PEPVAULT: UM SISTEMA DE PRONTUÁRIO ELETRÔNICO BASEADO NA PLATAFORMA *HEALTHVAULT*

Trabalho de Conclusão de Curso

Engenharia da Computação

Afif A. A. Fikani
Orientador: Prof. Joabe Bezerra de Jesus Júnior

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

AFIF AQUINO ALMEIDA FIKANI

PEPVAULT: UM SISTEMA DE PRONTUÁRIO ELETRÔNICO BASEADO NA PLATAFORMA *HEALTHVAULT*

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, novembro de 2014.

De acordo

Recife

____/_____/_____

_____
Joabe Bezerra de Jesus Júnior

Dedico estre trabalho à minha família, em especial ao meu avô, Almeida.

Agradecimentos

Agradeço à minha família pelo apoio incondicional em todos os momentos, por ter me proporcionado chegar à universidade, por ter acreditado no meu futuro, mesmo nos momentos em que fui ausente. Aos meus familiares Irineide, Irene, Almeida, Irinete e Philipe.

Agradeço ao meu orientador Joabe Bezerra de Jesus Júnior, pela sabedoria, paciência e disponibilidade com a qual conduziu este trabalho e por ter acreditado em mim. Também agradeço pelo apoio do Professor Raul da faculdade de medicina da UPE, sem ele não haveria este trabalho.

Agradeço aos meu colegas de graduação Dennis Cunha, Rafael Ferreira e Diego Martins. Também aos colegas de trabalho que me proporcionaram tempo para me dedicar a este trabalho.

Resumo

Este trabalho apresenta o PEP *Vault*, uma proposta de evolução do sistema de prontuário eletrônico do Núcleo de Telessaúde (NUTES) da Universidade de Pernambuco (UPE). O PEP *Vault* utiliza a plataforma Microsoft *HealthVault* como base para o desenvolvimento. O *Rational Unified* Process foi utilizado no desenvolvimento do software deste trabalho, são exaltados partes das etapas de análise de requisitos e projeto, demonstrando os pontos mais importantes e diagramas mais significantes. Para o desenvolvimento deste projeto, foram utilizadas as tecnologias *WCF* e o *HealthVault*. Além disso, o projeto do sistema levou em consideração o uso da arquitetura *Model-View-Controller (MVC)* e tanto as etapas de projeto quanto construção foram focadas no *backend* do sistema, implementado como um *web service.* Deste modo, este trabalho resulta numa nova plataforma para registro eletrônico em saúde apoiados pelo NUTES da UPE.

Abstract

This paper presents the PEP Vault, a proposal for upgrade of electronic medical records system of Núcleo de Telessaúde from University of Pernambuco. The PEP Vault uses the Microsoft HealthVault platform as a basis for the development of the new system. For this project development, the Rational Unified Process was used during the software development of this paper, the analysis steps of requirements and design are exalted, showing the most important and most significant diagrams. The technologies such as WCF and the HealthVault, were used also the Model-View-Controller architecture were used to generate this new system through web services. Thus, we obtained a new system to be used with the purpose of developing new healthcare software.

Sumário

Índice de Figuras

Tabela de Símbolos e Siglas

(Dispostos em ordem alfabética)

FCM – Faculdade de Ciências Médicas

MVC – Model View Controller

NUTES – Núcleo de Telessaúde

PEP – Prontuário Eletrônico do Paciente

POMR - Problem-Oriented Medical Record

RUP - Rational Unified Process

SGBD - Sistema de Gerenciamento de Banco de Dados

UML - Unified Modeling Language

UPE – Universidade de Pernambuco

web – World Wide Web

Capítulo 1

Introdução

O prontuário do paciente é a principal ferramenta utilizada por profissionais e instituições de saúde com o objetivo de acompanhar a evolução de seus pacientes [1]. É no prontuário onde todos os procedimentos e cuidados relacionados à saúde do paciente são registrados. Ele é organizado em forma de documentos padronizados e ordenados, abstraindo e resumindo dados de saúde complexos [2].

Nos prontuários em papel, a caligrafia pode ser um grande problema: há casos em que a mesma se torna incompreensível e faz com que a proposta do prontuário de documentar e acompanhar a evolução de um paciente se torne difícil [3]. Além disso, podemos citar a dificuldade de transportar grandes volumes de papel e a necessidade de um espaço físico grande para arquivar esses prontuários. Isso ocasiona outro problema, o de catalogar e organizar o arquivo de prontuários. Essa tarefa é bem complicada e é difícil alcançar o ideal de organização, principalmente devido à quantidade de movimentações de prontuários [4].

