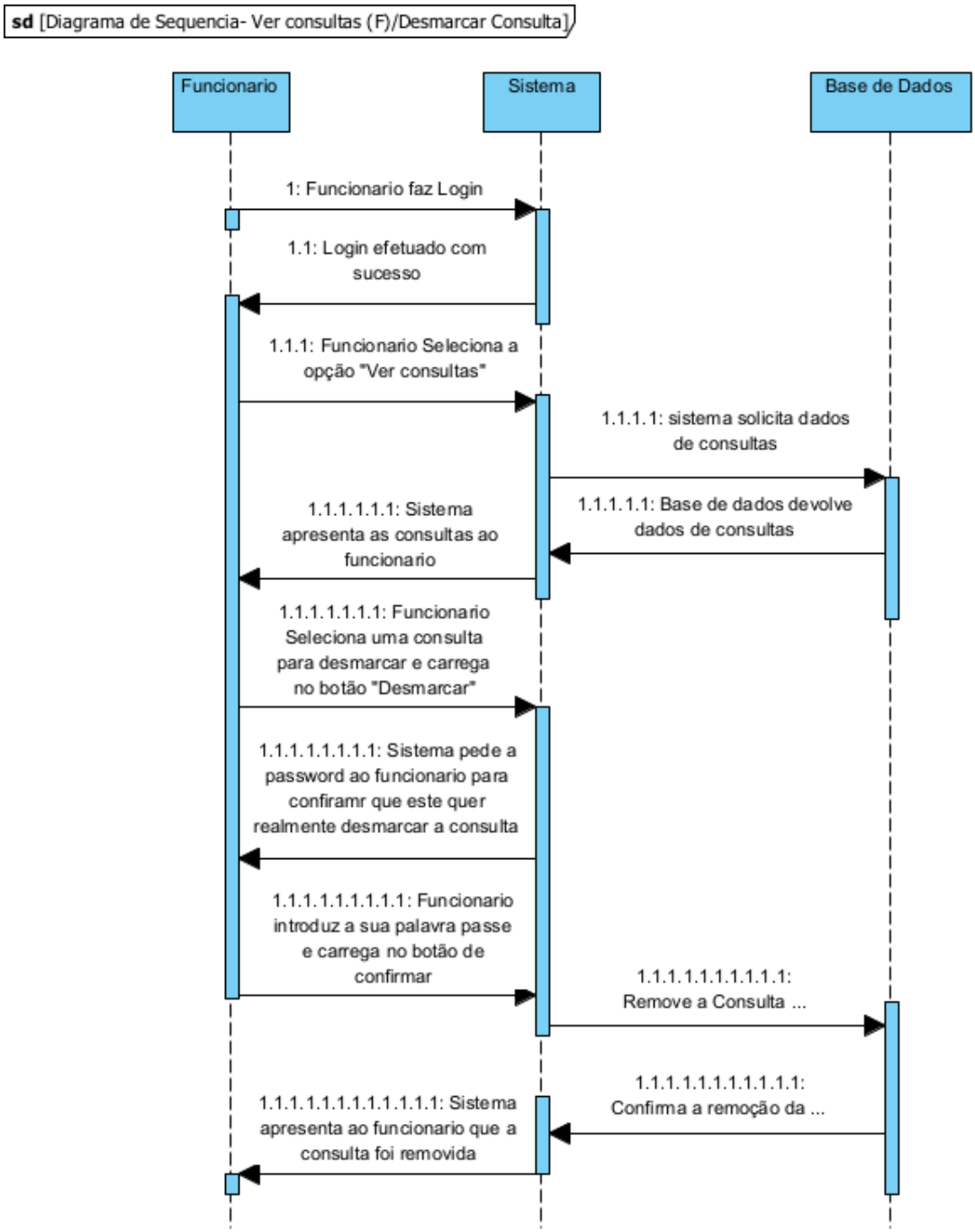

sd [Diagrama de sequencias- Eleminar Credenciais]
Gestor
Sistema
Base de dados
1: Gestor faz login no sistema
1.1: Login efetuado com sucesso
1.1.1: Gestor seleciona a opção "Eleminar"
1.1.1.1: Sistema aprensenta lista de credenciais existentes
1.1.1.1.1: O gestor seleciona a credencial a ser eleminada
1.1.1.1.1.1: Sistema apresenta pede a password ao gestor para o mesmo confirmar que realmente quer eleminar as credenciais
1.1.1.1.1.1.1: Gestor introduz a sua palavra-passe e carrega no botão confirmar
1.1.1.1.1.1.1.1: O Sistema elemina as credenciais da bas...
1.1.1.1.1.1.1.1.1: Cofirma a eleminação de ...
1.1.1.1.1.1.1.1.1.1: O sistema notifica o gestor que a eleminação foi bem-sucedida

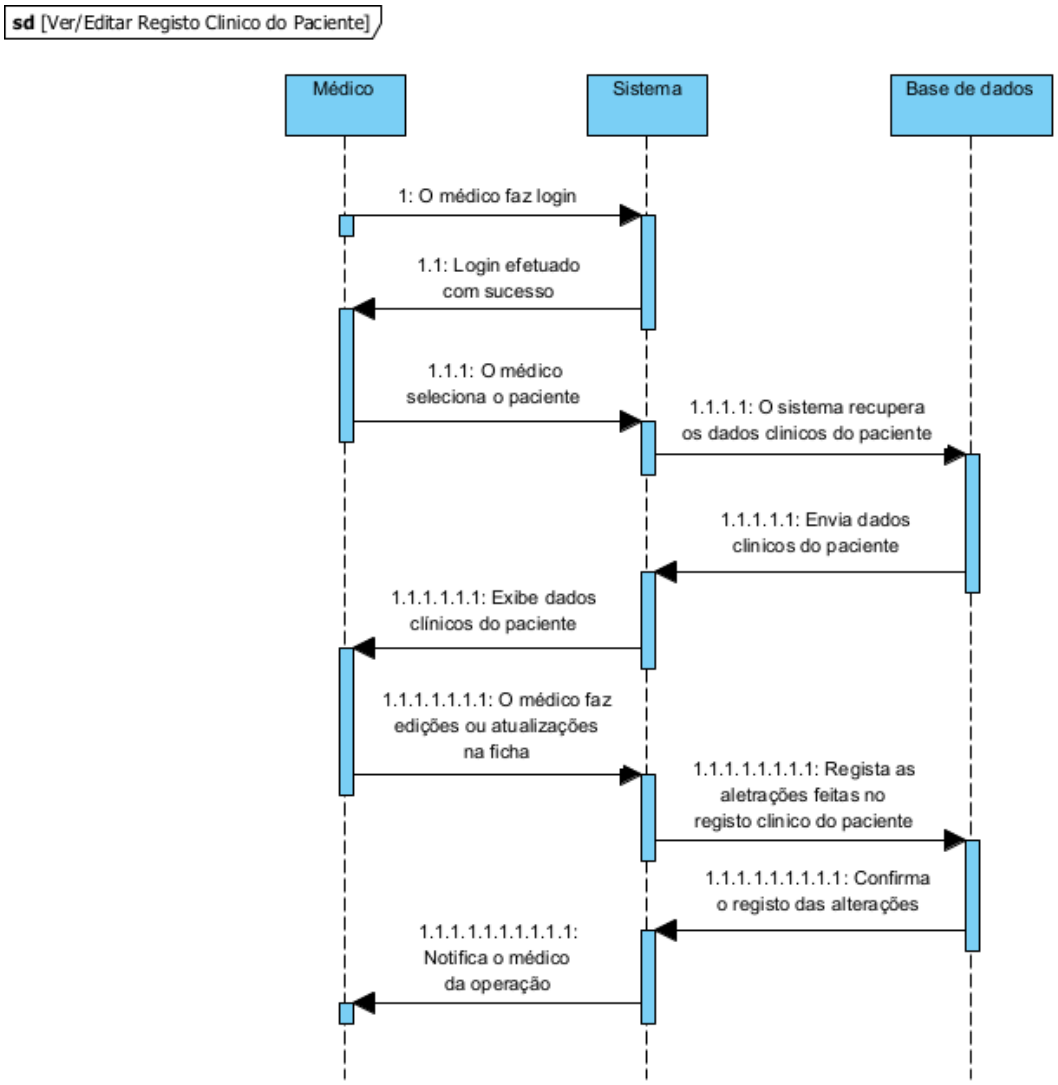

sd [Enviar receita para o Sns]
Médico
Sistema
Base de dados
1: O médico faz login
1.1: Login efetuado com sucesso
1.1.1: O médico seleciona a receita do paciente
1.1.1.1: O sistema recupra a receita médica selecionada
1.1.1.1.1: Exibe receita médica
1.1.1.1.1.1: Exibe receita médica
1.1.1.1.1.1.1: O médico seleciona a opção "Enviar para o SNS"
1.1.1.1.1.1.1.1: O sistema envia a receita eletronica para o sns
Sistema Sns (externo)
2: O Sns confirma o recebimento da receita
2.1: Notifica o médico da operação


