UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JEAM MARTINS ALVES

UTILIZAÇÃO DO MODELO MVC NO DESENVOLVIMENTO DE APLICAÇÕES E-SAÚDE UTILIZANDO O PADRÃO OPENEHR

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2017

JEAM MARTINS ALVES

UTILIZAÇÃO DO MODELO MVC NO DESENVOLVIMENTO DE APLICAÇÕES E-SAÚDE UTILIZANDO O PADRÃO OPENEHR

Trabalho de conclusão de curso apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Bacharel em Sistemas de informação.

Orientador: Prof. Marcos Eduardo Pivaro Monteiro

Coorientador: Prof. Paulo Roberto Bueno

CURITIBA

2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Coordenação do Curso de Bacharelado em Sistemas de Informação

TERMO DE APROVAÇÃO

"UTILIZAÇÃO DO MODELO MVC NO DESENVOLVIMENTO DE APLICAÇÕES E-SAÚDE UTILIZANDO O PADRÃO OPENEHR"

por

"Jeam Martins Alves"

Este Trabalho de Conclusão de Curso foi apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho **aprovado**.

Prof. Marcos Eduardo Pivaro Monteiro (Presidente - UTFPR/Curitiba)	**Profa. Giovanna Garcia Basilio** (Avaliador 1 - UTFPR/Curitiba)
Prof. Eduardo Nunes dos Santos (Avaliador 2 - UTFPR/Curitiba)	**Prof. Leyza Baldo Dorini** (Professor Responsável pelo TCC – UTFPR/Curitiba)
Prof. Leonelo Dell Anhol Almeida (Coordenador(a) do curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba)	

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso."

A Deus, por ser essencial em minha vida e meu guia, aos meus pais, minha esposa que, com muito carinho e apoio, me acompanharam nesta jornada.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse ao longo de minha vida, a Universidade Tecnológica Federal do Paraná, Campus de Curitiba, aos meus professores, especialmente ao meu professor orientador, Marcos Eduardo Pivaro Monteiro que me ajudou a concluir este trabalho, aos meus colegas de curso, amigos e familiares, principalmente minha esposa Isabela Gomes Kill Alves que esteve ao meu lado e a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

ALVES, Jeam Martins. UTILIZAÇÃO DO MODELO MVC NO DESENVOLVIMENTO DE APLICAÇÕES E-SAÚDE UTILIZANDO O PADRÃO OPENEHR. 80 f. Trabalho de conclusão de curso – Curso de Bacharelado em Sistemas de informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

É de suma importância o compartilhamento de informações entre sistemas de saúde, uma vez que a troca de dados é essencial para a reutilização de processos e interfere significativamente na qualidade do atendimento médico. Para realizar um diagnóstico, utiliza-se de diversas fontes de informação, dentre elas o relato de um paciente, exames físicos, análise laboratorial, entre outros. O padrão openEHR surgiu com o intuito de prover a interoperabilidade entre os diversos *softwares* de saúde, possibilitando aos profissionais da área modelar o conhecimento clínico, que por sua vez torna-se disponível para ser implementado pelos profissionais de informática nos sistemas de *software*. O objetivo deste trabalho é utilizar o padrão openEHR no desenvolvimento de um software para gestão de consultas médicas aplicando o modelo de arquitetura MVC. Este projeto utilizou conceitos de engenharia de software, arquitetura de sistemas e de serviços e foi desenvolvido utilizando a linguagem de programação C# com banco de dados SQL Server. O resultado se deu na construção de uma aplicação utilizando o padrão MVC disponibilizada para ambientes *desktop* e *web*, que possibilita o gerenciamento de consultas médicas e o acompanhamento do histórico de atendimento médico pelo paciente. Obteu-se também como resultado a camada de modelo implementada no padrão openEHR em classes relacionadas ao modelo clínico e aos registros eletrônicos do paciente.

Palavras-chave: Interoperabilidade, openEHR, Sistema, Padrão.

ABSTRACT

ALVES, Jeam Martins. USE THE MVC PATTERN IN DEVELOPMENT OF E-HEALTH APPLICATIONS USING THE OPENEHR STANDARD. 80 f. Trabalho de conclusão de curso – Curso de Bacharelado em Sistemas de informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

The information sharing between health systems is an important task since the data exchange is essential for the process reuse, increasing the quality of medical care and the ease and accuracy of the obtained diagnostic. Different information sources can be used such as patient's narrative, physical examination, laboratory analysis and others. To promote the interoperability between the different developed softwares, the openEHR standard was proposed, allowing health professionals to describe their clinical knowledge which is, in turn, used by computer professionals to develop applications. The propose this work is to use the openEHR pattern in the development of medical management software with the MVC pattern. Concepts of software engineering, systems architecture and services were applied in this project, which was developed using the C# programming language and SQL Server database. The result is a web and desktop MVC application for medical management, allowing historical record tracking by the patient. Also obtained as a result the model layer implemented in the openEHR pattern in classes related to the model clinical and electronic records of the patient.

Keywords: Interoperability, openEHR, System, Pattern.

LISTA DE FIGURAS

FIGURA 1 – Exemplo de arquitetura de um sistema 17
FIGURA 2 – Exemplo de diagrama de classe 18
FIGURA 3 – Exemplo de arquitetura MVC 19
FIGURA 4 – Exemplo de arquitetura orientada a serviços 20
FIGURA 5 – Classes de arquétipos 23
FIGURA 6 – Estrutura de um arquétipo 25
FIGURA 7 – Ficha de atendimento 26
FIGURA 8 – Autenticar administrador 36
FIGURA 9 – Autenticar médico 37
FIGURA 10 – Autenticar paciente 38
FIGURA 11 – Cadastrar paciente 39
FIGURA 12 – Listar pacientes 41
FIGURA 13 – Editar dados do paciente 42
FIGURA 14 – Cadastrar médico 43
FIGURA 15 – Cadastrar consulta médica 45
FIGURA 16 – Listar funções do sistema 46
FIGURA 17 – Editar funções do sistema 47
FIGURA 18 – Listar médicos no sistema 49
FIGURA 19 – Histórico de consultas do paciente 50
FIGURA 20 – Exibir consulta realizada pelo médico 52
FIGURA 21 – Exibir consulta realizada do paciente 54

LISTA DE SIGLAS

MVC	Model View Controller
UML	Unified Modeling Language
SOA	Service Oriented Architecture
API	Application Programming Interface
RES	Registro Eletrônico de Saúde
CKM	Clinical Knowledge Manager
ADL	Archetype Definition Language
AOM	Archetype Object Model

SUMÁRIO

1 INTRODUÇÃO 11
1.1 MOTIVAÇÃO 13
1.2 JUSTIFICATIVA 13
1.3 OBJETIVOS 14
1.3.1 Objetivo Geral 14
1.3.2 Objetivos Específicos 14
1.4 ORGANIZAÇÃO DO DOCUMENTO 15
2 METODOLOGIA 16
2.1 FUNDAMENTOS 16
2.1.1 Arquitetura de Software 16
2.1.2 Padrão de Arquitetura MVC (Model-View-Controller) 18
2.1.3 SOA - Arquitetura de Software Orientada a Serviços 19
2.1.4 OpenEHR 21
2.1.4.1 Arquétipos 22
2.1.4.2 As classes de um arquétipo 22
2.1.4.3 Archetype Definition Language (ADL) 24
2.1.4.4 Templates 26
2.1.4.5 Quem está usando o padrão openEHR 27
2.2 TECNOLOGIAS UTILIZADAS 32
2.2.1 Visual Studio 32
2.2.2 ASP.NET 33
2.2.3 Sql Server 33
3 PROJETO DE SOFTWARE 34
3.1 REQUISITOS 34
3.1.1 Requisitos funcionais 35
3.1.2 Requisitos não funcionais 35
3.2 CASOS DE USO 36
3.2.1 Especificação detalhada 36
3.3 DIAGRAMA DE CLASSES 56
3.4 MODELO ENTIDADE RELACIONAMENTO 56
3.5 RECURSOS UTILIZADOS 57
3.5.1 Recursos de hardware 57
3.5.2 Recursos de Software 57
4 DISCUSSÃO 58
4.1 OPENEHR 58
4.1.1 Vantagens e Desvantagens 59
4.1.2 Uso do MVC 59
5 CONCLUSÃO 61
5.1 CONTRIBUIÇÕES 61
5.2 TRABALHOS FUTUROS 61
REFERÊNCIAS 63

Apêndice A – TELA DE LOGIN DA ÁREA MÉDICA 66
Apêndice B – TELA INICIAL DA ÁREA MÉDICA 67
Apêndice C – FORMULÁRIO DE CONSULTA MÉDICA 68
Apêndice D – LISTA DE CONSULTAS DE UM MÉDICO 69
Apêndice E – VISUALIZAÇÃO DOS DADOS DE UMA CONSULTA MÉDICA 70
Apêndice F – ÁREA DE LOGIN DO ADMINISTRADOR E DO PACIENTE 71
Apêndice G – PÁGINA INICIAL DA ÁREA ADMINISTRATIVA 72
Apêndice H – PÁGINA INICIAL DA ÁREA DO PACIENTE 73
Apêndice I – LISTA DE FUNCIONALIDADES 74
Apêndice J – FORMULÁRIO DE CADASTRO DO PACIENTE 75
Apêndice K – HISTÓRICO DE CONSULTAS DO PACIENTE 76
Apêndice L – VISUALIZAÇÃO DOS DADOS DE UMA CONSULTA DO PACIENTE 77
Apêndice M – DIAGRAMA DE CLASSES DA CAMADA DE MODELO DAO (*DATA ACCESS OBJECT*) 78
Apêndice N – DIAGRAMA DE CLASSES DA CAMADA DE MODELO *DOMAIN MODEL* 79
Apêndice O – DIAGRAMA DE MODELO ENTIDADE RELACIONAMENTO 80

1 INTRODUÇÃO

Aplicações que auxiliam no monitoramento e controle da saúde surgem a todo momento, como *softwares* para caminhada, aferição de pressão arterial, informações biométricas, *software* de gestão de clínicas médicas e outros. É evidente que sistemas de informação aplicados à saúde trazem alguns benefícios como: tornar os atendimentos médicos mais ágeis e eficientes, facilitar o acesso aos dados e construir o registro eletrônico de saúde do paciente (VICENTINI et al., 2010).

Ao longo do tempo, uma pessoa pode realizar vários atendimentos médicos em diferentes hospitais ou clínicas e até mesmo fazer acompanhamento e/ou monitoramento de saúde por meio de aplicativos. Sendo assim, várias informações de saúde são armazenadas em diferentes bases de dados, e quando essas aplicações não provêem a interoperabilidade entre si, os registros de saúde acabam limitando-se a uma aplicação específica, sendo impossível a comunicação desses dados com outros sistemas (PESSANHA; BAX, 2015).

Interoperabilidade é a capacidade de troca de dados entre dois ou mais sistemas e o uso mútuo desses dados (AGUILAR, 2005). A interoperabilidade pode ser sintática (funcional) ou semântica. No caso funcional o objetivo é que a informação trocada seja legível para os seres humanos. Já no caso semântico, as informações podem ser interpretadas pelos seres humanos e também pelos sistemas. Isso possibilita um aprimoramento dos *softwares* de informações de saúde e o uso de sistemas de apoio à decisão (MEZAROBA; NICOLEIT, 2010).

A interoperabilidade semântica é essencial para o desenvolvimento de aplicações clínicas sofisticadas, no qual o ponto central é trocar dados e informações e reutilizar processos e dados de forma mais inteligente (AGUILAR, 2005). Visto que há necessidade de compartilhamento de informações de saúde, foi criado o padrão openEHR. A iniciativa GERH/openEHR começou em 1992 como projeto de pesquisa europeu. Atualmente, ele é mantido pela fundação openEHR, que apresenta uma comunidade que trabalha para transformar dados de saúde a partir da forma física, em formato eletrônico, e garantir a

interoperabilidade universal de dados (SOUZA, 2014).

O openEHR é um modelo padronizado de informações de saúde e bem estar do indivíduo, onde as informações são integradas e processadas por computador e disponibilizadas para uso por vários utilizadores autorizados de forma segura (MAIA, 2014). O padrão surgiu com o intuito de prover a interoperabilidade entre sistemas de saúde, sendo sua arquitetura orientada a serviços em dois níveis, a saber, o modelo de arquétipos e o modelo de referência, onde os profissionais de saúde modelam esses arquétipos e os profissionais de informática integram o modelo de referência no *software*.

Os arquétipos possuem uma linguagem própria, estão em conformidade com a norma ISO 13606-2, podem ser manipulados por profissionais de saúde de forma independente e são usados para gerar os modelos de referência (MEZAROBA; NICOLEIT, 2010). É possível representar modelos como "pressão sanguínea", um exame laboratorial, entre outros. Uma das vantagens dos arquétipos é o reuso do conhecimento clínico (PESSANHA; BAX, 2015), enquanto o modelo de referência é composto por diversos modelos de informação genéricos com o intuito de criar uma representação global de informações (MEZAROBA; NICOLEIT, 2010), proporcionando a interoperabilidade semântica.

As vantagens quanto ao uso do padrão openEHR são claras tanto no desenvolvimento de aplicações quanto para a área da saúde. A manutenabilidade de sistemas é baixa, há uma melhora na abstração do modelo de dados e a troca de informações entre centros de saúde, hospitais e clínicas é mais ágil. Sendo assim, garante uma melhor qualidade na informação (SOUZA, 2014), possibilitando também uma expansão futura no sistema sem a necessidade de atualização de código-fonte.

1.1 MOTIVAÇÃO

A informação é de extrema importância para os profissionais de saúde, uma vez que os serviços prestados dependem da qualidade dessa informação. Os médicos gastam um quarto de sua vida profissional realizando a gestão de informações clínicas e os enfermeiros gastam cinquenta por cento do seu tempo gerindo informações (BACELAR; CORREIA, 2015). Chegam aos profissionais informações de diferentes fontes, iniciando com os relatos de algum problema do paciente, exames físicos e ainda análises de laboratório. Toda a informação gerada deve ser armazenada para auxiliar atendimentos posteriores.

A disseminação da informação nos dias de hoje está cada vez mais veloz, e o volume de informação médica na internet, hoje, se duplica a cada seis meses. Os sistemas de informação voltados para a área de saúde precisam ser fáceis de atualizar, uma vez que conhecimentos médicos mudam, e para manter o sistema atualizado é necessário realizar alterações de base de dados e também no código (BACELAR; CORREIA, 2015).

O padrão openEHR tem o propósito de atender a necessidade de atualização dos modelos de dados de saúde. Para atualizar a aplicação com um novo conhecimento clínico, a alteração é simples, pois é necessário alterar os arquétipos e *templates*. Após as modificações, o sistema já está atualizado, não sendo necessário realizar modificações na estrutura do *software* e do banco de dados (BACELAR; CORREIA, 2015).

1.2 JUSTIFICATIVA

Visto que as informações médicas de um paciente estão pulverizadas em vários locais como clinicas médicas, hospitais, postos de saúde, entre outros, há uma necessidade de interligação dessas informações, uma vez que, ao consultar em uma clínica médica, não é possível disponibilizá-las para outros estabelecimentos médicos. Assim, não existe interoperabilidade entre os sistemas de saúde, as informações acabam se perdendo ou até mesmo gerando a mesma informação em vários lugares.

Um dos propósitos do openEHR é prover a interoperabilidade entre os diversos sistemas médicos, tornando possível a interligação das informações. Ao realizar uma consulta em uma clínica, o histórico do paciente pode ser recuperado, por exemplo, em um hospital, facilitando o atendimento médico e possibilitando aos profissionais de saúde realizarem um atendimento com um nível mais elevado de qualidade. Todo o histórico

de saúde é de propriedade do paciente e não de um estabelecimento médico. Com o openEHR, o paciente tem o controle das informações já geradas.