sd [Imprimir receita]
Médico
Sistema
Base de dados
Impressora
1: O médico faz login
1.1: Login efetuado com sucesso
1.1.1: O médico seleciona a receita do paciente
1.1.1.1: O sistema recupra a receita médica selecionada
1.1.1.1.1: Exibe receita médica
1.1.1.1.1.1: Exibe receita médica
1.1.1.1.1.1.1: O médico seleciona a opção "Imprimir"
1.1.1.1.1.1.1.1: O sistema envia a receita para a impressora
1.1.1.1.1.1.1.1.1: A impressora confirma operação bem sucedida
1.1.1.1.1.1.1.1.1.1: Notifica o médico da operação


sd [Diagrama de sequencia- Marcar consulta]
Funcionario
Sistema
Base de Dados
1: Funcionario faz Login
1.1: Login efetuado com sucesso
1.1.1: Funcionario seleciona a opção "Marcar Consulta", preenche os campos necessários, e seleciona a opção, "Disponibilidade"
1.1.1.1: Sistema verifica a disponibilidade de Médicos e Salas
Não ha disponibilidade de salas
Mensagem de erro é apresentada
1.1.1.1.1: Disponibilidade de médicos e Salas confirmada
1.1.1.1.1.1: Registar a consulta
1.1.1.1.1.2: Consulta Marcada com sucesso


sd [Receitar Medicação]
Médico
Sistema
Base de Dados
1: O médico faz login
1.1: Login efetuado com sucesso
1.1.1: O médico seleciona o paciente
1.1.1.1: O sistema recupera os dados clinicos do paciente
1.1.1.1.1: Envia os dados do paciente ao médico
1.1.1.1.1.1: Exibe dados clínicos do paciente e a opção de prescrever medicação
1.1.1.1.1.1.1: O médico preenche a receita e confirma
1.1.1.1.1.1.1.1: Regista a medicação prescrita no registo clinico do paciente
1.1.1.1.1.1.1.1.1: Confirma o registo da receita
1.1.1.1.1.1.1.1.1.1: Notifica o médico da operação


sd [Diagrama de Sequencia- Ver consultas (F)/Desmarcar Consulta]
Funcionario
Sistema
Base de Dados
1: Funcionario faz Login
1.1: Login efetuado com sucesso
1.1.1: Funcionario Seleciona a opção "Ver consultas"
1.1.1.1: sistema solicita dados de consultas
1.1.1.1.1: Base de dados devolve dados de consultas
1.1.1.1.1.1: Sistema apresenta as consultas ao funcionario
1.1.1.1.1.1.1: Funcionario Seleciona uma consulta para desmarcar e carrega no botão "Desmarcar"
1.1.1.1.1.1.1.1: Sistema pede a password ao funcionario para confiramr que este quer realmente desmarcar a consulta
1.1.1.1.1.1.1.1.1: Funcionario introduz a sua palavra passe e carrega no botão de confirmar
1.1.1.1.1.1.1.1.1.1: Remove a Consulta ...
1.1.1.1.1.1.1.1.1.1.1: Confirma a remoção da ...
1.1.1.1.1.1.1.1.1.1.1.1: Sistema apresenta ao funcionario que a consulta foi removida


sd [Ver/Editar Registo Clinico do Paciente]
Médico
Sistema
Base de dados
1: O médico faz login
1.1: Login efetuado com sucesso
1.1.1: O médico seleciona o paciente
1.1.1.1: O sistema recupera os dados clinicos do paciente
1.1.1.1.1: Envia dados clinicos do paciente
1.1.1.1.1.1: Exibe dados clínicos do paciente
1.1.1.1.1.1.1: O médico faz edições ou atualizações na ficha
1.1.1.1.1.1.1.1: Regista as aletrações feitas no registo clinico do paciente
1.1.1.1.1.1.1.1.1: Confirma o registo das alterações
1.1.1.1.1.1.1.1.1.1: Notifica o médico da operação

3 Modelo estrutural

3.1 Organização da solução

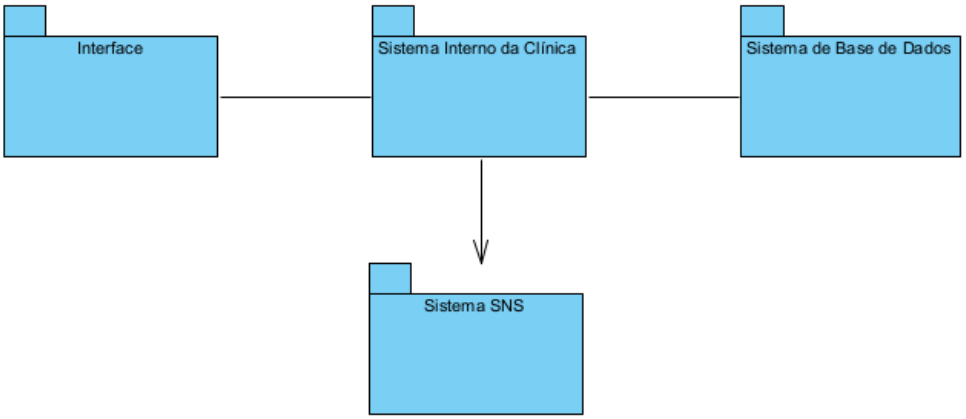


Figura 9 - Diagrama de Packages

3.2 Classes de software

3.2.1 Package medi.flow

O package medi.flow contém diversas classes que representam os componentes principais do nosso sistema. Essas classes estão organizadas de forma a gerenciar diferentes funcionalidades e entidades da aplicação, como clínicas, consultas, médicos, pacientes, utilizadores e textos. Cada classe dentro do package desempenha um papel fundamental na execução e na interação dos usuários com o sistema, abordando a lógica e os dados essenciais para a operação eficiente da aplicação.

3.2.1.1 Clinica

Atributos:

consultas (List<Consulta>) - Lista de todas as consultas agendadas na clínica. Usada para armazenar e consultar as consultas realizadas ou futuras.

medicos (List<String[]>) - Lista de médicos, onde cada elemento contém dados como o ID, nome e especialidade do médico.

horariosMedicos (List<Medico.HorarioMedico>) - Lista de horários de atendimento dos médicos, associando um médico a seus horários disponíveis.

utilizadores (List<Utilizador>) - Lista estática de utilizadores da clínica, incluindo médicos, administradores e outros tipos de utilizadores.

pacientes (List<Paciente>) - Lista de pacientes registrados na clínica, armazenando dados como nome, contacto e número SNS.

registros (List<RegistroClinico>) - Lista de registros clínicos de pacientes, contendo histórico médico, alergias, operações e tratamentos realizados.

Métodos:

Clinica() - Construtor que inicializa as listas de consultas, médicos, utilizadores, pacientes, horários e registros clínicos. Chama métodos auxiliares para obter todos os dados relevantes do sistema.

getConsultas() - Retorna a lista de consultas da clínica.

getMedicos() - Retorna a lista de médicos da clínica.

getUtilizador() - Retorna a lista de utilizadores da clínica.

getPacientes() - Retorna a lista de pacientes da clínica.

getHorarioMedico(int id) - Retorna o horário de um médico específico baseado no seu ID. Caso não exista, retorna null.

getRegistros() - Retorna a lista de registros clínicos de pacientes.

obterNomeMedicoPorId(int id) - Retorna o nome do médico baseado no seu ID, ou null se o médico não for encontrado.

obterPacientePorSns(int sns) - Retorna os dados de um paciente (nome e contacto) a partir do número SNS.

ultimoIdRegistro() - Retorna o maior ID de ficha (registro clínico) existente, utilizado para atribuir novos IDs.

addConsulta(Consulta consulta) - Adiciona uma nova consulta à lista de consultas.

addMedico(String[] medico) - Adiciona um novo médico à lista de médicos.

addUtilizador(Utilizador utilizador) - Adiciona um novo utilizador à lista de utilizadores.

addPaciente(Paciente paciente) - Adiciona um novo paciente à lista de pacientes.

addRegisto(RegistoClinico registo) – Adiciona um registo clínico a lista de registos.

removeConsulta(int id) - Remove uma consulta com o ID especificado da lista de consultas.

removeMedico(int id) - Remove o médico com o ID especificado da lista de médicos.

removeUtilizador(int id) - Remove o utilizador com o ID especificado da lista de utilizadores.

removePaciente(int nSns) - Remove o paciente com o número SNS especificado da lista de pacientes.

atualizaClinica()- Atualiza todas as listas da clínica (consultas, médicos, utilizadores, pacientes, horários de médicos e registos) com os dados mais recentes da base de dados.

Classes Internas:

RegistroClinico - Representa um registro clínico de um paciente, contendo histórico de doenças, alergias, operações realizadas e o número SNS do paciente. Inclui construtores para inicializar os dados do registro clínico.

EntradaRegistroClinico - Representa uma entrada específica dentro de um registro clínico, associada a uma consulta realizada, o médico responsável, data, e tratamentos prescritos.

Receita - Representa uma receita médica, contendo informações sobre o número da receita, data de emissão e a lista de medicamentos prescritos.

3.2.1.2 Consulta

Atributos:

idConsulta (int) - Identificador único da consulta. Utilizado para referenciar a consulta em diversas operações, como edição ou remoção.

data (String) - Data da consulta, representada como uma string. Usada para agendar e exibir informações sobre quando a consulta ocorrerá.

hora (String) - Hora da consulta, representada como uma string. Usada para definir o horário da consulta.

motivo (String) - Motivo da consulta, informando o motivo principal pelo qual o paciente busca atendimento médico.

nomePaciente (String) - Nome do paciente associado à consulta. Usado para exibir informações pessoais do paciente.

nomeMedico (String) - Nome do médico que atenderá o paciente durante a consulta.

snsPaciente (int) - Número SNS do paciente, usado para identificar o paciente no sistema de saúde.

numSala (int) - Número da sala onde a consulta será realizada, usado para direcionar o paciente e o médico para o local adequado.

idMedico (int) - Identificador único do médico associado à consulta. Usado para associar a consulta ao profissional de saúde que realizará o atendimento.

contacto (int) - Número de contato do paciente, utilizado para comunicações com o paciente.

Métodos:

Consulta(int idConsulta, String data, String hora, String motivo, String nomePaciente, String nomeMedico, int snsPaciente, int numSala, int idMedico, int contacto) - Construtor que inicializa todos os atributos da classe, permitindo a criação de um objeto Consulta com todos os dados necessários.

getIdConsulta() - Retorna o identificador único da consulta.

getData() - Retorna a data da consulta.

getHora() - Retorna a hora da consulta.

getMotivo() - Retorna o motivo da consulta.

getNomePaciente() - Retorna o nome do paciente associado à consulta.

getNomeMedico() - Retorna o nome do médico responsável pela consulta.

getSnsPaciente() - Retorna o número SNS do paciente.

getNumSala() - Retorna o número da sala onde a consulta será realizada.

getIdMedico() - Retorna o identificador único do médico responsável pela consulta.

getContacto() - Retorna o número de contato do paciente.

3.2.1.3 Main

Atributos:

clinica (Clinica) - Instância estática da classe Clinica que representa a clínica do sistema. Utilizada em diversas partes do programa para acessar informações relacionadas às consultas, pacientes, médicos e registros.

Métodos:

getClinica() - Método estático que retorna a instância da classe Clinica. Facilita o acesso à clínica de qualquer parte do programa.

main(String[] args) - Método principal que é executado ao iniciar o programa. Ele realiza as seguintes operações:

- Estabelece a conexão com a base de dados, utilizando a classe SqlGeral.DatabaseConnection para criar uma instância de conexão.
- Verifica se a conexão foi bem-sucedida. Caso positivo, imprime uma mensagem confirmando a inicialização da conexão, caso contrário, exibe uma mensagem de falha.
- Inicializa a interface gráfica, criando e exibindo a janela de login através da classe VistaDeLogin. A interface é carregada em um thread separado utilizando java.awt.EventQueue.invokeLater.

3.2.1.4 Medico

Atributos:

numOrdem (int) - Número da Ordem dos Médicos, utilizado para identificar unicamente o médico na Ordem.

especialidade (String) - Especialidade médica do médico, como "Cardiologista", "Pediatra", etc.

horario (List<HorarioMedico>) - Lista que armazena os horários do médico, representados pela classe interna HorarioMedico.

idMedico(int) – Identificador único do médico.

Métodos:

Medico(int id, int cc, String nome, String password, String tipoUtilizador, int numOrdem, String especialidade) - Construtor que inicializa os atributos do médico e os atributos herdados da classe Utilizador. A classe Utilizador é chamada com o método super, passando os parâmetros comuns aos utilizadores do sistema, enquanto os atributos específicos do médico (numOrdem e especialidade) são inicializados diretamente.

getNumOrdem() - Retorna o número da Ordem dos Médicos associado ao médico.

getEspecialidade() - Retorna a especialidade médica do médico.

Classe Interna: HorarioMedico

Atributos:

idMedico (int) - Identificador único do médico associado a esse horário.

horarios (List<String[]>) - Lista de horários, onde cada elemento representa uma entrada com informações de dias e horários.

Métodos:

HorarioMedico(int idMedico, List<String[]> horarios) - Construtor que inicializa os atributos da classe HorarioMedico, recebendo o identificador do médico e a lista de horários.

getIdMedico() - Retorna o identificador do médico associado ao horário.

getHorarios() - Retorna a lista de horários do médico.

3.2.1.5 Paciente

Atributos:

numeroSNS (int) - Número de identificação do paciente no sistema de saúde, utilizado para associar o paciente ao seu histórico médico e a consultas.

nome (String) - Nome completo do paciente, utilizado para identificação.

contacto (int) - Número de contacto do paciente, utilizado para estabelecer comunicação com o paciente, como agendamento de consultas ou envio de notificações.

Métodos:

Paciente(int numeroSNS, String nome, int contacto) - Construtor que inicializa os atributos numeroSNS, nome e contacto com os valores fornecidos na criação do objeto Paciente.

getNumeroSNS() - Retorna o número de identificação do paciente no sistema de saúde.

getNome() - Retorna o nome completo do paciente.

getContacto() - Retorna o número de contacto do paciente.

3.2.1.6 Text

Métodos:

splitStringToList (String string)- Divide uma string para numa lista de string usando letras maiúsculas como ponto de divisão.

listToString(List <String> lista) – Concatena uma lista de strings numa única string, separando-as por espaços.

dataJavaParaSql(String data, String hora) - Converte uma data e hora em formato String (dd/MM/yyyy HH:mm) para um objeto Date no formato utilizado em banco de dados SQL (yyyy-MM-dd HH:mm:ss). Isso é útil para interações com o banco de dados, onde esse formato é requerido.

dataSqlParaJava(String dateString) - Converte uma data no formato SQL (yyyy-MM-dd) para um formato de exibição amigável (dd/MM/yyyy). Esse método é utilizado quando se precisa exibir datas provenientes de um banco de dados em um formato compreensível para o usuário.

dataFormat(String data) - Formata uma data do tipo dd/MM/yyyy para dd/MM, útil quando se deseja exibir apenas o dia e o mês, sem a necessidade de mostrar o ano.

timeFormat(String time) - Converte uma hora no formato HH:mm:ss para um intervalo de uma hora, representado como HH - HH. Por exemplo, ao passar a hora 12:00:00, o método retorna o intervalo 12h - 13h. Esse formato simplifica a exibição de horários em um sistema.

nomeMedicoTransform(String nome) - Formata o nome de um médico, adicionando o título "Dr." antes do nome. Caso o nome tenha apenas dois componentes, o nome completo é retornado com o prefixo "Dr.". Caso contrário, o nome é apresentado com o primeiro nome e o último sobrenome, com o título "Dr.".

3.2.1.7 Utilizador

Atributos:

id (int) - Identificador único do utilizador no sistema. Utilizado para distinguir entre diferentes utilizadores.

cc (int) - Número de Cartão de Cidadão do utilizador. Serve como identificação adicional e única no sistema.

nome (String) - Nome completo do utilizador. Usado para identificação do utilizador nas interações com o sistema.

password (String) - Palavra-passe do utilizador. Utilizada para autenticação e segurança ao acessar o sistema.

tipoUtilizador (String) - Define o tipo de utilizador (exemplo: "admin", "médico", "paciente"). Utilizado para determinar as permissões e funcionalidades acessíveis ao utilizador dentro do sistema.

Métodos:

Utilizador(int id, int cc, String nome, String password, String tipoUtilizador) - Construtor que inicializa todos os atributos da classe com os valores fornecidos. Usado para criar um novo utilizador no sistema.

ultimoIdUtilizador()- Obtém o maior ID atribuido na lista de utilizadores.

getId() - Retorna o identificador único do utilizador no sistema.

getCc() - Retorna o número do Cartão de Cidadão do utilizador.

getNome() - Retorna o nome completo do utilizador.

getPassword() - Retorna a palavra-passe do utilizador.

getTipoUtilizador() - Retorna o tipo de utilizador (exemplo: "admin", "médico", "paciente").

3.2.1.8 Receita

Atributos:

numeroReceita(int) – Identificador único da receita.

dataEmissao(String) – Data em que a receita foi emitida.

medicacaoPrescrita (List <String>) – Lista contendo os medicamentos prescristos.