Outra justificativa é de que o Ministério da Saúde tem como meta a implantação dos padrões de Registro Eletrônico de Saúde (RES), especificamente o openEHR, no Sistema Único de Saúde (SUS) (SUGAI, 2008). Planejamento no qual foi iniciado no ano de 2008 incluindo a implantação de um repositório nacional de arquétipos e *templates* e um serviço central para armazenamento de dados dos pacientes. Os dados ficarão disponíveis para todo o Brasil em clínicas, hospitais, laboratórios de análises clínicas, e outros estabelecimentos de saúde devidamente regulamentados.

Finalmente, para o desenvolvimento deste trabalho deve-se ao fato de que o autor deste estudo possui um frequente contato com a área de saúde, e atuou cinco anos em empresas farmacêuticas no setor de processamento de dados, além de ter familiares desempenhando trabalhos diretamente ligados à saúde como enfermeiros, biólogos e também farmacêuticos. Assim, o autor sentiu-se motivado a contribuir para o desenvolvimento tecnológico nesta importante área de negócio.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Este trabalho tem como objetivo desenvolver uma aplicação utilizando o padrão openEHR com a arquitetura de *software* modelo-visualização-controle (MVC, do inglês *Model-View-Controller*), para desenvolvimento de sistemas e-saúde.

1.3.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um *software* para gestão de consultas médicas.
- Demonstrar a utilização do padrão openEHR em uma aplicação utilizando a arquitetura MVC;
- Demonstrar as boas práticas para o desenvolvimento de aplicações, entre elas a utilização de técnicas de modelagem para resolver problemas na área de desenvolvimento de *software*, a reutilização de código e o reuso de objetos de modelo.

1.4 ORGANIZAÇÃO DO DOCUMENTO

Este trabaho está subdividido em 5 capítulos. O Capítulo 2 metodologia, arquitetura de *software*, o padrão MVC e arquitetura orientada a serviços. Também apresenta o padrão openEHR, sua organização e composição, arquétipos e *templates* e um panorama de quem está utilizando o openEHR no mundo.

Já no Capítulo 3 é especificado o projeto do *software* desenvolvido contendo os requisitos funcionais e não funcionais do sistema, os casos de uso, o diagrama de classes e diagrama de entidade e relacionamento, além de apresentar os recursos de *hardware* e *software* utilizados durante o trabalho. No Capítulo 4 propõe uma discussão sobre o openEHR, bem como suas vantagens e desvantagens e a utilização do padrão de projeto MVC. Por fim, no Capítulo 5 é apresentada a conclusão com as contribuições e trabalhos futuros.

2 METODOLOGIA

2.1 FUNDAMENTOS

Nesta seção será apresentada a fundamentação teórica sobre os assuntos relacionados ao desenvolvimento deste trabalho, iniciando-se com a arquitetura de *software*, sua definição e a linguagem utilizada para representá-la. Em seguida, são apresentados o padrão de projeto MVC e uma arquitetura orientada a serviços. Por fim, o padrão openEHR é apresentado descrevendo a definição de arquétipos e suas classes, *templates*, pesquisas e utilização na indústria.

2.1.1 ARQUITETURA DE SOFTWARE

Com o aumento na complexidade e no tamanho dos *softwares*, nos anos 90 houve uma grande preocupação em entender como os sistemas de *software* são organizados, dando origem a área de arquitetura de *software* (NAKAGAWA, 2006). Com o seu uso é possível, antes do desenvolvimento, raciocinar sobre os cinco atributos básicos em um sistema de *software*: confiabilidade, funcionalidade, modularização, performa e segurança. Baseando-se apenas em modelos de design e descrições de arquitetura (CARVALHO, 2006).

A arquitetura de *software* é definida como um conjunto de componentes de um *software*, seus relacionamentos e as normas que regem os projetos e a evolução dos *softwares* (CARVALHO, 2006). Pode-se também definir a arquitetura de *software* como o conjunto de componentes e seus relacionamentos, que devem satisfazer os requisitos funcionais e não funcionais do sistema (ANDRADE et al., 2006). Por isso, a arquitetura de *software* é tida como o artefato mais importante para tornar possível um controle incremental e iterativo do sistema durante o seu ciclo de vida (BOOCH et al., 2006).

Uma arquitetura de *software* deve apoiar questões de projetos como: a organização do *software* como composição de componentes, estruturas de controles de forma

global, protocolos de comunicação, composição dos elementos de projeto e a designação desses componentes. Existem alguns aspectos que auxiliam a identificar o estilo de arquitetura de um sistema:

- Características dos componentes e conectores do sistema;
- Topologia da arquitetura;
- Restrições semânticas;
- Formas de interação entre os componentes.

A arquitetura de *software* é constituída de um modelo compreensível de como é a estrutura de um sistema como um todo e como seus elementos se organizam e trabalham em conjunto. Esse modelo é transferível no sistema podendo ser reutilizado em outros sistemas que tenham atributos e requisitos funcionais similares (RODRÍGUEZ; NAKAGAWA, 2015). A arquitetura de um sistema complexo pode ser resumida em cinco visões conforme Figura 1 (BOOCH et al., 2006).

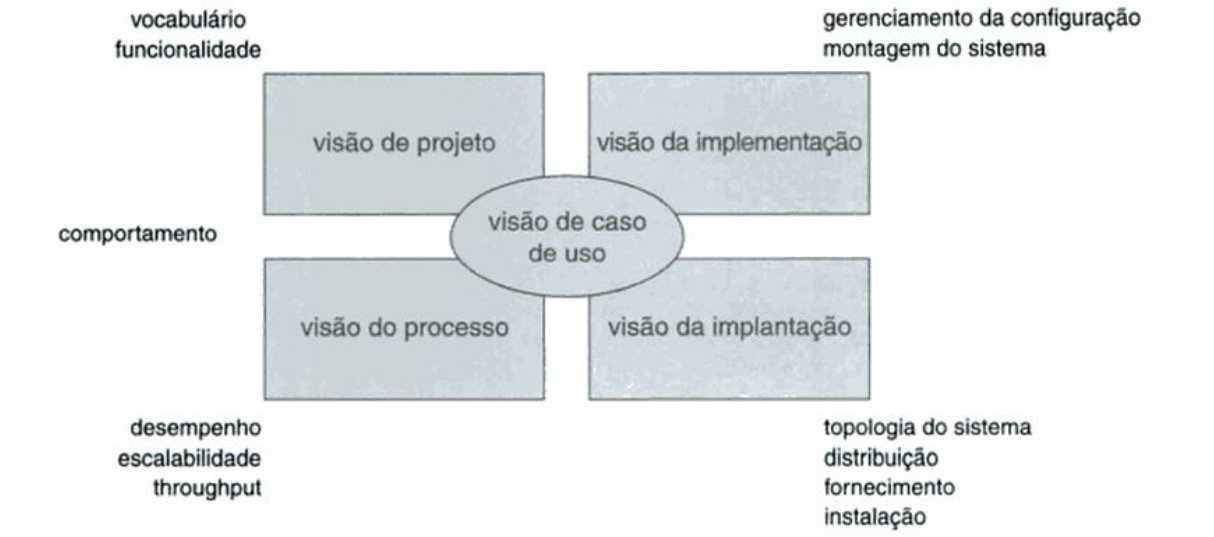


Figura 1: Exemplo de arquitetura de um sistema
Fonte: (BOOCH et al., 2006)

A visão de caso de uso, compreende os casos de uso que descrevem o comportamento do sistema conforme visualizado pelos usuários finais, analistas e *testers*. A visão de projeto abrange as classes, interfaces e colaborações que proporciona um suporte para os requisitos funcionais do sistema. A visão do processo engloba os processos que formam os mecanismos de concorrência e sincronização do sistema. Esta visão cuida de questões

relacionadas ao desempenho e escalabilidade. A visão da implementação se refere ao gerenciamento da configuração das versões do sistema, composta por arquivos e componentes independentes. A visão da implantação direciona o fornecimento, a distribuição e a instalação das partes que compõem o sistema físico (BOOCH et al., 2006).

A linguagem mais utilizada para descrever uma arquitetura de *software* é a UML (*Unified Modeling Language*), que apresenta uma linguagem-padrão para elaborar projetos de *software* e tem como objetivo a visualização, especificação, construção e documentação de artefatos que utilizam-se de sistemas de *softwares* complexos (BOOCH et al., 2006). A utilização da UML para documentar arquitetura de *software* é benéfica, pois é familiar para os desenvolvedores, facilita o mapeamento para a implementação e tem o apoio de ferramentas de *software* (NAKAGAWA, 2006). A Figura 2 mostra um exemplo visual da modelagem UML de um diagrama de classes. Podemos visualizar a classe JogoDeDados e a classe Dado, bem como seus atributos, métodos e sua cardinalidade.

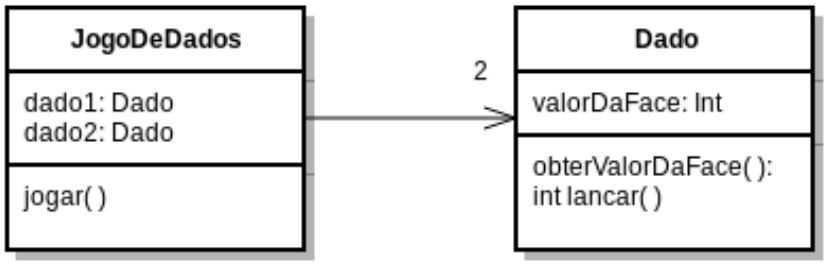


Figura 2: Exemplo de diagrama de classe UML
Fonte: (LARMAN, 2002)

2.1.2 PADRÃO DE ARQUITETURA MVC (MODEL-VIEW-CONTROLLER)

O padrão MVC é uma arquitetura de *software* estruturada em três camadas. A sigla MVC representa a definição dessas camadas, são elas: *Model*, responsável por representar a informação de domínio de negócios de uma aplicação, *View*, responsável pela apresentação dos dados e o *Controller*, chamado de controle, responsável por executar tarefas que envolvem o comportamento da aplicação, recebe informação da *view* e atualiza o modelo de dados de acordo com as ações do usuário.

A separação de uma aplicação em três camadas trás benefícios ao desenvolvedor, permitindo a reutilização de objetos do modelo em diferentes visualizações (DALL'OGLIO, 2009). O que deixa a aplicação mais organizada e proporcionando manutenabilidade do código, além de garantir eficiência e escalabilidade (SILVA, 2012). Como

podemos ver na Figura 3, a *view* passa as informações ou ações do usuário para o *controller*, que por sua vez instancia o objeto de modelo e, além de ser capaz de executar outras tarefas como, por exemplo, instanciar bibliotecas instaladas na aplicação, e, por fim retorna a resposta para a *view*.

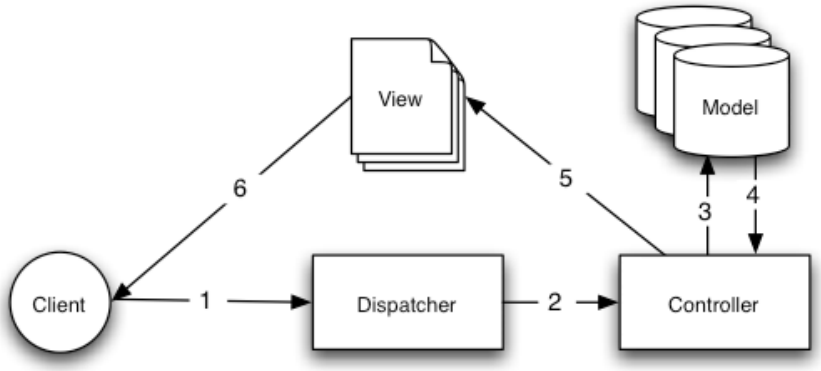


Figura 3: Exemplo de arquitetura MVC
Fonte: (FOUNDATION, 2017)

2.1.3 SOA - ARQUITETURA DE SOFTWARE ORIENTADA A SERVIÇOS

A SOA (*Service Oriented Architecture*) estabelece um modelo de arquitetura corporativa permitindo a criação de serviços que possibilita a interoperabilidade, a facilidade no reuso e o compartilhamento de regras de negócio entre sistemas (JUNIOR, 2013). Pode-se descrever o conceito de serviços em uma aplicação como a disponibilização de funcionalidades de um *software* através de uma interface API (*Application Programming Interface*) (JUNIOR, 2013).

A SOA surgiu com o intuito de reduzir custos e recursos com integrações de sistemas de informação envolvidos nas organizações. Consequentemente, impacta no aumento da agilidade no desenvolvimento de novos processos e produtos da organização. Com essa arquitetura, a integração entre sistemas torna-se rápida e flexível, possibilitando também a reutilização de aplicações legadas (ZAMBIASI et al., 2012). Para disponibilizar um componente como serviço, o desenvolvedor precisa especificar o que o serviço faz, seus dados de entrada e saída. Para a utilização do componente é irrelevante disponibilizar detalhes aprofundados da implementação (SILVA, 2006).

O baixo acoplamento é uma das características da arquitetura SOA, pois cada atividade de um processo de negócio pode ser implementada como um componente inde-

pendente no sistema (SILVA, 2006). Essa estratégia é interessante quando uma determinada atividade necessita ser compartilhada em várias partes do sistema, proporcionando seu reuso. Outra característica importante que a SOA proporciona é a possibilidade de realizar alterações nos componentes sem que envolva os consumidores desses serviços e também a possibilidade de reescrever somente partes da aplicação (JUNIOR, 2013).

Visto que arquitetura proporciona uma maior liberdade de desenvolvimento e uma neutralidade em relação à tecnologia, alcança-se a interoperabilidade, fator importante que proporciona a comunicação entre sistemas independente da tecnologia (SILVA, 2006). A Figura 4 ilustra um exemplo, no qual a camada "Serviços" disponibiliza os recursos para integração com outros sistemas.

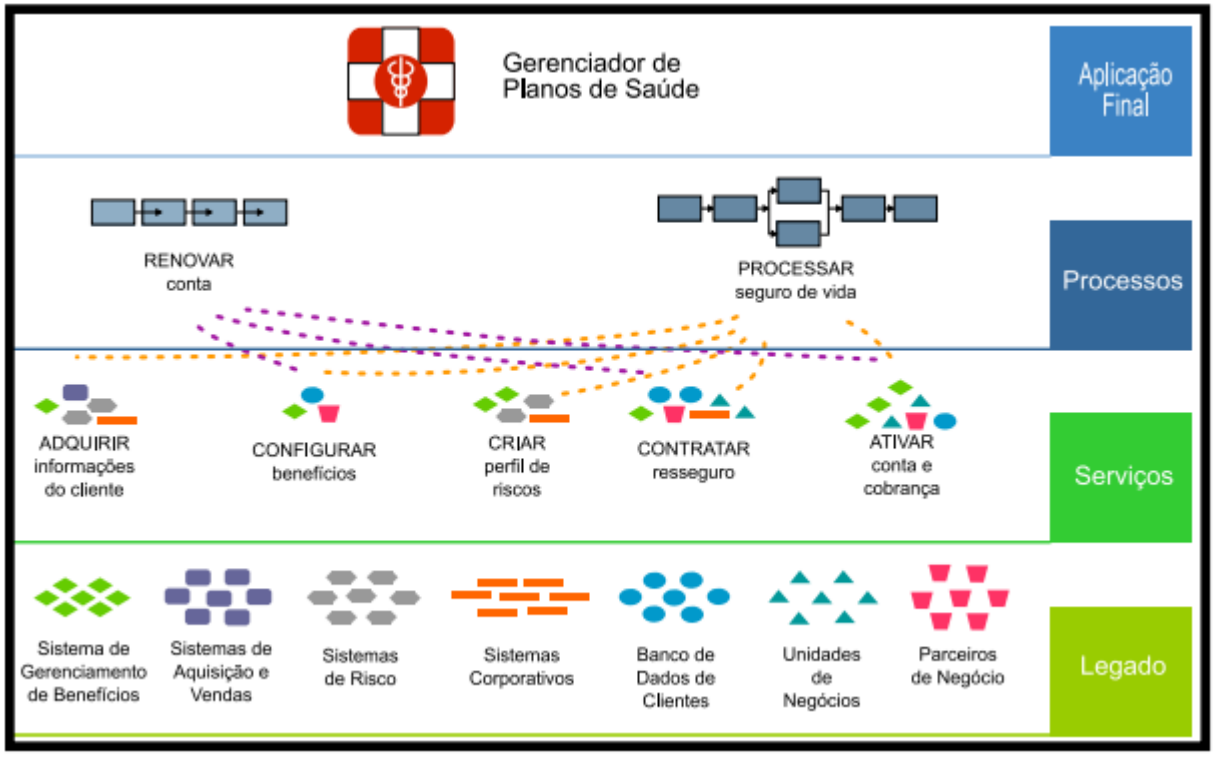


Figura 4: Exemplo de arquitetura orientada a serviços para um sistema de gerenciamento de plano de saúde

Fonte: (SILVA, 2006)

Na camada "Legado", temos os sistemas utilizados pelo cliente atualmente. São soluções desenvolvidas há algum tempo que divergem com a realidade atual do negócio (SILVA, 2006). Esses sistemas utilizam serviços disponibilizados na camada "Serviços".

2.1.4 OPENEHR

O openEHR é um conjunto de especificações e ferramentas livre para criação de registros eletrônicos de saúde (RES) modulares e interoperáveis (CORREIA, 2012). A norma openEHR segue o paradigma de modelagem multinível, separados em camadas de informação e conhecimento, disponibilizadas em uma arquitetura orientada a serviços (SOUZA, 2014). No qual a camada de informação representa os conceitos clínicos que são utilizados nos *softwares* e a camada de conhecimento é específica para os profissionais de saúde modelarem as características mais adequadas para o registro de saúde (PESSANHA; BAX, 2015). O padrão tem como objetivo:

- Protagonizar interação entre o paciente e a instituição de saúde;
- Adequação dos serviços de saúde prestados;
- Disponibilizar informações de saúde para os pacientes de forma segura.

Uma das vantagens do openEHR é a independência tecnológica, pois o conhecimento clínico é estruturado usando entidades do modelo de informação (BACELAR; CORREIA, 2015). A principal característica no desenvolvimento de registros clínicos é a modelagem da informação, a especificação openEHR separa de maneira clara o que é a abstração do mundo real e modelo de informação, por exemplo, conceito de quantidade e conceito de pressão sanguínea (CORREIA, 2012).

O conhecimento clínico é definido formalmente em arquétipos (camada de conhecimento), que pode ser descrito como um modelo formal, podendo ser reutilizável em outros cenários que haja a necessidade de sua aplicação (PESSANHA; BAX, 2015). Com uma linguagem própria, os arquétipos são manipulados pelos profissionais de saúde utilizando entidades da camada de informação (MEZAROBA; NICOLEIT, 2010). Em linhas gerais, o modelo de arquétipos pode ser visualizado como uma representação de metadados com o intuito de padronizar dados do conhecimento clínico, que por sua vez são obtidos de forma estruturada fora do *software* (PESSANHA; BAX, 2015).

Para expressar um conjunto de arquétipos em um contexto específico, são utilizados os *templates* como, por exemplo, um sumário de alta. Em suma, os *templates* são compostos por agregações de arquéipos (MEZAROBA; NICOLEIT, 2010). Os arquétipos e *templates* podem ser obtidos no repositório internacional CKM (*Clinical Knowledge Manager*), gerido pela fundação openEHR, entidade que faz a gestão das normas e disponibiliza ferramentas que permite sua utilização. Esse repositório permite que os profissionais

de saúde realizem a submissão dos modelos, que então tornam-se disponíveis para reuso (CORREIA, 2012).

2.1.4.1 ARQUÉTIPOS

A definição de arquétipos pode variar dependendo do perfil profissional. Para o profissional de saúde, o arquétipo é uma representação de um conceito clínico (peso, temperatura, altura), já para o profissional de informática, o arquétipo é um modelo computável com o objetivo de registrar dados. De fato, a definição de um arquétipo openEHR é a junção de ambas as definições, é um modelo computável de um conceito clínico estruturado. A utilização da linguagem ADL (*Archetype Definition Language*), é o que torna o modelo computável, uma vez que esta linguagem é um modelo de referência para definição de dados (BACELAR; CORREIA, 2015).

O arquétipo precisa ser único, amplo e o mais completo possível para um mesmo conceito, pois se houver mais de um modelo, voltaríamos ao tradicional problema de falta de interoperabilidade. Porém, há uma forma de utilizar apenas alguns atributos definidos no arquétipo ou os atributos necessários e, para isso, definimos as particularidades no desenvolvimento do *template*. Os arquétipos podem ser utilizados para atender uma necessidade clínica, serem compartilhados em repositórios (como, por exemplo, o CKM), ou serem reutilizados em diversos *templates* e em vários sistemas (BACELAR; CORREIA, 2015).

2.1.4.2 AS CLASSES DE UM ARQUÉTIPO

Os arquétipos diferenciam entre si e cada tipo de arquétipo é adequado à uma necessidade de informação sendo, por isso, os arquétipos estruturados em classes. Cada classe diz respeito a um fim específico, de maneira que existem classes de arquétipos voltadas para organização e estrutura, representação de histórico familiar, entradas administrativas, classe que representa outro arquétipo, entre outras finalidades. Na Figura 5 é exibida as principais classes de arquétipos e como são estruturadas.

Figura 5: Classes de arquétipos
Fonte: (BACELAR; CORREIA, 2015)

Composição

Os arquétipos do tipo composição podem ser utilizados para registrar um atendimento como, por exemplo, dados de uma consulta. Um arquétipo dessa classe pode ser vazio inicialmente, sendo mais tarde composto por arquétipos de outras classes. Tais arquétipos podem ser classificados como de natureza eventual ou persistente, onde a diferença é que o arquétipo de natureza eventual pode ser utilizado para registro de informações de um atendimento que futuramente terá pouco impacto e o arquétipo de natureza persistente serve como referência para atendimentos futuros.

Secção

Este tipo de arquétipo é utilizado somente para organizar a estrutura e navegação de um RES, de maneira que os documentos são separados em secções e subsecções, os conteúdos são organizados em compartimentos específicos, sendo possível representar um arquétipo de seção dentro de outra seção. É comumente utilizado para representar informações de cabeçalho de formulários como “exame físico”e “dados vitais”.

Entradas Administrativas

Em arquétipos da classe Entradas Administrativas, são representadas informações relacionadas a um processo clínico, seguro de saúde, ou seja, todo o tipo de informação administrativa, portanto, não é relevante clinicamente.

Entradas Clínicas

Todas as sub-classes como: Observação, Avaliação, Instrução, Ação e Cluster, são relacionadas a dados clínicos.

Observação - Representa a informação livre de interpretação, engloba tudo que

é relatado pelo paciente. É composto por dados, protocolo, estado do paciente e eventos.

Avaliação - A partir das observações, os profissionais de saúde analisam e geram suas interpretações como, por exemplo, um diagnóstico. É composto por dado e protocolo.

Instrução - São representadas informações para instrução de um cuidado de saúde como, por exemplo, solicitação de exames, prescrições médicas, entre outros. É modelado com os atributos de atividade e protocolo.

Ação - Esta classe modela os conceitos de atividades clínicas. Os arquétipos da classe Ação são semelhantes aos da classe Observação, onde a diferença é que os arquétipos da classe Ação são associados a uma intervenção e precedidos por um arquétipo de Instrução. Exemplo: procedimento cirúrgico, administração de medicamento intravenoso. Tem como atributos: descrição da atividade (*activity*), caminho percorrido para a execução (*Pathway*) e protocolo.

Cluster -Os arquétipos dessa classe possuem uma característica particular, podem ser reutilizados dentro de outros arquétipos, ou seja, é uma parte de um arquétipo. Conceitos clínicos que podem ser reutilizados em outros contextos dentro do fluxo de atendimento (BACELAR; CORREIA, 2015).

2.1.4.3 ARCHETYPE DEFINITION LANGUAGE (ADL)

A linguagem ADL utiliza três sintaxes: modo de definição de dados (dADL), modo de restrições a *constraint* ADL (cADL) e lógica de predicados de primeira ordem a *First-Order Predicate Logic* (FOPL). A dADL é utilizada para expressar dados baseados em um modelo de informação, de maneira que esta sintaxe é utilizada pelas seções: *language* e *translation*, *description*, *terminologies_available*, *term_definitions*, *constraint_definitions*, *term_binding* e *constraint_binding*.

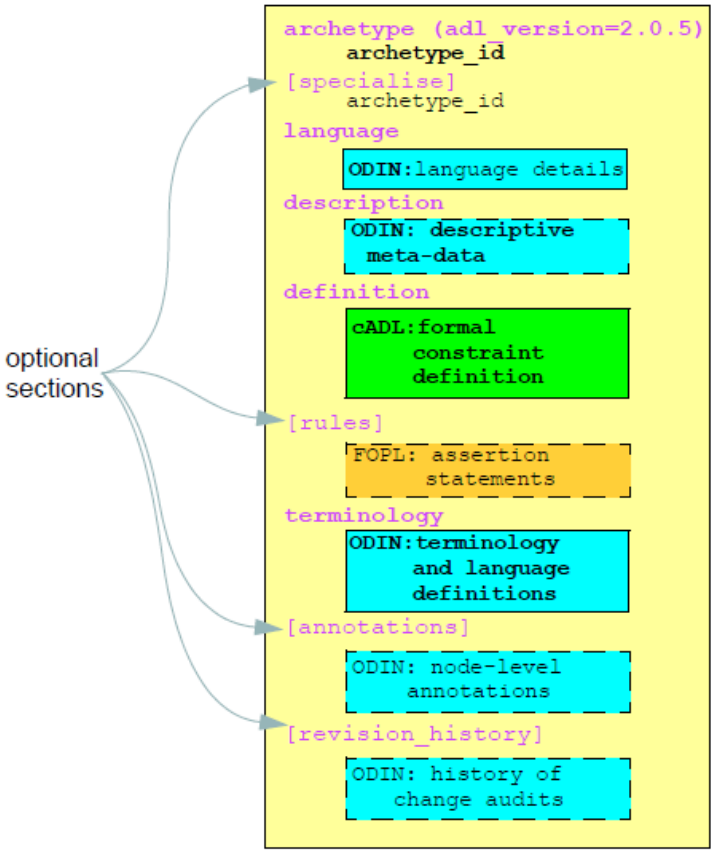


Figura 6: Estrutura de um arquétipo

Fonte: (OPENEHR, 2017a)

A cADL é aplicada em restrições sobre dados definidos por modelos orientados a objetos. Pode ser utilizada por algoritmos em tempo de execução para validação e em desenvolvimento (PESSANHA; BAX, 2016). Conforme a Figura 6, a estrutura básica de um arquétipo é composta por *header*, *definition*, *rules* ou *invariant*, *ontology* e *revision_history*; as seções *specialise*, *rules*, *anotations* e *revision_history* de um arquétipo são opcionais (PESSANHA; BAX, 2016):

- *Header*: Embora não esteja descrito na figura 6, ele é composto pelas subseções: *archeytpe*, *specialise*, *language*, *translation* e *description*. Descreve a identificação do arquétipo, objetivo, idioma original, traduções (caso existir);
- *Definition*: Onde são definidas as restrições para atributos e objetos;
- *Invariant/rules*: São definidas a lógica de primeira ordem (FOPL);
- *Ontology*: Define a representação de objetos. As subseções são: *terminology* e *annotations*;

- *Revision_history*: É documentado as alterações realizadas no arquétipo.

2.1.4.4 TEMPLATES

Como descrito anteriormente, um arquétipo é a representação da informação em uma forma mais ampla, ou seja, um modelo genérico que pode ser utilizado em todos os casos de uso. Porém, nem sempre é necessário utilizar todas as propriedades modeladas nos arquétipos e, por isso, surge o conceito de *template*, para modelar uma situação específica (BACELAR; CORREIA, 2015). Um *template* pode ser composto por um ou mais arquétipos e é definido por AOM (*Archetype Object Model*) e ADL, incluindo metadata especialização semântica, ontologias, regras e *annotations* em seções; são também utilizados para geração de formulários ou até mesmo relatórios. Todo dado gerado a partir de um *template* segue a especificação dos arquétipos (OPENEHR, 2017b).

Figura 7: *Template* de ficha de atendimento
Fonte: Adaptado do curso Capacitação em openEHR

Na Figura 7 temos um *template* de um formulário de exame físico composto por arquétipos de sinais vitais, temperatura corpórea, peso corporal entre outros. Para criar o *template* é utilizado uma ferramenta chamada *template designer*, disponibilizado pela fundação OpenEHR. Os profissionais de saúde criam os *templates* e, após sua criação, o arquivo é exportado e então os profissionais de informática os implementam em *softwares*

baseados no padrão openEHR (OPENEHR, 2017a).

2.1.4.5 QUEM ESTÁ USANDO O PADRÃO OPENEHR

A Indústria e a pesquisa estão atuando de maneira conjunta na evolução e desenvolvimento do OpenEHR. Existem atualmente dez países atuando em áreas de pesquisas acadêmicas e algumas empresas que desenvolveram soluções ou que auxiliam no desenvolvimento do padrão e relacionadas com o openEHR. A Tabela 1 mostra a lista de países com pesquisas científicas:

Tabela 1: Países que participam de pesquisas relacionadas ao openEHR.

País	**Instituição**	**Descrição**
Argentina	Hospital Universitário Austral	Implementação do openEHR como um projeto de longo prazo.
Brasil	UFRJ - Universidade Federal do Rio de Janeiro	Tese de doutorado em sistemas baseados em arquétipos.
	SENAI - Cimatec	OpenEHR sistema de informação de saúde abrangente para a atenção primária: Primeira Experiência Brasileira de Atendimento Público.
	Universidade Federal Fluminense	Sistema de Apoio à Vigilância Epidemiológica.
	UFMG - Universidade Federal de Minas Gerais	Tese de doutorado sobre interoperabilidade semântica envolvendo o uso de modelos openEHR RM e AM.
	Hospital das Clínicas Universidade de São Paulo Escola de Medicina de Ribeirão Preto	Aplicação de arquétipos para construção de observações clínicas e relatórios radiológicos.

Tabela 1 : (continuação)

País	Instituição	Descrição
Alemanha	Universidade de Heidelberg / Heilbronn University	Atividades desenvolvidas: • Expressando Conjuntos de Dados Clínicos com Arquetipos openEHR. • Modelagem de um registro eletrônico abrangente de pacientes para o Departamento de Neonatologia do Hospital Universitário de Heidelberg utilizando a abordagem openEHR.
Japão	Ehime University	Implementação openEHR ruby on rails.
Nova Zelândia	Universidade de Auckland, Instituto Nacional de Inovação em Saúde	O GastrOS é um aplicativo de relatório endoscópico baseado em padrões abertos: openEHR e MST. O objetivo é investigar a manutenção e interoperabilidade do *software*.
Portugal	Faculdade de Medicina, Universidade do Porto	Tese de Mestrado (Gustavo Bacelar) sobre a conversão de declarações de diretrizes clínicas para arquétipos e modelos openEHR.