Métodos:

Receita(int numeroReceita, String dataEmissao, List <String> medicacaoPrescrita)- Inicializa os atributos da classe.

getNumeroReceita() - Retorna o número da receita.

getDataEmissao() - Retorna a data de emição da receita.

GetMedicacaoPrescrita() – Retorna a medicação prescrita.

3.2.1.9 Registo Clínico

Atributos:

numeroSns (int) - Número de Segurança Social do paciente, que serve como identificador principal.

historicoDoencas (List<String>) - Lista de doenças do paciente.

alergias (List<String>) - Lista de alergias.

operacoes (List<String>) - Lista de operações realizadas.

entradasRegistoClinico (List<EntradaRegistoClinico>) - Lista detalhada das entradas no registro clínico.

Métodos:

RegistoClinico(int numeroSns) - Construtor da classe registo clínico que inicializa os atributos com valores normais.

getHistoricoDoencas() – Retorna a lista de histórico de doenças.

getAlergias() – Retorna a lista de alergias.

getOperacoes() – Retorna a lista de operações realizadas no paciente.

getNumeroSns() – Retorna o número do SNS do paciente.

getEntradasRegistoClinico() – Retorna a lista de entradas do registo clínico do paciente.

setHistoricoDoencas(List<String> historicoDoencas) – Atualiza a lista de histórico de doenças.

setAlergias(List<String> alergias) – Atualiza a lista de alergias.

setOperacoes(List<String> operacoes) – Atualiza a lista de operações realizadas no paciente.

setEntradasRegistosClinicos(List<EntradaRegistoClinico> entradasRegistosClinicos) – Atualiza a lista de entradas no registo clínico.

addEntradaRegistoClinico(EntradaRegistoClinico entrada) – Adiciona uma entrada ao registo clínico.

listToString(List<String> lista) – Converte a lista de strings em uma string única.

getInfoPaciente() – Retorna um array com o nome e o contacto do paciente correspondente ao número SNS.

Classe interna EntradaRegistoClinico

Atributos:

- **id_medico(int)** – Identificador único do médico que fez a entrada no registo clínico.
- **data(String) –** Data da entrada no registo clínico.
- **assunto(List<String>) –** Lista de assuntos tratados na entrada.
- **tratamento(List<String>) –** Lista de tratamentos prescritos ou realizados.

Métodos:

- **EntradaRegistoClinico(int id_medico, String data, List<String> assunto, List<String> tratamento) –** Inicializa os atributos da entrada de registo clínico.
- **getIdMedico() –** Retorna o identificador do médico.
- **getData() –** Retorna a data da entrada.
- **getAssunto() –** Retorna a lista de assuntos tratados.
- **getTratamentos() –** Retorna a lista de tratamentos realizados.
- **getNumeroSns()** – Retorna o número do SNS do paciente associado à entrada.

3.2.1.9 Registo Clínico

Atributos:

numSala(int) – Número identificador da sala.

idMedico(int) – Identificador único do médico responsável pela sala (se houver).

ocupada(boolean) – Indica se a sala está ocupada (verdadeiro) ou livre (falso).

tipoSala(String) – Tipo de sala (ex: consulta, cirurgia, emergência, etc.).

Métodos:

Sala(int numSala, int idMedico, boolean ocupada, String tipoSala) – Construtor que inicializa os atributos da sala. Define se a sala está ocupada com base no idMedico (se o idMedico for diferente de -1, a sala é marcada como ocupada).

getNumSala() – Retorna o número da sala.

getIdMedico() – Retorna o identificador do médico responsável pela sala.

isOcupada() – Retorna um valor booleano indicando se a sala está ocupada.

getTipoSala() – Retorna o tipo de sala.

setNumSala(int numSala) – Define o número da sala.

setIdMedico(int idMedico) – Define o identificador do médico responsável pela sala.

setOcupada(boolean ocupada) – Define se a sala está ocupada.

setTipoSala(String tipoSala) – Define o tipo da sala

3.2.2 Package sql.server

O package sql.server contém diversas classes responsáveis pela interação com a camada de persistência do sistema. Os ficheiros estão organizados para implementar a lógica de acesso aos dados, incluindo métodos para manipular informações relacionadas a palavras-passes, operações gerais na base de dados, e funcionalidades específicas para gestores, médicos e secretários.

3.2.2.1 SqlGeral

Atributos:

URL (String) - Representa o endereço do servidor da base de dados. Utilizado para estabelecer a conexão com a base de dados.

USER (String) - Nome de utilizador necessário para autenticar na base de dados.

PASSWORD (String) - Palavra-passe associada ao utilizador que realiza a autenticação no base de dados.

connection (Connection) - Objeto que representa a conexão com a base de dados. É utilizado em todas as operações de consulta e manipulação de dados.

Métodos:

SqlGeral.verificarLogin(String idUtilizador, String password) - Este método valida as credenciais do utilizador. Ele recebe o identificador do utilizador e a palavra-passe, cifra a senha fornecida e compara-a com os dados armazenados na base de dados. Para isso, o método estabelece uma conexão com a base e utiliza o procedimento armazenado VerificarLogin, passando o ID do utilizador e a palavra-passe cifrada como parâmetros. Caso as credenciais estejam corretas, o método retorna o tipo de utilizador associado. Caso contrário, retorna null. Em caso de erros durante a execução ou problemas de conexão, exibe mensagens de erro na consola.

SqlGeral.obterTodasConsultas() - Este método procura todas as consultas futuras armazenadas no sistema. Ele conecta-se à base de dados e utiliza o procedimento armazenado SelecionarConsultas para procurar os dados. Após obter os resultados, o método filtra as consultas com base na data atual, ignorando aquelas que já ocorreram. Para cada consulta futura, cria um objeto da classe Consulta e adiciona-o a uma lista. Por fim, retorna essa lista com os dados relevantes das consultas, incluindo informações como ID, data, hora, paciente, médico, entre outros.

DatabaseConnection.getInstance() - Este método assegura que apenas uma instância da conexão com a base de dados seja criada e mantida durante a execução da aplicação. Ele verifica se a conexão atual está nula ou encerrada e, em caso positivo, cria uma nova conexão utilizando os atributos URL, USER e PASSWORD. Em seguida, retorna o objeto da conexão ativa. Caso ocorram erros durante a criação da conexão, exibe uma mensagem de erro no console.

DatabaseConnection.closeConnection() - Este método encerra a conexão ativa com a base de dados, libertando os recursos utilizados. Ele verifica se existe uma conexão ativa e tenta encerrá-la. Em caso de sucesso, exibe uma mensagem informando que a conexão foi encerrada. Caso ocorram erros ao encerrar a conexão, trata as exceções e exibe mensagens de erro no console.

3.2.2.2 SqlGestor

Métodos:

obterTodosUtilizadores() - Este método retorna uma lista com todos os utilizadores armazenados na base de dados. Para cada utilizador encontrado, é criado um objeto Utilizador ou Medico (caso o tipo seja "Medico"). A lista é preenchida e retornada com todos os utilizadores. Ele estabelece uma conexão com a base de dados. Executa uma consulta para obter todos os utilizadores. Para os médicos, faz uma segunda consulta para obter detalhes adicionais como o número de ordem e a especialidade.

criarUtilizador(String nome, String password, String tipoUtilizador, int cc, String especialidade, int numOrdem) - Este método cria um novo utilizador. Ele recebe os dados do utilizador, incluindo a especialidade e o número de ordem caso seja um médico. A password é cifrada antes de ser inserida na base de dados. Executa um procedimento armazenado para criar o utilizador na base de dados. Se o tipo for "Médico", executa um outro procedimento para inserir os dados específicos do médico.

eliminarUtilizador(int idUtilizador) - Este método elimina um utilizador da base de dados com base no ID. Ele executa uma consulta SQL para excluir o utilizador da tabela Utilizador. Executa uma consulta DELETE para remover o utilizador da base de dados. Retorna true se o utilizador foi removido com sucesso, ou false caso contrário.

obterSalas() – Obtém todos os números de salas cadastradas na base de dados. Retorna uma lista com os números das salas.

3.2.2.3 SqlMedico

Métodos:

obterTodosRegistros() - Obtém todos os registros clínicos cadastrados na base de dados. Para cada registo, converte os históricos de doenças, alergias e operações de uma string para uma lista e os adiciona ao objeto RegistoClinico. Retorna uma lista de objetos RegistoClinico.

obterEntradasRC(RegistoClinico registoClinico) – Obtém todas as entradas relacionadas a um RegistoClinico específico. Para cada entrada, converte os assuntos e tratamentos de string para listas. Retorna uma lista de objetos EntradaRegistoClinico associados ao RegistoClinico.

criarNovaEntrada(EntradaRegistoClinico entrada) – Cria uma nova entrada no registo clínico, armazenando os dados fornecidos (número de SNS, ID do médico, data, assuntos e tratamentos) na base de dados.

alterarRC(RegistoClinico registoClinico) – Altera os dados de um RegistoClinico existente na base de dados, atualizando o histórico de doenças, alergias e operações com os novos valores.

3.2.2.4 SqlSecretaria

Métodos:

obterTodosMedicos() - Este método retorna uma lista contendo todos os médicos registrados na base de dados. A consulta SQL chama o procedimento armazenado

ObterTodosMedicos, que retorna os IDs e especialidades dos médicos. Cada médico é representado como um array de strings contendo o ID e a especialidade.

obterTodosPacientes() - Este método retorna uma lista de pacientes armazenados na base de dados. A consulta SQL chama o procedimento ObterTodosPacientes, que retorna os dados dos pacientes, como o número de SNS, nome e contacto, que são então usados para criar uma lista de objetos Paciente.

criarConsulta(String data, String hora, String motivo, String nomePaciente, int snsPaciente, int contacto, int numSala, int idMedico, String nomeMedico) - Este método cria uma nova consulta médica. Ele recebe os parâmetros necessários (como a data, hora, motivo, nome do paciente, etc.) e os formata conforme necessário para realizar a chamada do procedimento armazenado MarcarConsulta. O ID da consulta gerado pelo banco de dados é retornado.

desmarcarConsulta(int IDConsulta) - Este método permite desmarcar uma consulta existente, chamando o procedimento armazenado DesmarcarConsulta com o ID da consulta. Ele executa a consulta e remove a consulta do sistema.

criarPaciente(int numero, String nome, int contacto) - Este método cria um novo paciente caso o paciente não exista. Ele chama o procedimento armazenado CriarPaciente, passando os dados do paciente (número, nome, contacto) para o banco de dados. Se o paciente já existir, o método não faz nada.

criarNovoRC(int numeroSns) – Cria um novo registo clínico para um paciente baseado no número de SNS fornecido.

todosHorariosMedicos() - Este método retorna todos os horários ocupados de todos os médicos registrados na base de dados. Ele chama o procedimento ObterTodosMedicos para obter os IDs dos médicos e, para cada médico, chama o método horariosOcupadosMedico() para obter os horários ocupados.

horariosOcupadosMedico(int id) - Este método retorna os horários ocupados de um médico específico. Ele recebe o ID do médico, chama o procedimento ObterHorariosMedico, e para cada horário ocupado retornado, ele formata a data e hora e os armazena em uma lista que é retornada.

3.2.3 Package inter.face

O package inter.face contém diversos componentes que compõem a interface gráfica do nosso sistema. Os ficheiros estão organizados em classes e formulários representando a lógica e os elementos visuais da aplicação.

3.2.3.1 Vista de Login

Métodos:

Construtor VistaDeLogin - Inicializa os componentes gráficos da interface, chamando o método initComponents(). Adiciona **KeyListeners** a todos os componentes da janela, permitindo capturar eventos de teclas pressionadas.

botaoLoginActionPerformed(java.awt.event.ActionEvent evt) - Executado quando o botão de login é clicado. Este método obtém o nome de utilizador e a palavra-passe introduzidos e valida as credenciais através do método SqlGeral.verificarLogin. Dependendo do tipo de utilizador retornado, redireciona para a interface correspondente: VistaGestor para gestores, VistaMedico para médicos (definindo o identificador do médico em idMedicoAUtilizarOSistema) e VistaSecretaria para secretárias. Caso as credenciais sejam inválidas, apresenta uma mensagem de erro.

formKeyPressed(java.awt.event.KeyEvent evt) - Captura eventos de teclas, permitindo que o botão de login seja ativado automaticamente ao pressionar a tecla "Enter".

addKeyListenerToComponents(java.awt.Container container) - Adiciona um **KeyListener** a todos os componentes do **Container** de forma recursiva, garantindo que todos respondem aos eventos de teclado.

main(String[] args) - Configura o estilo gráfico da aplicação para **Nimbus** e exibe a janela de login, chamando o método new VistaDeLogin().setVisible(true).

3.2.3.2 Vista de Gestor

Métodos:

Construtor VistaGestor - Inicializa os componentes gráficos da interface com initComponents() e carrega as credenciais existentes na base de dados através do método carregarCredenciaisBaseDeDados().

carregarCredenciaisBaseDeDados() - Limpa o painel de credenciais, carrega as credenciais dos utilizadores a partir da base de dados, ordena-os pelo tipo de utilizador e ID, e atualiza dinamicamente o painel de credenciais. Também reinicia a posição do scroll para o topo e atualiza graficamente o painel.

criarPainelCredencial(Utilizador utilizador) - Cria e adiciona ao painel principal uma instância do painel Credencial, representando as informações do utilizador fornecido.

botaoCriarCredencialActionPerformed(java.awt.event.ActionEvent evt) - Esconde a barra de pesquisa e o painel de credenciais, exibe o painel de criação de credenciais e carrega novamente as credenciais da base de dados.

botaoEliminarCredencialActionPerformed(java.awt.event.ActionEvent evt) - Exibe o painel para eliminar credenciais, juntamente com a barra de pesquisa e o botão de pesquisa, e chama carregarCredenciaisBaseDeDados() para atualizar as credenciais exibidas.

botaoPesquisaActionPerformed(java.awt.event.ActionEvent evt) - Obtém os utilizadores da base de dados e filtra os resultados com base no texto inserido na barra de pesquisa. Se encontrar correspondências, atualiza dinamicamente o painel com os utilizadores filtrados; caso contrário, exibe uma mensagem indicando que nenhum utilizador foi encontrado.

exitButtonMouseClicked(java.awt.event.MouseEvent evt) - Fecha a janela atual e exibe a interface de login (VistaDeLogin).

tipoFuncionarioActionPerformed(java.awt.event.ActionEvent evt) - Ajusta dinamicamente os campos exibidos com base no tipo de funcionário selecionado (exemplo: exibe campos adicionais para médicos, como especialidade e número da ordem).

concluirButtonActionPerformed(java.awt.event.ActionEvent evt) - Recolhe os dados inseridos no formulário para criar um novo utilizador. Valida os campos obrigatórios, como CC e dados específicos para médicos. Após validar os dados, chama o método para criar o utilizador na base de dados, exibe mensagens de erro ou sucesso conforme o caso, e atualiza o painel de credenciais.

main(String[] args) - Configura o look-and-feel Nimbus e inicializa a interface gráfica, exibindo a janela do gestor (VistaGestor).

3.2.3.3 Credencial

Atributos:

idUtilizador (int) - Identifica unicamente o utilizador ao qual a credencial está associada. Utilizado para operações relacionadas ao utilizador, como eliminação ou edição de informações.

Métodos:

Credencial(Utilizador utilizador) - Construtor que inicializa os componentes da interface e define os valores apresentados com base nos dados do utilizador recebido.

botaoEliminarActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Eliminar". Ele solicita a confirmação da exclusão e a introdução da palavra-passe do gestor através de um diálogo modal. A palavra-passe inserida é encriptada e comparada com a palavra-passe armazenada no sistema. Caso a autenticação seja bem-sucedida, o utilizador é eliminado através da camada de persistência, e o componente correspondente é removido da interface gráfica. Se a autenticação falhar, mensagens de erro adequadas são exibidas ao utilizador.

3.2.3.4 Vista Secretaria

Métodos:

VistaSecretaria() - Construtor da classe. Inicializa os componentes da interface gráfica e carrega as consultas da base de dados.

carregarConsultasBaseDeDados() - Este método limpa o painel de consultas, obtém as consultas da base de dados e cria um painel para cada consulta, exibindo as informações relevantes. Ajusta o tamanho do painel conforme o número de consultas e reposiciona a barra de rolagem para o topo.

criarPainelConsulta(Consulta consulta) - Este método cria um painel individual para cada consulta, ajustando seu tamanho e adicionando-o ao painel principal (consultasPanel).

botaoVerConsultasActionPerformed(java.awt.event.ActionEvent.evt) - Este método é acionado ao clicar no botão "Ver Consultas". Ele exibe o painel de consultas e recarrega os dados das consultas da base de dados.

botaoMarcarConsultasActionPerformed(java.awt.event.ActionEvent.evt) - Este método é acionado ao clicar no botão "Marcar Consultas". Ele exibe o painel para marcar consultas e oculta o painel de consultas.

botaoPesquisaActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Pesquisar". Ele permite a pesquisa de consultas usando o número de SNS ou nome do paciente, filtrando os resultados e exibindo-os no painel. Caso não haja resultados, uma mensagem de erro é exibida.

botaoHorariosActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Horários". Ele verifica se a data da consulta foi preenchida corretamente e, se for válida, exibe a janela de disponibilidade de médicos.

botaoPacientesActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Pacientes". Ele busca os dados do paciente a partir do número de SNS e os exibe nos campos correspondentes. Se o paciente não for encontrado, exibe uma mensagem de erro.

exitButtonMouseClicked(java.awt.event.MouseEvent evt) - Este método é acionado ao clicar no botão de sair. Ele fecha a janela atual e abre a tela de login (VistaDeLogin).

botaoMarcarActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Marcar". Ele valida os dados inseridos para marcar uma nova consulta, como verificar se todos os campos estão preenchidos corretamente, validar o formato do número de SNS e do contacto do paciente, e se a consulta está sendo marcada no futuro. Caso algum dado esteja incorreto, uma mensagem de erro será exibida. Se a consulta for criada com sucesso, os campos são limpos e a consulta é adicionada ao sistema.