Tabela 1 : (continuação)

País	Instituição	Descrição
Espanha	Universidade Técnica de Valência, Grupo de Informática Biomédica	Desde 2004, o grupo de pesquisa vem trabalhando na aplicação da metodologia de modelo duplo em sistemas legados, fornecendo ferramentas e metodologias para atualizar dados existentes em instâncias arquetípicas de acordo com qualquer modelo de referência selecionado.
	Universidade de Seville	Integração de sistemas de EHR federados usando técnicas de *Web* Semântica - a integração de sistemas heterogêneos usando o modelo de referência openEHR ea metodologia do arquétipo.
	Universidade de Murcia	O projeto POSEACLE teve início em 2004 com o objetivo de facilitar a gestão semântica de informações e conhecimentos relacionados com os cuidados de saúde eletrônicos. Este projeto de investigação foi realizado em cooperação com o grupo de Informática Biomédica da Universidade Técnica de Valência (Espanha).
	Universidade de Santiago de Compostela	Projeto de gestão de terminologias para arquétipos.

Tabela 1 : (continuação)

País	Instituição	Descrição
República Eslovaca	Universidade Técnica Eslovaca, Bratislava	Atividades desenvolvidas: • Localização de arquétipos openEHR para o Sistema Nacional de Informação em Saúde; • Desenvolvimento de arquétipos para GP Sistemas de informação; • Projeto de pesquisa: transformação de prontuários de texto livre para a forma arquetípica.
Suécia	Instituto Karolinska	Programa de doutorado para explorar como a tecnologia de EHR semântica, openEHR em particular, pode ser estendida para apoiar os processos clínicos distribuídos. Isto será feito explorando como incorporar diretrizes de prática clínica e vias clínicas em openEHR, usando diretrizes do domínio clínico de cuidados de AVC.

Tabela 1 : (continuação)

País	Instituição	Descrição
Reino Unido	CHIME (Centro de Informática em Saúde e Educação Multiprofissional), University College London	Opereffa é uma prova de conceito de implementação de aspectos-chave da especificação openEHR. Sua versão inicial explora os principais requisitos de implementação, desde uma interface baseada na *Web* até um *back-end* baseado em RDMS. Ele inclui alguns plugins do Eclipse e Eclipse BIRT (*Business Intelligence e Reporting Tools*) integração.

Fonte: (OPENEHR, 2017a)

Segundo (CESAR, 2013) é possível listar algumas empresas envolvidas no desenvolvimento de soluções baseadas em openEHR no mundo:

- *Cambio Healthcare System*: Fundou-se em 1993, possui escritórios em Estocolmo, Reino Unido, Dinamarca e Sri Lank. O principal produto é o Cambio COSMIC, um sistema de RES centrado no paciente, que oferece soluções em todos setores de saúde para as organizações de saúde;
- *Code24 B.V*: Uma empresa holandesa, seu principal produto é o Base24 database suite, uma sofisticada solução de banco de dados projetada para armazenar, recuperar e disponibilizar dados de saúde;
- *Critical Software*: Nascida em Portugal, a empresa desenvolve soluções de *software* desde 1998, sua principal aplicação é o *Critical Health*;
- *DIPS ASA*: Uma das maiores fornecedoras de sistemas de prontuário eletrônico do paciente para hospitais da Noruega. Seu principal produto é o DIPS EPR systems, que é um RES que possibilita uma integração com ERP (*Enterprise Resource Planning*), HL7 (*Helth Level Seven*) e XML (*Extensible Markup Language*);
- *Infinity Solutions*: Empresa Russa que desenvolve soluções inovadoras no campo da saúde. A principal área de desenvolvimento é o RES integrado;

- *Marand*: Empresa eslovena líder na área de tecnologia de informação e comunicação e os processos de negócio de suporte de informação. Provê soluções centradas no cliente em saúde;
- *Ocean Informatics*: Fundada em 1998, é uma empresa formada por uma experiente equipe de engenharia clínica. Seus principais produtos são:
  - *Multiprac IC*: Solução de controle de infecção hospitalar;
  - *Multiprac SH*: Solução pessoal de saúde que fornece um registro de saúde pessoal para o pessoal da organização, englobando funcionalidades como: triagem de funcionários e gerenciamento contínuo incluindo vacinas e exposições;
  - *archetype Editor*: Uma ferramenta cliente usado por médicos especialistas para desenvolver arquétipos para o uso em ponto de atendimento e as configurações de pesquisa;
  - *Template Designer*: Ferramenta para a criação de templates openEHR de arquétipos através de *drag-and-drop*;
  - *ADL Workbench*: Ferramenta de código livre que fornece uma implementação de referência da definição de arquétipos em ADL 1.5 e OMA 1.5, para uso da comunidade openEHR.

2.2 TECNOLOGIAS UTILIZADAS

Durante o desenvolvimento do *software* proposto neste trabalho, várias tecnologias foram utilizadas contribuindo não só para a concretização deste trabalho, mas também com o nível de produtividade em relação ao desenvolvimento da aplicação proporcionando um controle maior na construção do código. Nas próximas seções, cada uma das seguintes tecnologias serão abordadas com maiores informações:

- IDE Visual Studio como ambiente de desenvolvimento;
- Plataforma de desenvolvimento ASP.NET;
- Banco de dados SQL Server.

2.2.1 VISUAL STUDIO

IDE funcional, proporciona uma nível alto de produtividade, permite a criação de qualquer aplicativo. Essa ferramenta possui banco de dados SQL Server integrado,

ambiente para testes e depuração, gerenciador de dependências, controle de versão, colaboração, entre outras funcionalidades (MICROSOFT, 2017b).

2.2.2 ASP.NET

ASP.NET é um framework *opensource* já integrado com a IDE Visual Studio, e tem como objetivo o desenvolvimento de aplicações *web* e serviços da *web*. É possível a criação de *websites* e sistemas de pequeno, médio e grande porte com HTML5, CSS e *javascript* (MICROSOFT, 2017a).

2.2.3 SQL SERVER

O SQL Server é um sistema gerenciador de banco de dados desenvolvido pela Microsoft que tem por função principal armazenar e recuperar dados requisitados por outras aplicações de *software*. Existem várias versões diferentes que atende necessidades específicas. Atende desde pequenas aplicações até aplicações de grande porte com milhões de acessos. Suas linguagens de consulta primárias são T-SQL e ANSI SQL (RESOURCE, 2017).

3 PROJETO DE SOFTWARE

3.1 REQUISITOS

Esta seção descreve as funcionalidades implementadas no software proposto. O levantamento dos requisitos foi baseado em relatos de profissionais da área da saúde, no qual o autor deste trabalho possui contato. Foi considerado também os conhecimentos adquiridos em disciplinas estudadas durante o curso de sistemas de informação como, por exemplo, o desenvolvimento integrado de sistemas, onde é possível construir aplicações capazes de gerir múltiplas interfaces de maneira integrada.

Para o protótipo proposto, foram considerados apenas três tipos de usuários: paciente, administrador e médico. Para o usuário do tipo paciente, a necessidade identificada foi a de acessar o histórico de consultas realizadas e a atualização de dados pessoais. O usuário do tipo médico necessita de uma área exclusiva com foco em gerenciar consultas médicas. Por fim, o usuário do tipo administrador precisa administrar os usuários, permitir acessos e gerenciar as funcionalidades do sistema.

Foi decidido construir duas interfaces, *web* e *desktop*, pelo fato de que alguns setores de hospitais e clínicas possuem suporte apenas para sistemas *desktop*. Para que o paciente possa consultar seus dados clínicos é necessário uma interface *web*, que possibilita o acesso em computadores e *smartphones*.

Os requisitos foram escritos em linguagem natural (texto) no padrão de formação de sentença, onde cada requisito funcional recebe um código gerado através das iniciais RF (Requisito funcional) seguido de um número e a sentença do requisito:

Exemplo: RF0001 - O software deverá listar os pacientes.

Para requisitos não funcionais, o código é gerado através das iniciais RNF (Requisito Não Funcional) seguido de um número e a sentença do requisito:

Exemplo: RNF001 - O campo de senha deverá ser do tipo texto.

3.1.1 REQUISITOS FUNCIONAIS

RF001 - O software deverá cadastrar um paciente.
RF002 - O software deverá cadastrar um médico.
RF003 - O software deverá ter um administrador cadastrado.
RF004 - O software deverá realizar login do paciente cadastrado.
RF005 - O software deverá realizar login do administrador cadastrado.
RF006 - O software deverá realizar login do médico cadastrado.
RF007 - O software deverá realizar logoff de um usuário autenticado.
RF008 - O software deverá ter um grupo de usuários cadastrado.
RF009 - O software deverá permitir listar pacientes.
RF0010 - O software deverá permitir listar médicos.
RF0011 - O software deverá permitir visualizar dados do paciente.
RF0012 - O software deverá permitir visualizar dados do médico.
RF0013 - O software deverá permitir alterar dados do paciente.
RF0014 - O software deverá permitir alterar dados do médico.
RF0015 - O software deverá permitir cadastrar consultas médicas.
RF0016 - O software deverá permitir listar consultas realizadas de um paciente.
RF0017 - O software deverá permitir listar consultas realizadas pelo médico.
RF0020 - O software deverá permitir listar funcionalidades do sistema.
RF0025 - O software deverá permitir alterar funcionalidade do sistema.
RF0030 - O software deverá permitir buscar consulta pelo nome do paciente.
RF0035 - O software deverá permitir visualizar consulta realizada.

3.1.2 REQUISITOS NÃO FUNCIONAIS

RNF001 - O paciente deverá logar-se pela interface web.
RNF002 - O administrador deverá logar-se pela interface web.
RNF003 - O médico deverá logar-se pela interface desktop.
RNF004 - As funcionalidades do sistema deverão ser gerenciadas pelo administrador.
RNF005 - A lista de médicos deverá ser exibida somente para o administrador.
RNF006 - A lista de pacientes deverá ser exibida para o administrador e o médico.
RNF007 - O paciente deverá visualizar as consultas realizadas pertencentes a ele.
RNF008 - O médico deverá visualizar as consultas realizadas por ele.

3.2 CASOS DE USO

3.2.1 ESPECIFICAÇÃO DETALHADA

De acordo com os requisitos descritos neste trabalho, faz-se necessário a elaboração de casos de uso para uma descrição detalhada do comportamento do sistema, bem como seu entendimento detalhado. A estrutura de documentação dos casos de uso inicia-se com o diagrama UML seguido de uma identificação única, o título, os atores envolvidos. Também faz parte da especificação, a descrição, as precondições caso seja necessário, os fluxos básico e alternativo que relata os passos a serem realizados no sistema e a referência dos requisitos cobertos pelo caso de uso.

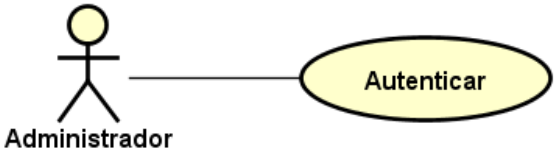


Figura 8: Caso de uso que representa a funcionalidade autenticação de usuário do tipo administrador.

ID:	US001
Caso de uso:	Autenticar Administrador
Atores:	Administrador, Sistema
Ator Primário:	Administrador
Descrição:	Funcionalidade para permitir acesso ao sistema.
Precondições:	O "Administrador"deve estar cadastrado e ativo no sistema.
Poscondições:	Permissão concedida para o administrador acessar o sistema. No apêndice F e G encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Administrador"deverá acessar o sistema utilizando um navegador web;
2. O "Administrador"deverá informar os dados de usuário e senha e selecionar o tipo de usuário "Administrador";
3. Requisitar o acesso ao sistema.

Referências: RF005, RNF002

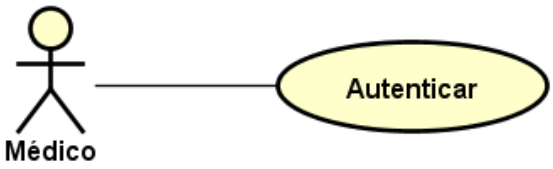


Figura 9: Caso de uso que representa a funcionalidade autenticação de usuário do tipo médico.

ID:	US002
Caso de uso:	Autenticar Médico
Atores:	Médico, Sistema
Ator Primário:	Médico
Descrição:	Funcionalidade para permitir acesso ao sistema.
Precondições:	O "Médico"deve estar cadastrado e ativo no sistema.
Poscondições:	Permissão concedida para o médico acessar o sistema. No apêndice A e B encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Médico" deverá acessar o sistema através de uma interface desktop;
2. O "Médico" deverá informar os dados de usuário e senha;
3. Requisitar o acesso ao sistema.

Referências: RF006, RNF003

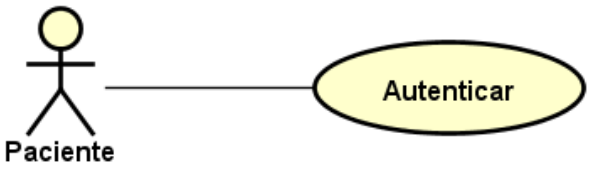


Figura 10: Caso de uso que representa a funcionalidade autenticação de usuário do tipo paciente.

ID:	US003
Caso de uso:	Autenticar Paciente
Atores:	Paciente, Sistema
Ator Primário:	Paciente
Descrição:	Funcionalidade para permitir acesso ao sistema.
Precondições:	O "Paciente" deve estar cadastrado e ativo no sistema.
Poscondições:	Permissão concedida para o paciente acessar o sistema. No apêndice F e H encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Paciente"deverá acessar o sistema através de um navegador web;
2. O "Paciente"deverá informar os dados de usuário e senha e selecionar o tipo de usuário "Paciente";
3. Requisitar o acesso ao sistema.

Referências: RF006, RNF001

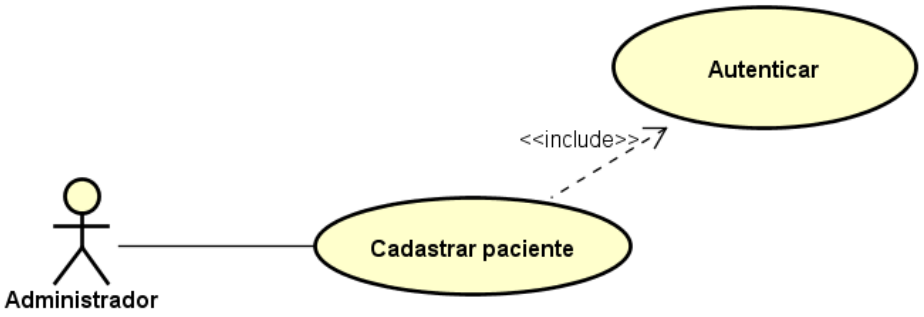


Figura 11: Caso de uso que representa a funcionalidade cadastro de pacientes.

ID: US004

Caso de uso: Cadastrar paciente

Atores: Administrador, Paciente

Ator Primário: Paciente

Descrição: Funcionalidade para a inclusão de pacientes no sistema.

Precondições: Um "Administrador"deverá estar cadastrado e ativo no sistema.

Poscondições: Cadastro do "Paciente"criado e ativo no sistema. No apêndice J encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Administrador"deverá acessar o sistema através de um navegador web;
2. O "Administrador"deverá logar-se no sistema;
3. O "Administrador"deverá acessar a sessão pacientes;
4. O "Administrador"deverá realizar o cadastro do "Paciente".

Referências: US001, RF001

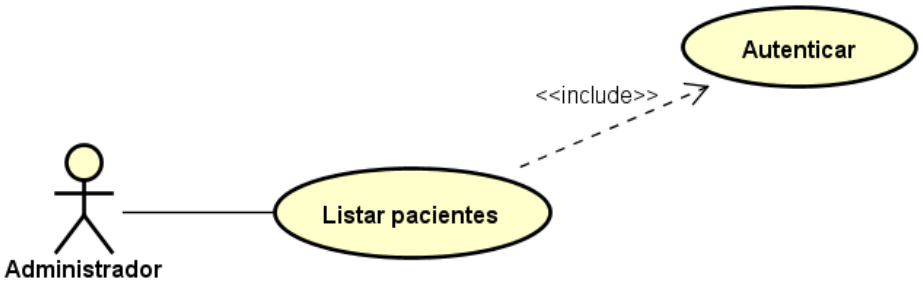


Figura 12: Caso de uso que representa a funcionalidade listar pacientes.

ID:	US005
Caso de uso:	Listar pacientes
Atores:	Administrador, Paciente, Sistema
Ator Primário:	Paciente
Descrição:	Funcionalidade para listar os pacientes existentes no sistema.
Precondições:	Deverá existir um "Administrador"cadastrado e ativo no sistema.
Poscondições:	Lista com pacientes cadastrados no sistema.

Fluxo básico:

1. O "Administrador"deverá acessar o sistema utilizando um navegador web;
2. O "Administrador"deve se logar no sistema;
3. O "Administrador"deverá acessar a sessão "pacientes";
4. A lista de pacientes deverá ser exibida.(A1)

Fluxo alternativo: **A1** Sistema sem registro de pacientes

1. O "Sistema" deverá exibir uma mensagem explicativa para o "Administrador" informando que não há registro de pacientes no momento.

Referências: US001, RF009, RNF006

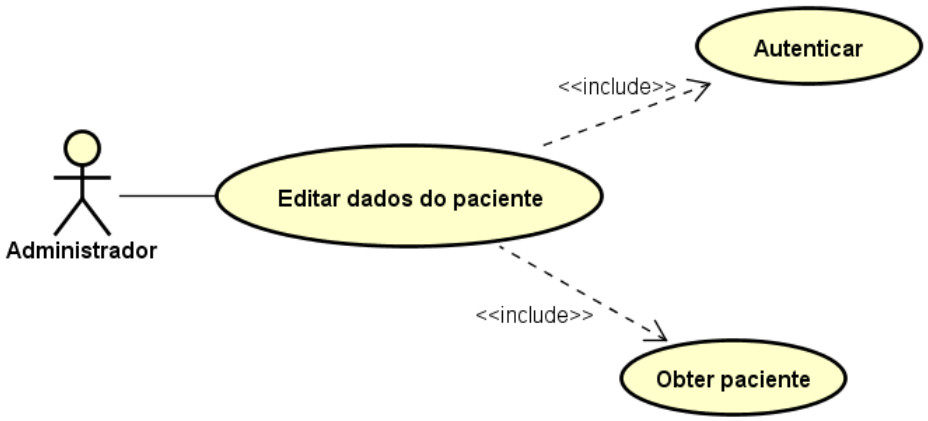


Figura 13: Caso de uso que representa a funcionalidade editar dados do paciente.

ID: US006

Caso de uso: Editar dados do paciente

Atores: Paciente, Administrador

Ator Primário: Administrador

Descrição: Funcionalidade para editar dados do paciente cadastrado no sistema.

Precondições:

1. Deverá existir um "Administrador" ativo no sistema;
2. O "Paciente" deverá estar cadastrado e ativo no sistema.

Poscondições: Cadastro do "Paciente"alterado.

Fluxo básico:

1. O "Administrador"deverá acessar o sistema através do ambiente web;
2. O "Administrador"deverá logar-se no sistema;
3. O "Administrador"deverá acessar a sessão de pacientes;
4. O "Administrador"deverá escolher o "Paciente"(A1);
5. O "Administrador"deverá editar os dados do "Paciente".

Fluxo alternativo: **A1** Sistema sem registro de pacientes

1. O "Sistema"deverá exibir uma mensagem explicativa para o "Administrador"informando que não há registro de pacientes no momento.

Referências: US001, US005, RF0013

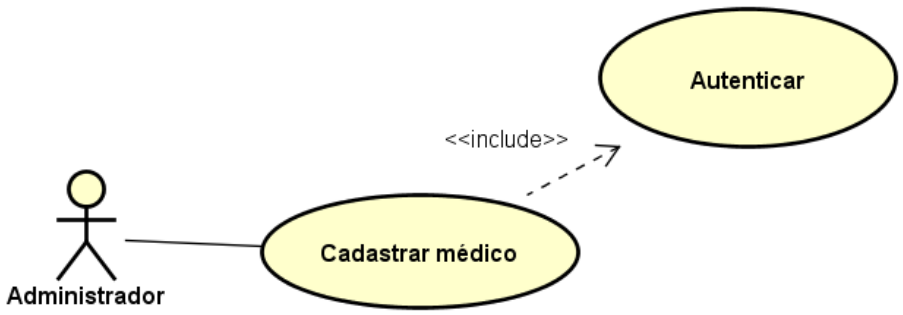


Figura 14: Caso de uso que representa a funcionalidade cadastrar médico.

ID: US007

Caso de uso: Cadastrar médico

Atores: Médico, Administrador

Ator Primário: Administrador

Descrição: Funcionalidade para cadastrar médicos no sistema.

Precondições: Deverá existir um "Administrador"cadastrado e ativo no sistema.

Poscondições: Cadastro do "Médico"realizado.

Fluxo básico:

1. O "Administrador"deverá acessar o sistema através de um ambiente web;
2. O "Administrador"deverá logar-se no sistema;
3. O "Administrador"deverá acessar a sessão de médicos;
4. O "Administrador"deverá informar os dados do "Médico"a ser cadastrado;
5. O "Administrador"deverá realizar o cadastro do "Médico".

Referências: US001, RF002, RNF005

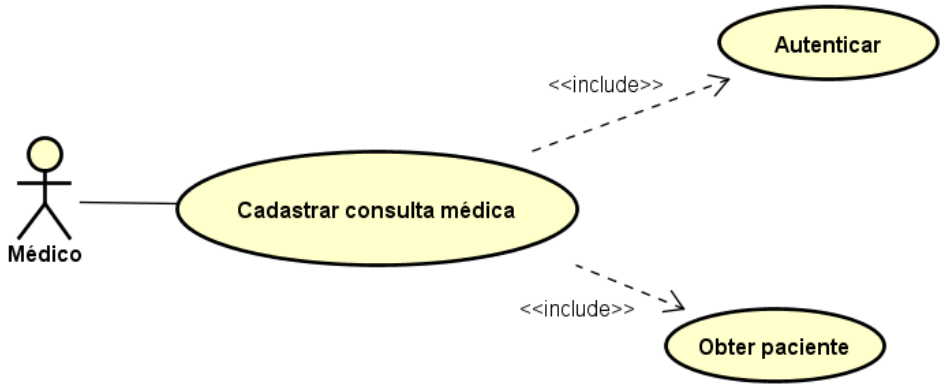


Figura 15: Caso de uso que representa a funcionalidade cadastrar consulta médica.

ID:	US008
Caso de uso:	Cadastrar consulta médica
Atores:	Médico, Paciente, Sistema
Ator Primário:	Médico
Descrição:	Funcionalidade que realiza uma consulta médica.
Precondições:	Deverá existir um "Médico"e um "Paciente"cadastrado e ativo no sistema.
Poscondições:	Consulta registrada no sistema. No apêndice C encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Médico" deverá acessar o "Sistema" através de uma interface desktop;
2. O "Médico" deverá logar-se no sistema;
3. O "Médico" deverá acessar a sessão correspondente ao propósito da consulta. Ex: medir dados vitais, medir pressão arterial, etc.
4. O "Médico" deverá escolher o "Paciente";
5. O "Médico" deverá preencher o formulário da consulta e após, gravá-los.

Referências: US002, RF0015, RNF003, RNF006

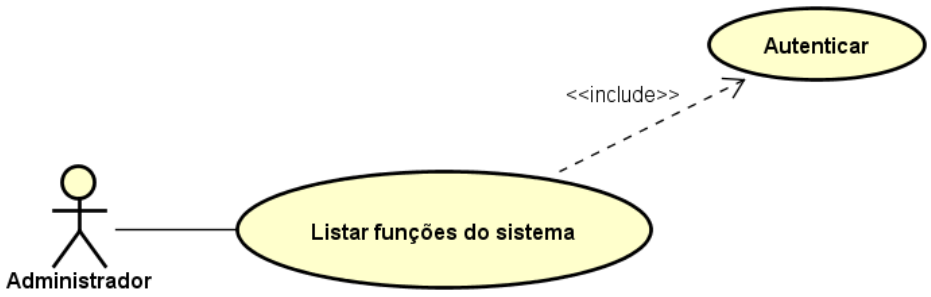


Figura 16: Caso de uso que representa a funcionalidade listar funções do sistema.

ID: US009

Caso de uso: Listar funções do sistema

Atores: Administrador, Sistema

Ator Primário: Administrador

Descrição: Lista de funcionalidades cadastradas no sistema.

Precondições:

1. O “Administrador” deverá estar cadastrado e ativo no sistema;

Poscondições: Função do sistema alterada. No apêndice I encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O “Administrador” deverá acessar o “Sistema” através de um navegador web;
2. O “Administrador” deverá logar-se no sistema;
3. O “Administrador” deverá acessar a sessão de funcionalidades do sistema;
4. O “Sistema” deverá listar as funcionalidades cadastradas.

Referências: US001, RF0020, RNF002, RNF004

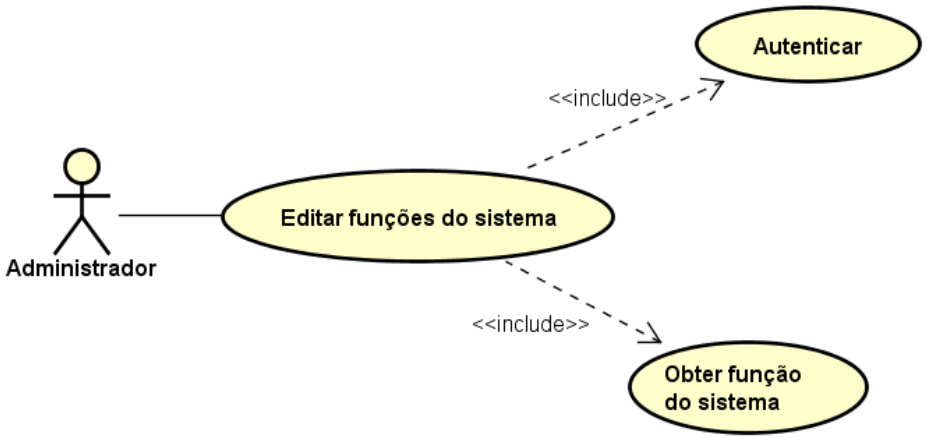


Figura 17: Caso de uso que representa a funcionalidade editar funções do sistema.

ID: US010

Caso de uso: Editar funções do sistema

Atores: Administrador, Sistema

Ator Primário: Administrador

Descrição: Funcionalidade que permite ativar/desativar funções no sistema, exibir ou não no menu, alterar permissão de acesso e edição de nome e descrição.

Precondições:

1. O "Administrador"deverá estar cadastrado e ativo no sistema;

Poscondições: Função do sistema alterada.

Fluxo básico:

1. O "Administrador"deverá acessar o "Sistema"através de um navegador web;
2. O "Administrador"deverá logar-se no sistema;
3. O "Administrador"deverá acessar a sessão de funcionalidades do sistema;
4. O "Administrador"deverá selecionar a função desejada para edição;
5. O "Administrador"poderá realizar as alterações necessárias.

Referências: US001, RF0025, RNF002, RNF004

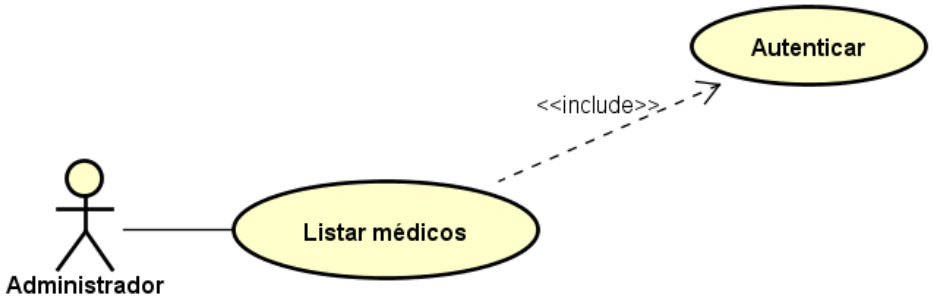


Figura 18: Caso de uso que representa a funcionalidade listar médicos no sistema.

ID:	US011
Caso de uso:	Listar médicos no sistema
Atores:	Administrador, Médico, Sistema
Ator Primário:	Médico
Descrição:	Funcionalidade que exibe uma lista os médicos cadastrados no sistema.

Precondições:

1. Deverá existir um "Administrador"cadastrado e ativo no sistema;

Poscondições: Lista com médicos cadastrados no sistema.

Fluxo básico:

1. O "Administrador"deverá acessar o "Sistema"através de um navegador web;
2. O "Administrador"deverá logar-se no sistema;
3. O "Administrador"deverá acessar a sessão "Médicos";
4. O "Sistema"deverá listar os médicos cadastrados. (A1)

Fluxo alternativo: **A1** Sistema sem registro de médicos

1. O "Sistema" deverá exibir uma mensagem explicativa para o "Administrador" informando que não há registro de "Médicos" no momento.

Referências: US001, RF010, RNF005

Figura 19: Caso de uso que representa a funcionalidade listar histórico de consultas realizadas do paciente.

ID: US012

Caso de uso: Listar histórico de consultas realizadas do paciente

Atores: Paciente, Sistema

Ator Primário: Paciente

Descrição: Funcionalidade que exibe o histórico de consultas de um paciente no sistema.

Precondições:

1. Deverá existir um "Paciente" cadastrado e ativo no sistema;

Poscondições: Lista de consultas realizadas do paciente no sistema. No apêndice K encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O “Paciente”deverá acessar o “Sistema”através de um navegador web;
2. O “Paciente”deverá logar-se no sistema;
3. O “Paciente”deverá acessar a sessão ”Histórico de consultas”(A1);
4. O “Sistema”deverá listar as consultas do paciente já realizadas (A2).

Fluxo alternativo: **A1** Acesso ao histórico de consultas através da sessão de perfil do usuário

1. O “Paciente”deverá acessar a área do usuário no menu;
2. O “Paciente”deverá acessar o link “Meus procedimentos”;

Fluxo alternativo: **A2** Sistema sem registro de consultas realizadas

1. O “Sistema”deverá exibir uma mensagem explicativa para o “Paciente”informando que não há registro de consultas realizadas no momento.

Referências: US003, RF0016, RNF001, RNF007

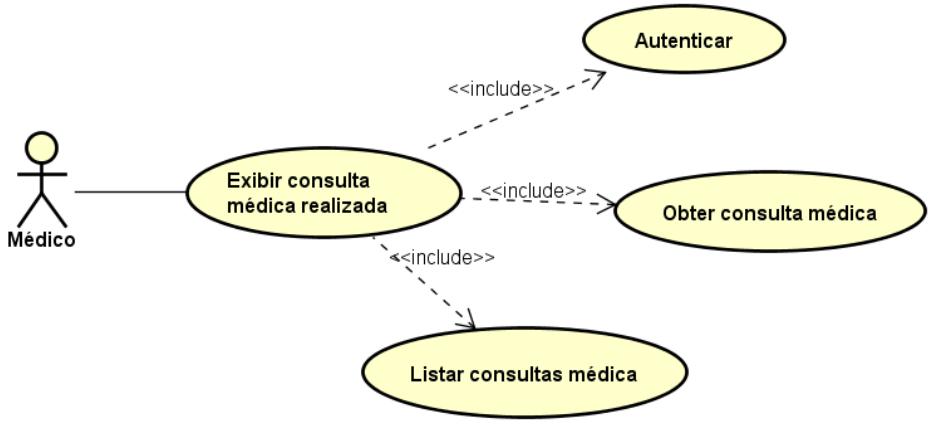


Figura 20: Caso de uso que representa a funcionalidade exibir consulta realizada pelo médico.