3.2.3.5 Consulta Funcionario

Atributos:

idConsulta (int) - Identifica unicamente a consulta associada a um determinado painel de consulta. Este atributo é usado para realizar operações como desmarcar a consulta na base de dados e removê-la da lista de consultas da clínica.

Métodos:

ConsultaFuncionario(Consulta consulta) - Construtor que inicializa os componentes da interface gráfica com os dados da consulta fornecida. Preenche os campos da interface (como nome do paciente, SNS, nome do médico, sala, data e hora da consulta) e verifica se algum valor é nulo, substituindo-o por um valor padrão, como "Não disponível". Além disso, o ID da consulta é atribuído ao atributo idConsulta.

botaoDesmarcarActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado ao clicar no botão "Desmarcar". Ele realiza as seguintes operações: Chama o método desmarcarConsulta(idConsulta) da classe SqlSecretaria para remover a consulta da base de dados depois, remove a consulta da lista de consultas da clínica através de getClinica().removeConsulta(idConsulta)e por fim atualiza o painel pai ao remover o painel de consulta atual, ajustando o tamanho do painel para refletir a remoção e redesenhando o painel.

3.2.3.6 Disponibilidade Medicos

Métodos :

DisponibilidadeMedicos() - Construtor que inicializa os componentes da interface gráfica e carrega os dados dos médicos na interface através do método carregarMedicosBaseDeDados().

carregarMedicosBaseDeDados() - Este método carrega todos os médicos da base de dados e cria um painel para cada médico. Ele também ajusta o tamanho do painel conforme o número de médicos e posiciona o scroll no topo. Após adicionar os componentes, o painel é atualizado visualmente.

criarPainelMedico(String id, String espec) - Método que cria um painel para um médico específico, utilizando o ID e a especialidade fornecidos, e adiciona esse painel ao contêiner medicosPanel.

botaoPesquisaActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado quando o botão de pesquisa é clicado. Ele verifica se a especialidade foi inserida e, caso afirmativo, filtra os médicos da base de dados de acordo com a especialidade fornecida. Caso haja resultados, novos painéis de médicos são criados e adicionados ao painel; se não houver, uma mensagem de erro é exibida.

barraPesquisaActionPerformed(java.awt.event.ActionEvent evt) - Método vazio que pode ser usado para implementar a funcionalidade de pesquisa quando o usuário pressionar "Enter" no campo de pesquisa.

refreshMouseClicked(java.awt.event.MouseEvent evt) - Este método é acionado ao clicar no rótulo refresh. Ele limpa o painel de médicos e recarrega os médicos da base de dados, reiniciando o painel com a lista completa de médicos.

3.2.3.7 MedicoPanel

Métodos:

MedicoPanel(String id, String espec) - Este construtor inicializa os componentes da interface gráfica chamando o método initComponents(). Ele preenche os campos de ID e

especialidade do médico na interface, utilizando os parâmetros id e espec passados ao instanciar o painel.

botaoPesquisaActionPerformed(java.awt.event.ActionEvent evt) - Este método é acionado quando o botão de pesquisa é clicado. Ele verifica se a especialidade foi inserida e, caso afirmativo, filtra os médicos da base de dados de acordo com a especialidade fornecida. Caso haja resultados, novos painéis de médicos são criados e adicionados ao painel; se não houver, uma mensagem de erro é exibida.

3.2.3.8 Horarios Medico

Atributos:

idMedico (int) - Identificador único do médico cujos horários estão sendo gerenciados.

todosDias (String[]) - Array que contém os cinco dias que são usados para mostrar os horários do médico, incluindo dois dias antes e dois dias depois da data de consulta.

horarios (List<String[]>) - Lista contendo os horários do médico.

horariosFiltered (List<String[]>) - Lista filtrada dos horários do médico, com base no ano da consulta.

Método:

HorariosMedico(int id) throws ParseException - Construtor que inicializa os atributos idMedico, horarios, e horariosFiltered. Além disso, chama os métodos para inicializar os componentes da interface e carregar os dados de horários.

carregarTexto() throws ParseException - Método que define o nome do médico na interface gráfica e atualiza os textos dos dias (antes e depois da consulta), exibindo as informações do período de horários.

diasAntesEDepois(String data) throws ParseException - Método que calcula os cinco dias relevantes (dois antes, o dia da consulta, e dois dias depois) com base na data de consulta fornecida. Ele usa o formato dd/MM/yyyy para manipulação da data.

carregarHorarios() - Método responsável por preencher os horários dos médicos nos dias calculados, organizando-os em listas e atualizando a interface gráfica. Ele também valida que o atributo todosDias foi inicializado corretamente.

updateHorarios() - Método que define o renderizador personalizado (CustomCellRenderer) para cada lista de horários, permitindo diferenciar horários ocupados de horários disponíveis visualmente.

CustomCellRenderer (Classe interna) - Classe personalizada que renderiza as células das listas de horários, alterando a cor de fundo para vermelho em caso de horários ocupados e ajustando as cores para quando o item está selecionado.

filtrarPorAno(List<String[]> horarios, int ano) throws ParseException - Método que filtra os horários de um médico com base no ano da data de consulta. Ele ajusta a data para o formato dd/MM/yyyy se o ano não estiver presente e retorna a lista de horários filtrada.

isolarAno(String data) throws ParseException - Método que extrai o ano de uma string de data, verificando o formato da data (dd/MM/yyyy ou dd/MM) e corrigindo se necessário para adicionar o ano atual.

main(String[] args) - Método principal que configura a interface gráfica e exibe a janela de horários do médico, chamando o construtor de HorariosMedico para instanciar o objeto com um determinado ID de médico.

3.2.3.9 Vista Medico

Métodos:

VistaMedico() - Construtor que inicializa os componentes da interface gráfica e carrega as consultas existentes no banco de dados, chamando o método carregarConsultasBaseDeDados().

carregarConsultasBaseDeDados() - Método responsável por carregar as consultas da base de dados. Ele limpa o painel de consultas e, em seguida, obtém todas as consultas da clínica. Para cada consulta, verifica se o médico da consulta corresponde ao médico atual que está utilizando o sistema. Em seguida, cria um painel para cada consulta e adiciona ao painel de consultas.

criarPainelConsulta(Consulta consulta) - Método que cria um painel específico para exibir os dados de uma consulta. Este painel é adicionado ao painel principal de consultas.

exitButtonMouseClicked(java.awt.event.MouseEvent evt) - Método que é acionado quando o botão de saída é clicado. Ele fecha a janela atual e abre a janela de login (VistaDeLogin).

main(String[] args) - Método principal que configura a interface gráfica e exibe a janela VistaMedico, chamando o construtor para instanciar o objeto.

3.2.3.10 Consulta Medico

Atributos:

nSns (int) - Armazena o número do SNS (Sistema Nacional de Saúde) do paciente para facilitar a busca pelo seu histórico médico.

listaConsulta (javax.swing.JList<String>) - Lista que exibe os motivos ou detalhes adicionais sobre a consulta.

Métodos:

ConsultaMedico(Consulta consulta) - Construtor que recebe uma instância de Consulta como parâmetro. Dentro dele atribui o número SNS do paciente, preenche os campos da

interface com as informações da consulta (nome do paciente, data, hora) ou coloca um texto padrão ("Não disponível") caso algum dado não esteja presente e a função quebraPontos(listaConsulta, consulta.getMotivo()) é chamada, provavelmente para quebrar o texto de motivos da consulta em várias linhas ou formatá-lo de maneira adequada na lista.

fichaMedicaButtonActionPerformed(java.awt.event.ActionEvent evt) - Método acionado quando o botão fichaMedicaButton é clicado. Ele obtém uma lista de registros clínicos da clínica. Para cada registro clínico, verifica se o número do SNS corresponde ao do paciente e quando encontra o registro correspondente, cria e exibe a janela de detalhes do histórico médico do paciente (classe RegistoClinico).