ID:	US013
Caso de uso:	Exibir consulta médica realizada
Atores:	Médico, Paciente
Ator Primário:	Médico
Descrição:	Funcionalidade que exibe uma consulta realizada pelo médico.
Precondições:	1. Deverá existir um "Médico"cadastrado e ativo no sistema;
Poscondições:	Exibição da consulta realizada pelo médico. No apêndice D e E encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Médico" deverá acessar o "Sistema" através da aplicação desktop;
2. O "Médico" deverá logar-se no sistema;
3. O "Médico" deverá acessar a sessão "Minhas consultas/procedimentos";
4. O "Sistema" deverá exibir a lista de consultas realizadas (A1);
5. O "Médico" deverá escolher uma consulta para exibição e clicar em "Visualizar" (A2);
6. O "Sistema" deverá exibir uma janela com os dados da consulta.

Fluxo alternativo: **A1** Sistema sem registro de consultas realizadas

1. O "Sistema" deverá exibir uma mensagem explicativa para o "Médico" informando que não há registro de consultas realizadas no momento.

Fluxo alternativo: **A2** Buscar consulta pelo nome do paciente

1. O "Médico" deverá informar na caixa de busca o nome do paciente;
2. O "Sistema" deverá exibir a lista de consultas realizadas do paciente informado no campo de busca.

Referências: US002, RF0017, RF0030, RF0035, RNF003, RNF008

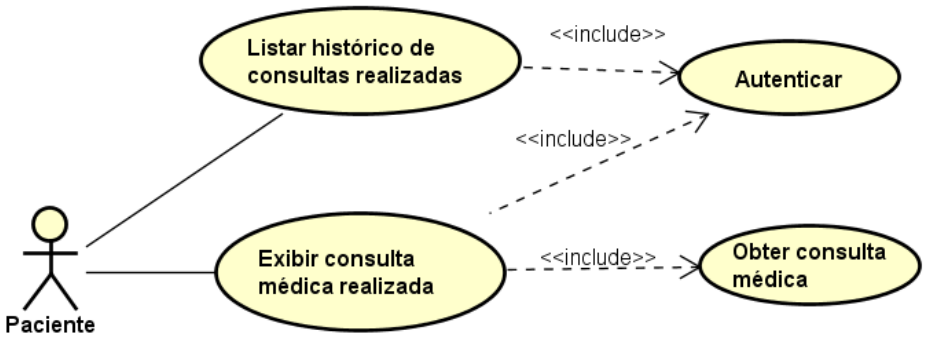


Figura 21: Caso de uso que representa a funcionalidade exibir consulta do paciente realizada.

ID:	US014
Caso de uso:	Exibir consulta do paciente realizada
Atores:	Paciente, Sistema
Ator Primário:	Paciente
Descrição:	Funcionalidade que exibe os dados de uma consulta realizada do "Paciente".
Precondições:	1. Deverá existir um "Paciente"cadastrado e ativo no sistema;
Poscondições:	Exibição dos dados de uma consulta realizada. No apêndice L encontra-se a figura do protótipo relacionado a este caso de uso.

Fluxo básico:

1. O "Paciente" deverá acessar o "Sistema" através de um navegador web;
2. O "Paciente" deverá logar-se no sistema;
3. O "Paciente" deverá acessar a sessão "Histórico de consultas";
4. O "Sistema" deverá listar as consultas do paciente já realizadas (A1);
5. O "Paciente" deverá escolher uma consulta para visualizar os dados.
6. O "Sistema" deverá exibir os dados da consulta escolhida.

Fluxo alternativo: **A1** Sistema sem registro de consultas realizadas

1. O "Sistema" deverá exibir uma mensagem explicativa para o "Paciente" informando que não há registro de consultas realizadas no momento.

Referências: US003, US012, RF0035, RNF001, RNF007

3.3 DIAGRAMA DE CLASSES

A modelagem de classes é uma das principais etapas no processo de desenvolvimento de *software*, é neste processo que são definidos os relacionamentos e comportamentos entre os recursos de uma aplicação. O diagrama de classes tem por finalidade modelar os recursos para montagem e operação de um sistema e, a partir da estrutura de classes, pode-se ter uma ampla visão do todo (PENDER, 2004).

Os diagramas de classe desenvolvido para o *software* proposto são apresentados nos apêndices M e N. No apêndice M está representado o diagrama da camada de acesso a dados. As classes de acesso a dados podem ser instanciadas em toda a aplicação, mas o objetivo não é realizar consultas ou inserir registros no banco de dados utilizando diretamente suas instâncias, visto que esta habilidade compete somente à camada de domínio da aplicação.

No apêndice N está modelado o diagrama de classes da camada de domínio do sistema, classes na qual não podem ser instanciadas por conter membros estáticos, pois o propósito da camada de domínio é comportamental. Para obter uma lista de pacientes, é preciso requisitar o método da camada de domínio *getPatients()*, que por sua vez instancia a classe *Patient* e realiza a lógica para obter os dados em uma lista e retornar para quem a requisitou.

3.4 MODELO ENTIDADE RELACIONAMENTO

Os diagramas de entidade e relacionamento auxiliam na modelagem da representação de dados de um *software*. É um modelo em alto nível e, nele, é representado as entidades e os relacionamentos entre os objetos. Se tratando de projetos para pequenas aplicações, é viável para o projetista decidir as relações a serem criadas, seus atributos e suas relações, entretanto, para projetos de médio e grande porte, é necessário a interação entre várias pessoas envolvidas na aplicação afim de obter as necessidades reais dos usuários para que possa ser traduzida para a estrutura conceitual (SILBERSCHATZ et al., 2012).

No apêndice O encontra-se a modelagem de entidade e relacionamento para a aplicação proposta. Foi aplicado os conceitos referentes ao openEHR nas entidades *patientRecord*, *TemplateAttribute*, *DataListAttribute*, *Data* e *OperationalsTemplate*. Entidades que trabalham de forma dinâmica, onde o *template* de um domínio clínico pode ser

cadastrado, bem como seus atributos, tipos (texto, lista, inteiros, entre outros) e lista de dados sem a necessidade de criar uma entidade para cada *template*. Podem ser persistidas no banco de dados informações de consultas médicas utilizando qualquer domínio clínico disponível no sistema.

3.5 RECURSOS UTILIZADOS

São descritos neste capítulo todos recursos de *hardware* e *software* que tornaram viáveis o desenvolvimento deste projeto.

3.5.1 RECURSOS DE HARDWARE

Para o desenvolvimento da aplicação proposta nesse trabalho foi utilizado um Notebook de alto desempenho com sistema operacional Windows 8.1, 8GB de memória, 1TB de HDD, devidamente licenciado de propriedade particular do autor. Também foi necessário conexão com a internet para que fosse possível realizar este trabalho.

3.5.2 RECURSOS DE SOFTWARE

Alguns *softwares* tiveram importância vital para a conclusão da parte prática deste trabalho, ressaltando que todos os recursos de *software* utilizados que exigem licença em relação ao seu uso estão devidamente licenciados. Os *softwares* utilizados foram:

- IDE para desenvolvimento de *software* Visual Stúdio Enterprise 2015;
- Linguagem de programação C#;
- Banco de dados SQL Server para desenvolvimento local.

4 DISCUSSÃO

Este trabalho buscou satisfazer não somente os objetivos descritos neste documento mas também a dificuldade de manutenção e atualização da base do conhecimento médico de um *software* voltado para a área da saúde. Outra demanda importante satisfeita pelo programa desenvolvido é o acompanhamento do histórico de consultas realizadas pelo paciente, sendo possível a visualização detalhada dos procedimentos realizados como a data da consulta, o médico que o atendeu e também outras particularidades previstas na consulta.

As funcionalidades presentes no *software* são baseadas em demandas reais de clínicas médicas, sendo possível, futuramente, adicionar outras funções para o enriquecimento do mesmo. O *software* desenvolvido atendeu aos requisitos definidos neste trabalho, implementado com as mais novas tecnologias para o desenvolvimento de sistemas utilizadas na indústria e seguindo as boas práticas no desenvolvimento de aplicações, projetando-se para as demandas futuras.

4.1 OPENEHR

O padrão openEHR veio para solucionar problemas de atualização de *software*, padronizar atividades e/ou procedimentos médicos e também prover a interoperabilidade entre sistemas. Na seção 3.4 mostra como se deu a aplicação do openEHR ilustrando como os profissionais de desenvolvimento de aplicações focam na implementação das funcionalidades e não em modelar regras específicas de domínio, o que contribui para um *software* mais conciso e coerente às demandas de negócio.

Para que a interoperabilidade entre sistemas e a padronização de procedimentos médicos aconteça, se faz necessário ir além do desenvolvimento de aplicações. É preciso que a adoção do openEHR aconteça partindo de médicos, enfermeiros, profissionais de saúde em geral e também das áreas gerenciais do setor para organizar às demandas e também os recursos tangíveis e intangíveis envolvidos.

4.1.1 VANTAGENS E DESVANTAGENS

Além de prover a interoperabilidade entre sistemas, o openEHR possui outras vantagens como:

- Diminuição do tempo de desenvolvimento, pois as regras de domínio estão no modelo;
- A padronização de arquétipos que asseguram a interoperabilidade semântica dos registros eletrônicos de saúde;
- Torna os sistemas de informação flexíveis para serem adaptados aos processos;
- Disponibilizar o software em vários idiomas, visto que os arquétipos podem ser traduzidos em diversas línguas;
- Evita a duplicação de dados de pacientes;

Entretanto, o padrão possui suas desvantagens:

- Documentação escassa e não muito detalhada em relação ao desenvolvimento de *software*;
- A complexidade dos dados de saúde faz com que haja uma latência maior na adequação de dados para seu uso;
- Baixa difusão do padrão proposto (CESAR, 2013);
- No Brasil cerca de 20% a 25% dos hospitais possuem algum tipo de sistema de informação (BACELAR; CORREIA, 2015);
- Repositório de arquétipos e *templates* ainda estão em desenvolvimento no Brasil.

4.1.2 USO DO MVC

Ao utilizar o padrão MVC no sistema desenvolvido, possibilita a expansão e a manutenabilidade do código. A camada de modelo foi subdividida na camada de domínio, onde ficam as regras lógicas do sistema, e na camada de acesso a dados, onde são realizadas as operações no banco de dados como a inserção de registros, alteração, entre outras funções. As classes de modelo são comuns para todo o *software*.

Estruturou-se a camada de visualização para suportar diversos tipos de interfaces, e, embora a estrutura atual tenha suporte para *desktop* e *web*, é possível a expandir para *mobile*, *restful web services* e outros. Para a camada de controle, foi adotada uma estratégia diferente para que o sistema fosse altamente escalável, ao invés de termos a camada de controle tradicional, foi decidido que cada interface tivesse seu próprio controle, pois assim é possível adicionar diversos tipos de interface com suas particularidades de controle e usufruir apenas de um modelo.

5 CONCLUSÃO

Ao desenvolver este trabalho, procurou-se relacionar assuntos estudados durante o curso de graduação como engenharia de *software*, desenvolvimento integrado de sistemas e definições de linguagem de programação, conceitos com os quais tornou-se possível o desenvolvimento do atual estudo. Este trabalho resultou na construção de um *software* de gerenciamento de consultas médicas, possibilitando adicionar novas funcionalidades de domínio clínico sem a necessidade de realizar alterações no código, possibilitando também a transparência em relação ao acompanhamento do histórico de consultas dos pacientes.

Apesar de atingidos os objetivos descrito neste projeto, alguns pontos relevantes servem de incentivo para trabalhos futuros na tentativa de evolução tanto das técnicas e dos padrões aplicados quanto do artefato criado para o objetivo proposto. Ao aplicar o padrão openEHR no desenvolvimento do sistema, foi possível identificar a importância da criação de modelos, manter a propriedade dos dados com o paciente e a possibilidade de compartilhar dados mais precisos.

5.1 CONTRIBUIÇÕES

Este trabalho contribui de forma significativa no desenvolvimento de *software* para a área de saúde, não só pelas técnicas empregadas no projeto, mas também por contribuir para a difusão do openEHR. A contribuição deste estudo também é relevante para a engenharia de *software*, pois é necessário uma mudança na forma de projetar aplicações, dando prioridade na facilidade de atualização e na livre comunicação entre sistemas.

5.2 TRABALHOS FUTUROS

Pensando em uma evolução e melhoria contínua, este trabalho tem abertura para aplicação de novas técnicas, além de poder ser utilizado como base para outros projetos

relacionados ao openEHR ou até mesmo à arquitetura de *software*. Os itens elencados são:

- Automatizar a inclusão de novas funcionalidades de domínio clínico;
- Estudo e aplicação de técnicas relacionados a banco de dados não relacionais;
- Adição de interface *restful webservices*;
- Adição de interface *mobile*;
- Implementação da funcionalidade de agendamento de consultas.

REFERÊNCIAS

AGUILAR, A. Semantic interoperability in the context of ehealth. In: CITESEER. **Research Seminar, DERI Galway, December 15th**. [S.l.], 2005.

ANDRADE, R. M. d.; ARAKAKI, R.; BECERRA, J. L. R. O uso de provas de conceito como ferramenta para gestÃo de aprendizado de arquitetura de software. **3º Congresso Internacional de Gestão da Tecnologia e Sistemas de Informação**, 2006.

BACELAR, G.; CORREIA, R. **As bases do openEHR**. [S.l.]: VirtualCare, 2015.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. [S.l.]: Elsevier Brasil, 2006.

CARVALHO, A. G. d. Uma proposta de arquitetura de software genérica para mobile games utilizando j2me. **SBGames'2006 - V Brazilian Symposium on Computer Games and Digital Entertainment**, 2006.

CESAR, H. V. Registro eletrônico de saúde: dos moldes tradicionais 'a padronização proposta pela fundação openehr. 2013. Disponível em: <http://www.senaicimatec.com.br/wp-content/uploads/2017/03/Disserta%C3%A7%C3%A3o_Hilton_Vicente_Cesar.pdf>. Acesso em: 04 de Novembro de 2017.

CORREIA, R. Uma norma aberta para a criação de registos de saúde eletrónicos (ehr). **eSaúde - Magazine dos Sistemas de Informação na Saúde**, 2012.

DALL'OGLIO, P. **PHP - PROGRAMANDO COM ORIENTAÇAO A OBJETOS**. [S.l.]: NOVATEC, 2009. ISBN 9788575222003.

FOUNDATION, C. S. **CakePHP 1.2 Cookbook**. [s.n.], 2017. Disponível em: <https://book.cakephp.org/1.2/pt/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html>. Acesso em: 15 de Novembro de 2017.

JUNIOR, A. M. Métodos de formação de preço de venda em sistemas erp por intermédio de arquitetura orientada à serviços do framework framemk. 2013. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/635>.

LARMAN, C. **Utilizando UML e padrões**. [S.l.]: Bookman Editora, 2002.

MAIA, T. A. Processo de desenvolvimento de arquétipos do registro eletrônico em saúde em minas gerais: Estudo de caso. Belo Horizonte, MG, Brasil, 2014.

MEZAROBA, W. F.; NICOLEIT, E. R. Protótipo para representação de informações demográficas em um s-res baseado nas especificações openehr. v. 5, Sep 2010. ISSN 2359-2656. Disponível em: <http://periodicos.unesc.net/sulcomp/article/view/264/270>.

MICROSOFT. Asp.net. 2017. Disponível em: <https://www.asp.net/>. Acesso em: 02 de Novembro de 2017.

MICROSOFT. Ide do visual studio. 2017. Disponível em: <https://www.visualstudio.com/pt-br/vs/>. Acesso em: 02 de Novembro de 2017.

NAKAGAWA, E. Y. **Uma contribuição ao projeto arquitetural de ambientes de engenharia de software**. Tese (Doutorado) — Universidade de São Paulo, 2006.

OPENEHR, F. Academic research. 2017. Disponível em: <http://openehr.org/who_is_using_openehr/academic_research>. Acesso em: 13 de Maio de 2017.

OPENEHR, F. openehr templates. 2017. Disponível em: <http://openehr.org/releases/trunk/architecture/am/tom.pdf>. Acesso em: 13 de Maio de 2017.

PENDER, T. **Uml - A Biblia**. [S.l.]: Campus Editora, 2004.

PESSANHA, C. P.; BAX, M. P. Implementando o prontuário eletrônico openehr em sistemas gestores de conteúdo: Uma aproximação. **XVI Encontro Nacional de Pesquisa em Ciência da Informação**, João Pessoa, PB, Brasil, 2015. Disponível em: <http://www.ufpb.br/evento/lti/ocs/index.php/enancib2015/enancib2015/paper/viewFile/2756/1277>. Acesso em: 9 de maio de 2016.

PESSANHA, C. P.; BAX, M. P. Codificando arquétipos em linguagem adl com base no modelo de referência da norma iso 13606. **XVI Encontro Nacional de Pesquisa em Ciência da Informação**, MG, Brasil, 2016. Disponível em: <http://sres.saude.mg.gov.br/upload/manual/PaperADL-SESMG.pdf>. Acesso em: 29 de maio de 2016.

RESOURCE, D. D. Sql server overview. 2017. Disponível em: <http://www.databasedesign-resource.com/sql-server-overview.html>. Acesso em: 02 de Novembro de 2017.

RODRÍGUEZ, L. M. G.; NAKAGAWA, E. Y. **Arquitetura de Software**. Tese (Doutorado) — Universidade de São Paulo, 2015.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de bancos de dados**. [S.l.]: Elsevier, 2012. 155, 157, 188–189 p. ISBN 9788535245356.

SILVA, F. M. da. **SOA - Arquitetura Orientada a Serviços**. 2006. Disponível em: <https://www.linux.ime.usp.br/ cef/mac499-06/monografias/filipemadeira/monografia.pdf>. Acesso em: 7 de fevereiro de 2017.

SILVA, V. M. da. Revisão sistemática da evolução mvc na base acm. 2012. Disponível em: <https://goo.gl/lvK0EO>.

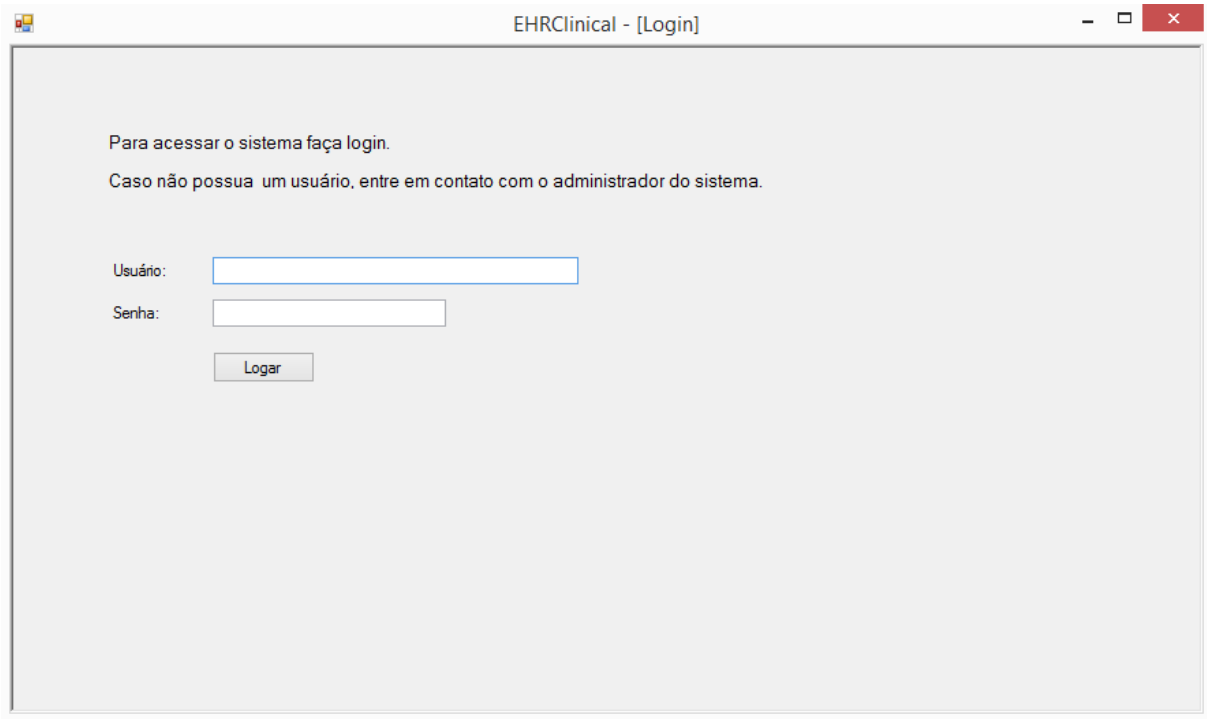
SOUZA, D. A. B. d. Um panorama no uso de openehr. Recife, PB, Brasil, 2014. Disponível em: <http://tcc.ecomp.poli.br/20142/Denise%20Assis.pdf>.

SUGAI, R. Registro eletrônico de saúde. **Planejamento Estratégico do MS para 2008-2011**, 2008.

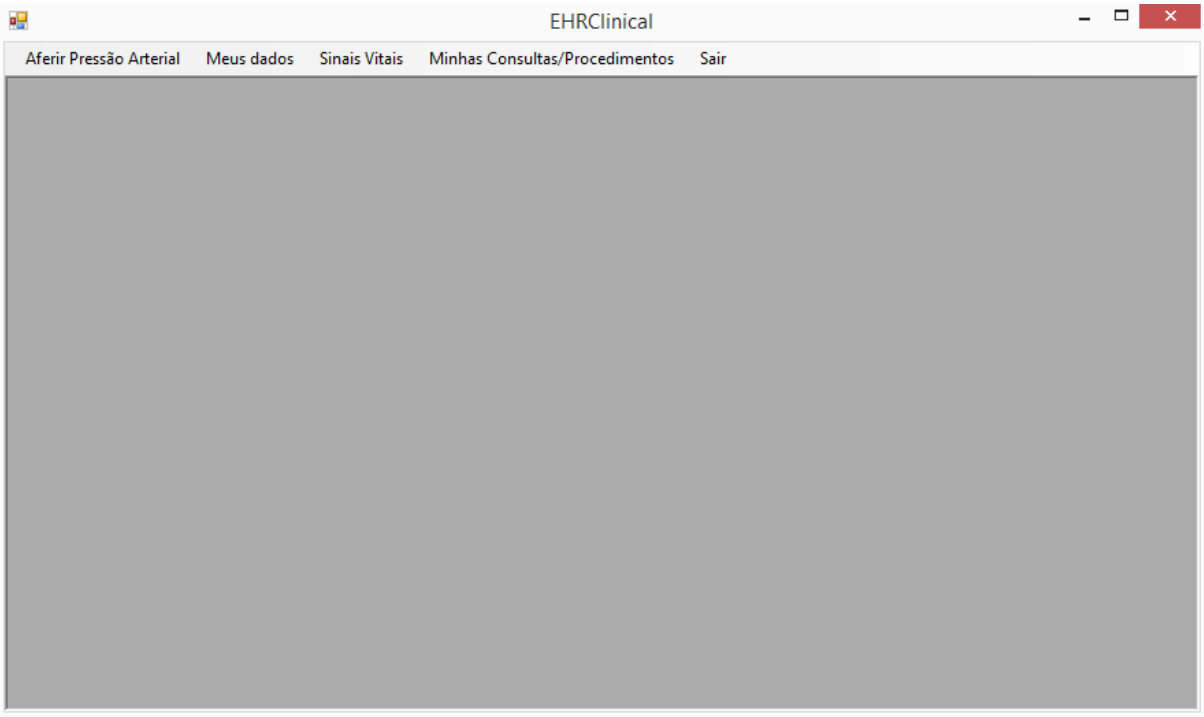
VICENTINI, C. F. et al. Pehs – arquitetura de um sistema de informação pervasivo para auxílio às atividades clínicas. **Revista Brasileira de Computação Aplicada**, Passo Fundo, RS, Brasil, 2010. Disponível em: <http://www.upf.br/seer/index.php/rbca/article/view/970/783>. Acesso em: 9 de maio de 2016.

ZAMBIASI, S. P. et al. Uma arquitetura de referência para softwares assistentes pessoais baseada na arquitetura orientada a serviços. Florianópolis, 2012.

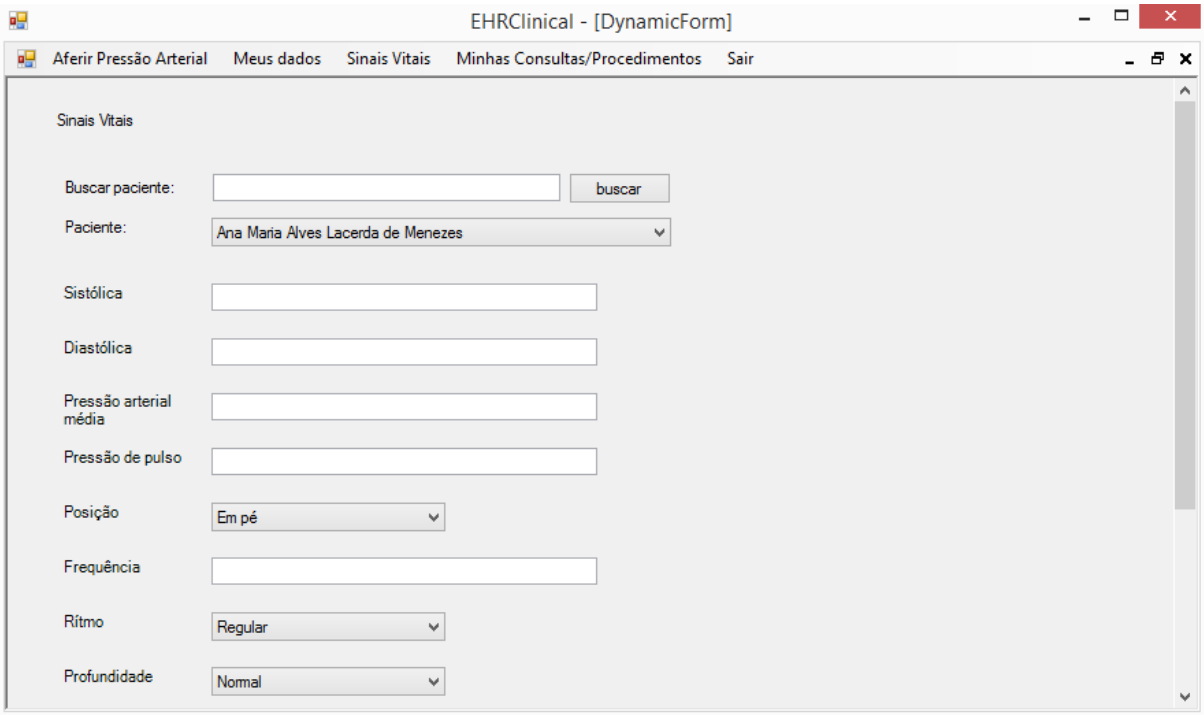
APÊNDICE A – TELA DE LOGIN DA ÁREA MÉDICA



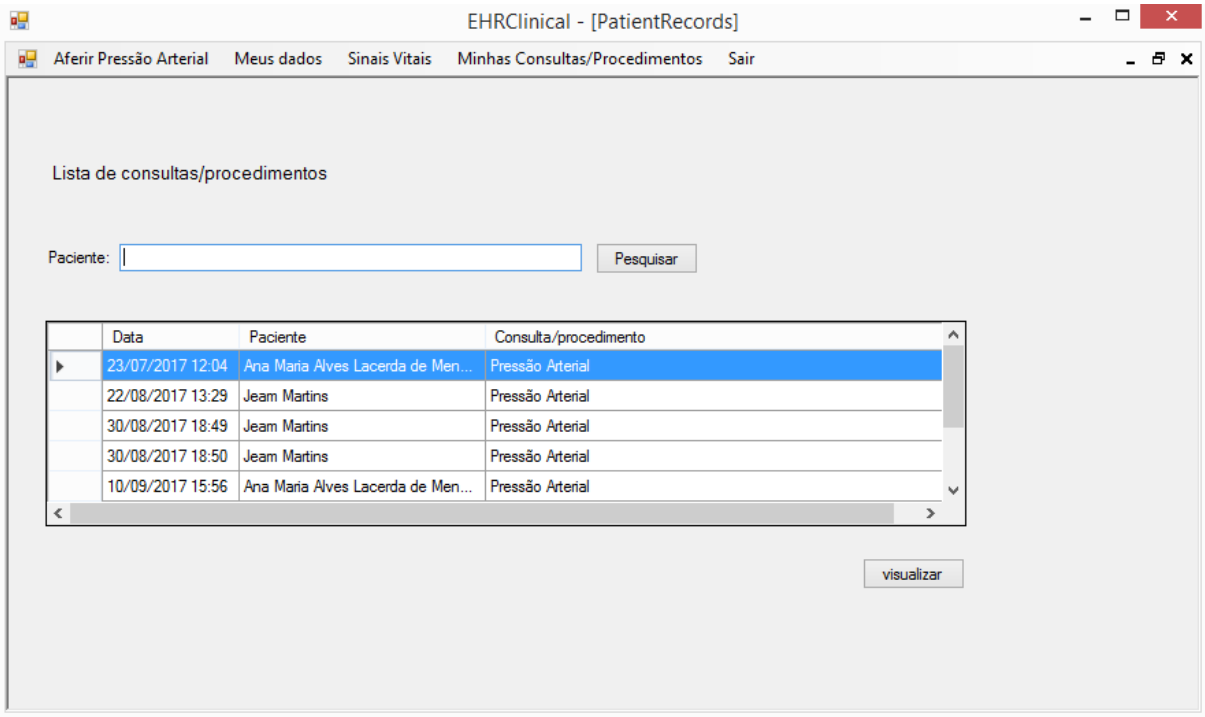
APÊNDICE B – TELA INICIAL DA ÁREA MÉDICA