3.2.3.11 Registo clinico

Métodos:

RegistoClinico(Clinica.RegistroClinico registroClinico): Construtor da classe, que é chamado para inicializar o registro clínico com base nos dados passados. A primeira coisa que faz é procurar o paciente correspondente ao número de SNS do registro clínico (registroClinico.getNumeroSns()). Depois, preenche os campos da interface gráfica, como:

- nomePaciente: Exibe o nome do paciente.
- nSns: Exibe o número do SNS do paciente.
- contacto: Exibe o número de contato do paciente.

snsButtonActionPerformed(java.awt.event.ActionEvent evt) - Este método é chamado quando o botão "CONCLUIR" é pressionado. Ele valida se todos os itens na lista de medicamentos estão no formato correto (exemplo: "1xParacetamol - 500mg"). Se algum item não estiver no formato correto, uma mensagem de erro é exibida. Caso contrário, o médico pode escolher entre enviar a receita para o SNS ou imprimir a receita. Após a escolha, a janela é fechada.

3.2.3.12 Nova Entrada RC

Métodos:

NovaEntradaRC(Consulta consulta) - O construtor recebe uma consulta como parâmetro e a associa à nova entrada. A interface gráfica é inicializada, e o objeto consulta é armazenado para ser utilizado posteriormente ao criar a nova entrada.

concluirButtonActionPerformed(java.awt.event.ActionEvent evt) - Este método é chamado quando o botão "Concluir" é pressionado. Ele verifica se os campos "assuntos" e "tratamentos" foram preenchidos. Caso algum dos campos esteja vazio, uma mensagem de erro é exibida. Se ambos os campos estiverem preenchidos, a entrada é criada e adicionada ao registro clínico do paciente. Além disso, a entrada é salva na base de dados.

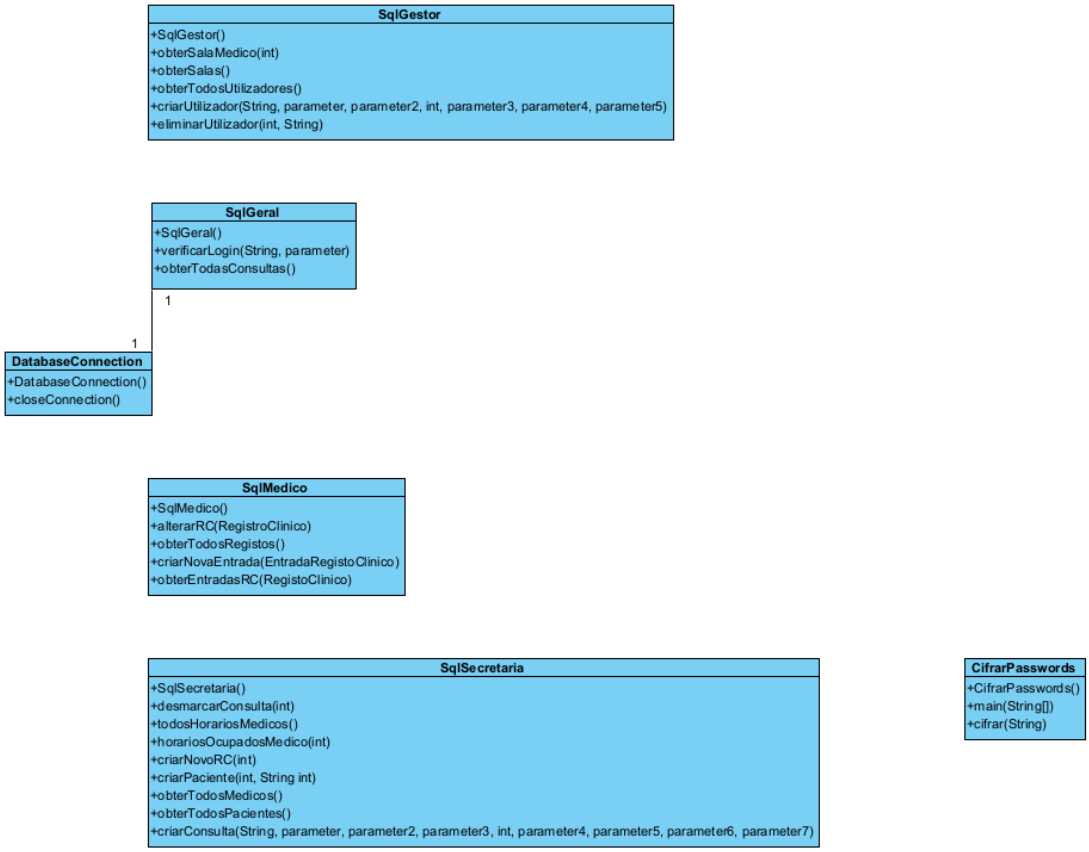
main(String args[]) - O método main é o ponto de entrada para criar e exibir a interface gráfica. Ele chama a criação do formulário e o inicializa para que o médico possa inserir as informações necessárias.