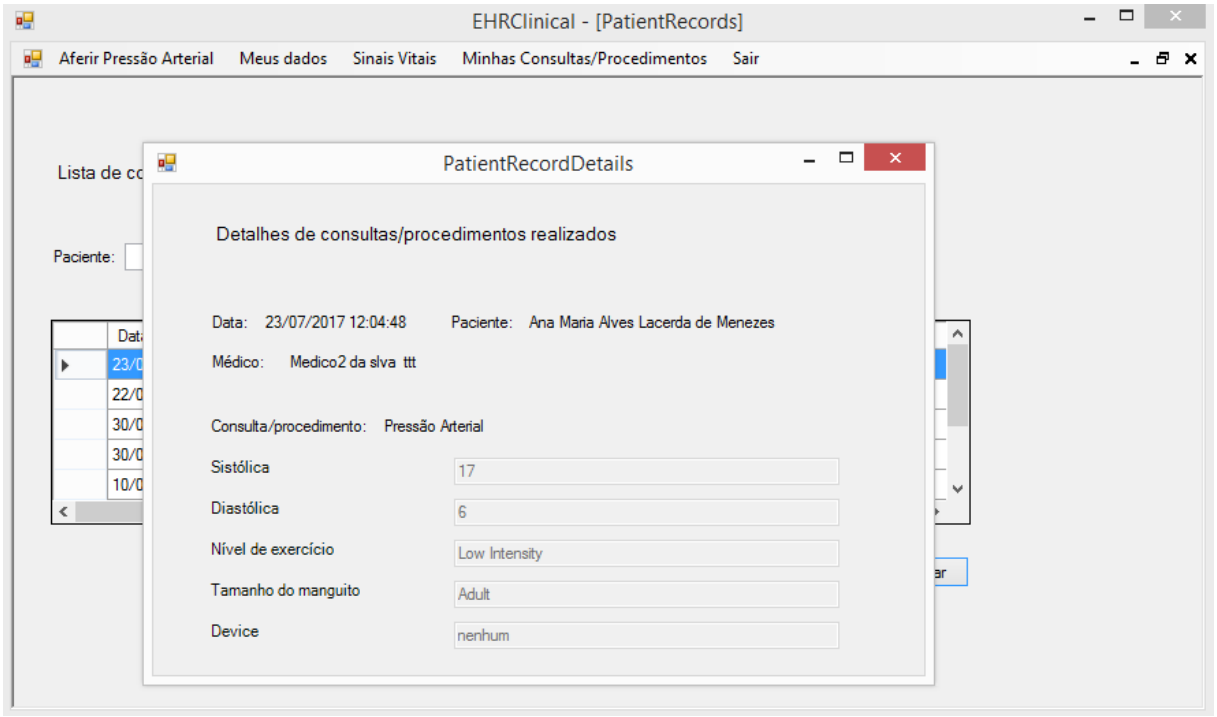
APÊNDICE C – FORMULÁRIO DE CONSULTA MÉDICA



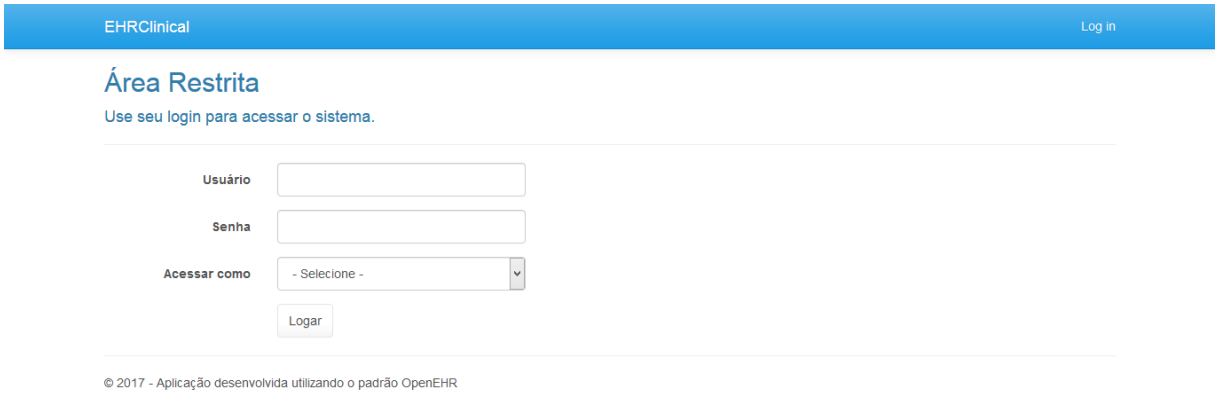
APÊNDICE D – LISTA DE CONSULTAS DE UM MÉDICO



APÊNDICE E – VISUALIZAÇÃO DOS DADOS DE UMA CONSULTA MÉDICA



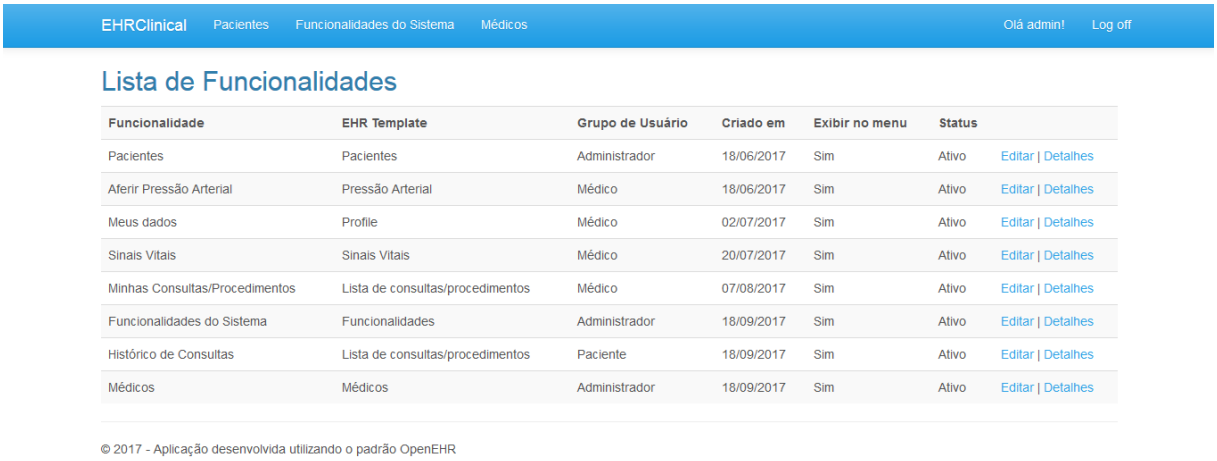
APÊNDICE F – ÁREA DE LOGIN DO ADMINISTRADOR E DO PACIENTE



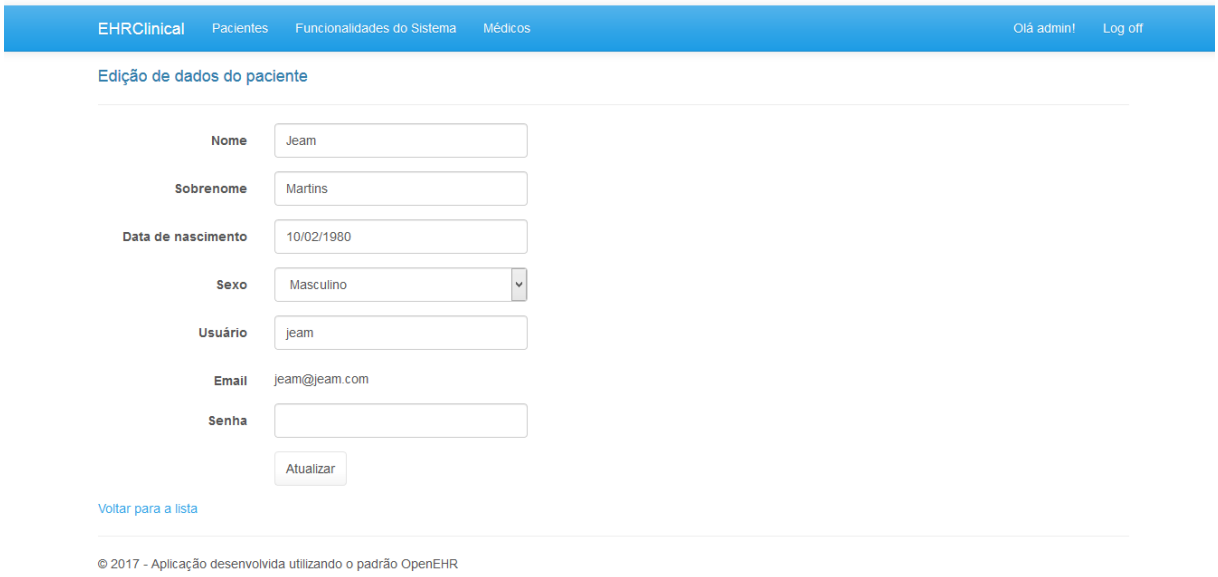
APÊNDICE G – PÁGINA INICIAL DA ÁREA ADMINISTRATIVA

APÊNDICE H – PÁGINA INICIAL DA ÁREA DO PACIENTE

APÊNDICE I – LISTA DE FUNCIONALIDADES

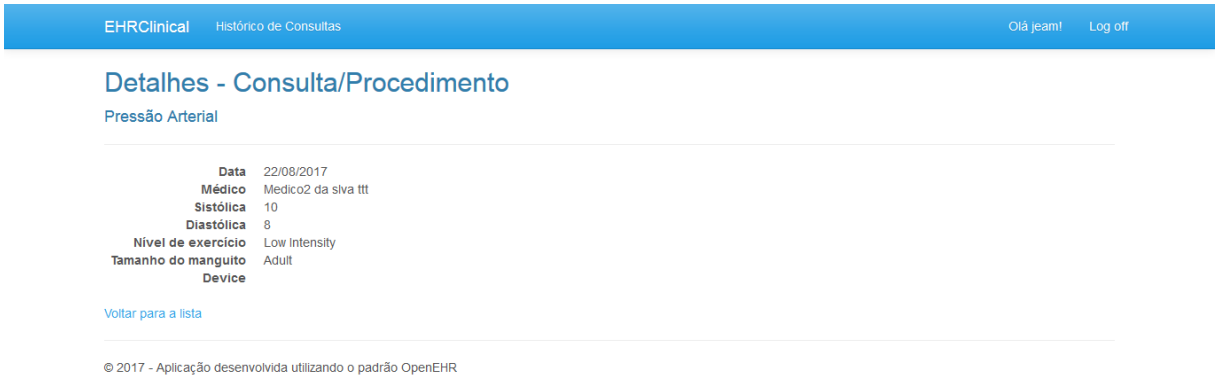


APÊNDICE J – FORMULÁRIO DE CADASTRO DO PACIENTE

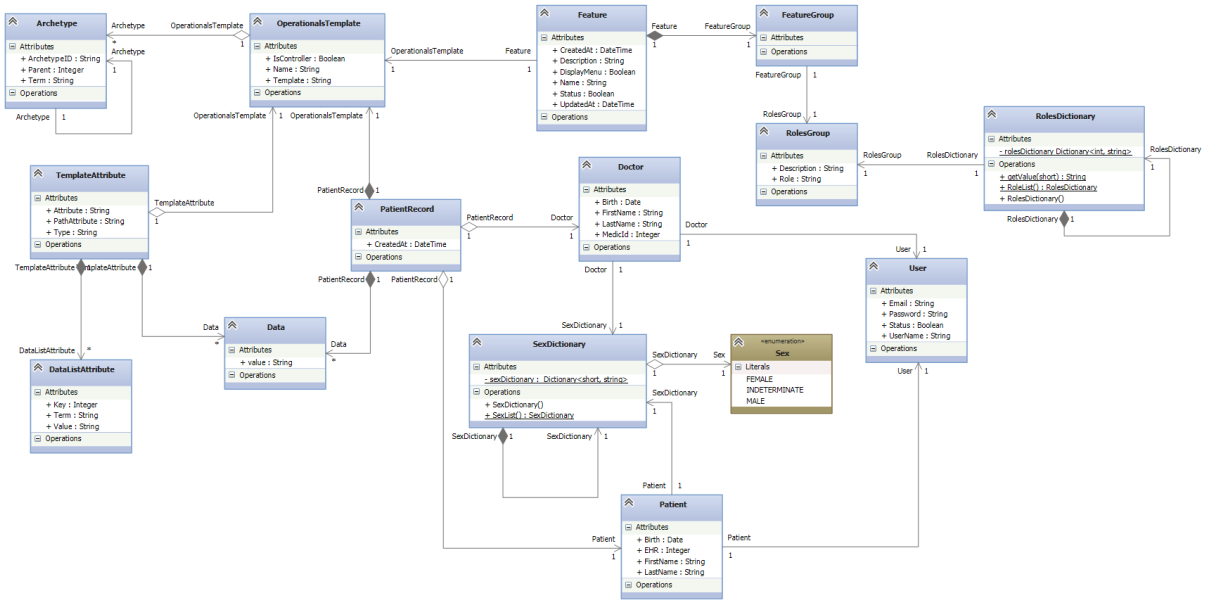


APÊNDICE K – HISTÓRICO DE CONSULTAS DO PACIENTE

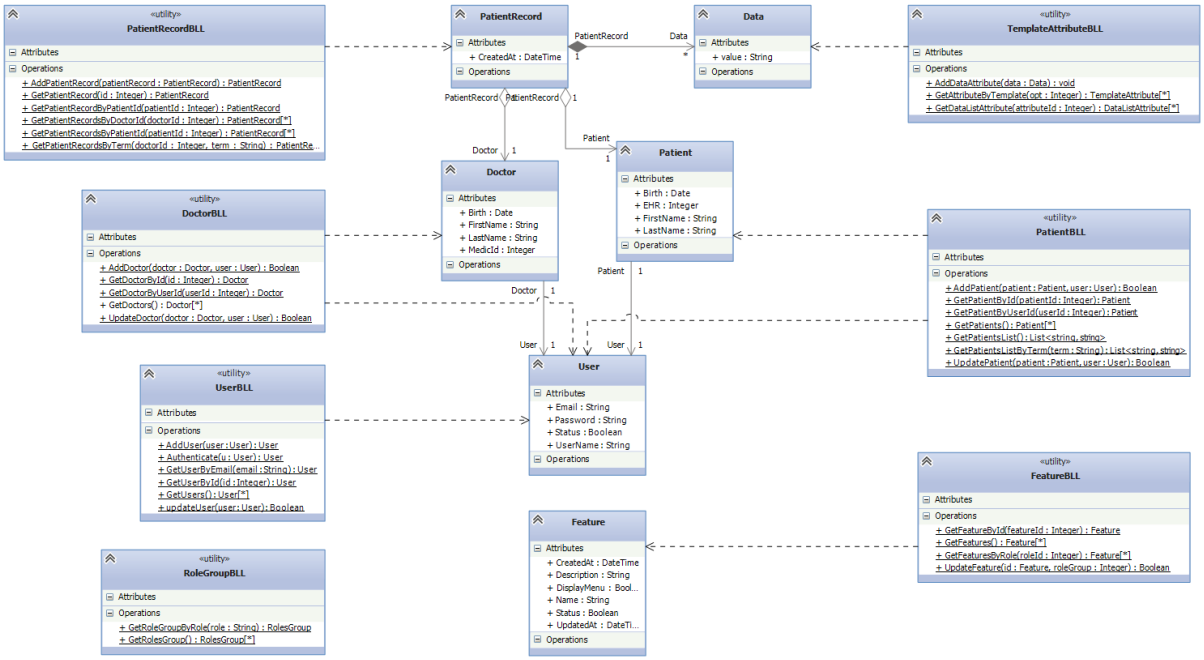
APÊNDICE L – VISUALIZAÇÃO DOS DADOS DE UMA CONSULTA DO PACIENTE



APÊNDICE M – DIAGRAMA DE CLASSES DA CAMADA DE MODELO DAO (*DATA ACCESS OBJECT*)



APÊNDICE N – DIAGRAMA DE CLASSES DA CAMADA DE MODELO *DOMAIN MODEL*



APÊNDICE O – DIAGRAMA DE MODELO ENTIDADE RELACIONAMENTO