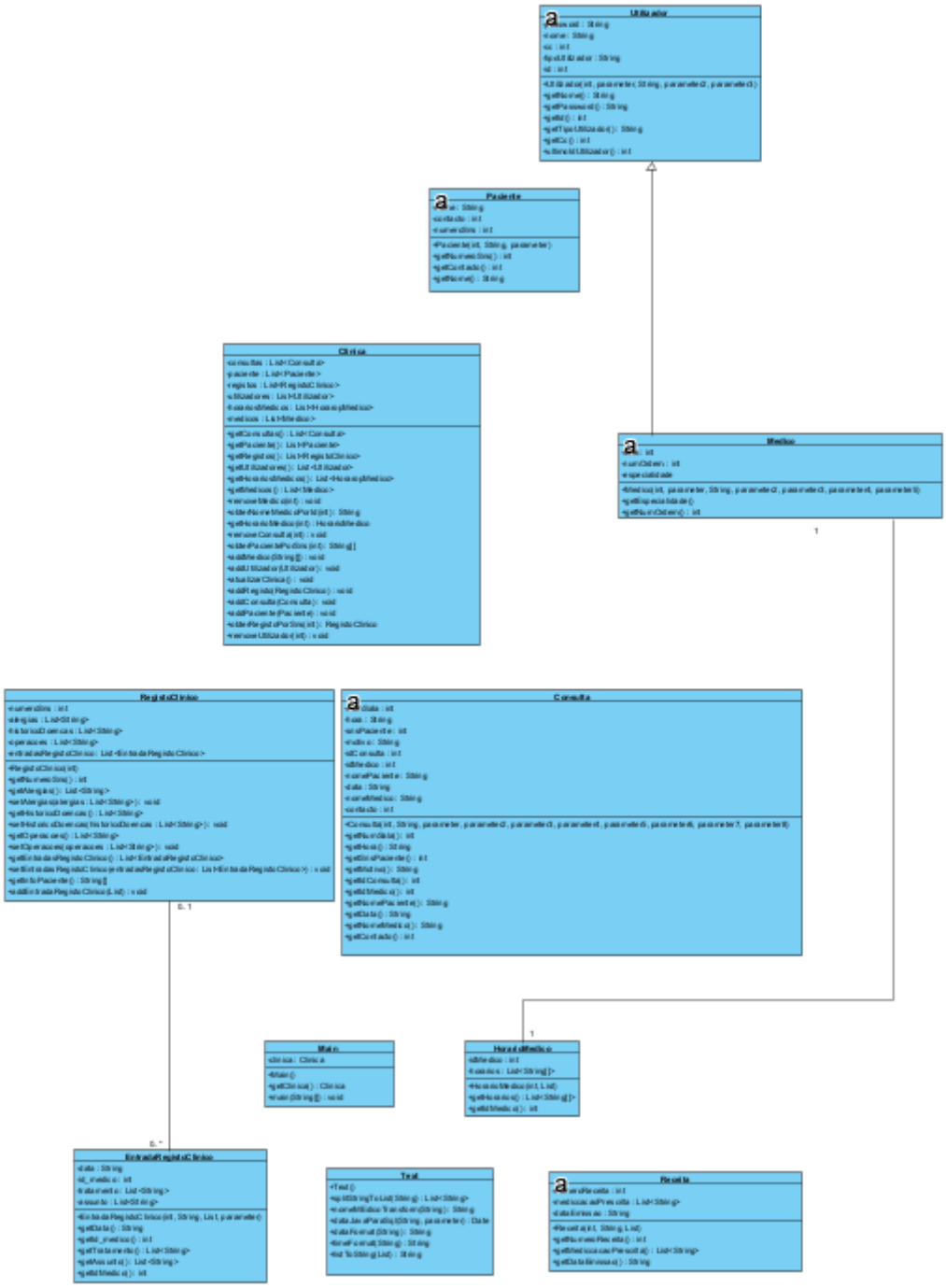
3.2.3.13 Entrada Registo Clinico

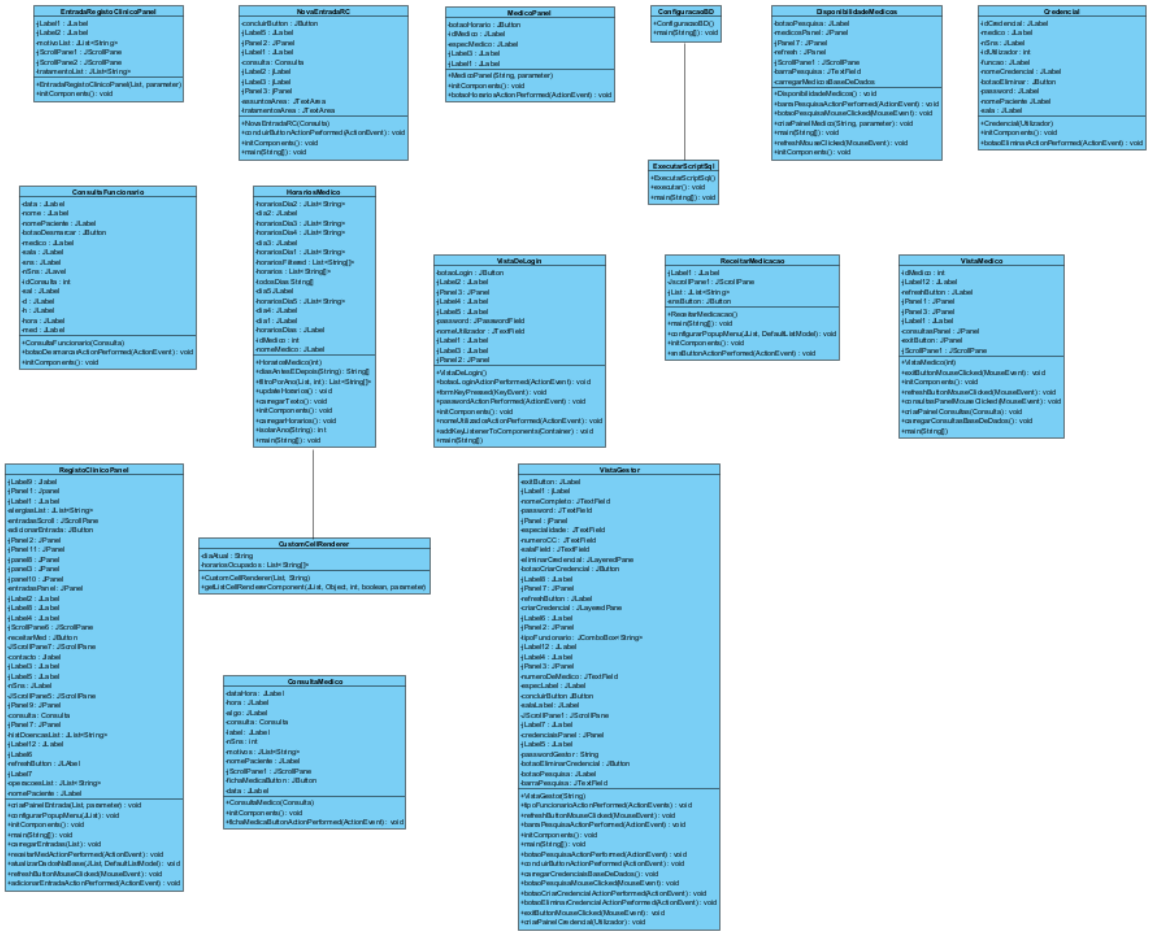
Métodos:

EntradaRegistoClinicoPanel(List<String> assunto, List<String> tratamento) - O construtor recebe duas listas de strings: uma para os "assuntos" (motivos da consulta) e outra para os "tratamentos" realizados. As listas são usadas para preencher os componentes gráficos (listas) do painel.

3.3 Diagrama de classes

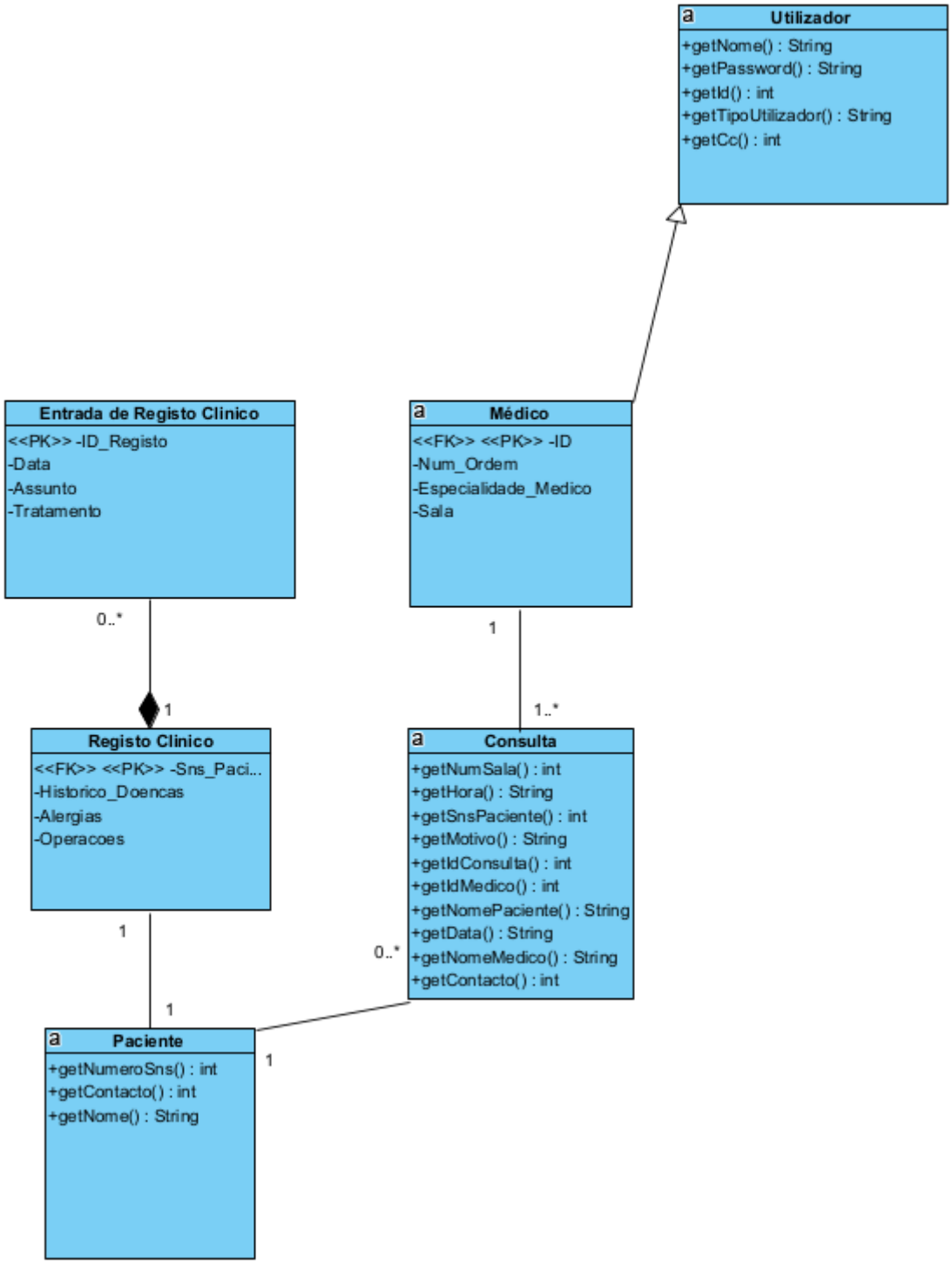





EntradaRegistoClinicoPanel
NovaEntradaRC
MedicoPanel
ConfiguracaoBD
DisponibilidadeMedicos
Credencial
ExecutarScriptSql
ConsultaFuncionario
HorarioMedico
VistaDeLogin
ReceitarMedicacao
VistaMedico
RegistoClinicoPanel
CustomCellRenderer
VistaGestor
ConsultaMedico

4 Modelo de dados persistente

4.1 Modelo conceitual



4.2 Modelo relacional

Utilizador(ID{PK}; CC_Utilizador{UNIQUE}; Nome_Utilizador; Password; Tipo_Utilizador)

Médico(ID{FK, NN}; Num_Ordem{PK}; Especialidade; Sala{UNIQUE, NN})

Paciente(Sns_Paciente{PK}; Nome_Paciente; Contacto_Paciente)

Registo Clínico(Sns_Paciente{PPK}; Historico_Doenças; Alergias; Operações)

Entrada Registo Clínico(ID_Registo{PK}; Sns_Paciente{FK}; Data; Assunto; Tratamento)

Consulta(ID_Consulta{PK}; ID_Medico{FK,NN}; Sns_Paciente{FK,NN}; Data_Consulta; Hora_Consulta; Motivo_Consulta; Nome_Medico)

5. Anexos

Anexo 1: Link para o Figma - Esboço inicial da interface gráfica do Projeto
URL:https://www.figma.com/design/jEofAZOfpLpfcnzFc152NW/MediFlow?node-id=0-1&t=4PWxIAP90hTPaQsl-1