Em instituições como o Hospital Universitário Oswaldo Cruz, parte integrante da Universidade de Pernambuco, muitos profissionais, professores e estudantes enfrentam a dificuldade de acesso aos prontuários necessários, como por exemplo, durante projetos de pesquisa. Outro problema está relacionado à deterioração do papel. Muitos prontuários antigos estão em estado de decomposição e podem possuir informações valiosas que acabam se perdendo. Ademais, essas informações perdidas poderiam ajudar a solucionar novos casos clínicos.

Como forma de diminuir essas dificuldades a tecnologia da informação fez surgir o prontuário eletrônico, uma forma digital de substituir o prontuário em papel.

A utilização de um prontuário eletrônico pode resultar em uma grande aumento na qualidade e organização dos registros dos pacientes. Ele consegue armazenar grandes volumes de informação e facilita muito o acesso à informação.

Um sistema de prontuário eletrônico pode facilmente cruzar e interligar dados entre registros e assim gerar novas informações [5] [6] [1].

Com a vantagem de um sistema informatizado, a tomada de decisão de um profissional de saúde pode se tornar mais eficiente e eficaz. Além disso, podem ser adicionadas ao sistema funções de teleassistência e segunda opinião formativa tendo em vista a capacidade de compartilhar informações com outros profissionais mesmo à distância [5] [6].

Ao vencer as dificuldades inicias de um profissional de saúde se familiarizar e se habituar a utilizar prontuários por meios eletrônicos, um prontuário eletrônico só tem a ajudar as instituições, os profissionais e a sociedade, agilizando e facilitando todo o processo de manipulação de dados de saúde.

1.1 Objetivo

Este trabalho tem como objetivo utilizar a tecnologia *HealthVault* [11], uma plataforma da área de saúde desenvolvida pela *Microsoft*, para atualizar o software de prontuário eletrônico do Núcleo de Telessaúde (NUTES) da Universidade de Pernambuco (UPE). Aplicando os conceitos da metodologia *RUP* a finalidade é trazer os principais requisitos no sistema legado e fazer o novo sistema funcionar como uma plataforma *back end* contendo as regras de negócios. Este novo sistema deverá ser disponibilizado na forma de serviços *web* para que possa ser utilizado por outras plataformas com o foco em sistemas *front end*[1].

1.2 Metodologia

Este trabalho aplicou conceitos do *Rational Unified Process (RUP)* [7] durante o processo de produção de *software*. O *RUP* é uma metodologia de desenvolvimento de *software* formal e bem definida que propõe tornar mais eficiente a elaboração um *software*. Este processo aborda o desenvolvimento interativo e

[1] http://en.wikipedia.org/wiki/Front_and_back_ends

incremental, utiliza a *Unified Modeling Language (UML)* para especificar, modelar e documentar artefatos.

A linguagem *UML* é utilizada amplamente para representar modelos de arquitetura de software e provê uma grande quantidade de artefatos visuais para modelar cada aspecto de um sistema [8] [9] [10]. O diagrama de classes, diagrama de caso de uso, diagrama de colaboração e diagrama de sequência, são exemplos de artefatos visuais pertencentes à linguagem.

1.2.1 Escopo

O *RUP* define fases para o desenvolvimento de *software*, neste trabalho serão abordadas as seguintes fases:

- Concepção: Nesta fase não será abordada a modelagem de negócios, pois já existe um sistema legado com o qual será trabalhado e por este motivo considera-se esta etapa já realizada. Serão listados os principais requisitos do sistema legado e gerados os seus respectivos casos de uso.

- Elaboração: será desenvolvido um modelo arquitetural com detalhes que servirão de base para o desenvolvimento do sistema, principalmente diagramas de classes utilizando a arquitetura definida.

- Construção: o desenvolvimento em sí do projeto, com base na fase de elaboração do projeto e aplicando as tecnologias necessárias para que o projeto seja realizado.

- Transição: será realizada uma validação do sistema gerado.

1.3 Estrutura do Trabalho

A estrutura deste trabalho é organizada em quatro capítulos. O primeiro capítulo trata de introduzir, contextualizar o problema e apresentar a metodologia utilizada. O capítulo dois trata sobre o estado da arte e explicita a base teórica que

foi utilizada no desenvolvimento deste projeto. O capítulo três demonstra a elaboração e construção do sistema utilizando as tecnologias propostas. Por último, o capítulo quatro conclui o trabalho e destaca os trabalhos futuros deste projeto. O apêndice A possui toda a documentação gerada utilizando o processo *RUP*, requisitos funcionais, diagramas de casos de uso, diagramas de sequência e diagramas de classe.

Capítulo 2

Estado da Arte

Este capítulo apresenta a fundamentação teórica necessária para o entendimento deste trabalho. Na seção 2.1 é abordado o processo *RUP* e *UML* enfatizando suas etapa das mais relevantes. A seção 2.2 discorre sobre prontuários eletrônicos, prontuários problema orientados e o sistema legado no qual este trabalho se baseia. A seção 2.3 explica a plataforma *HealthVault*. A seção 2.4 fala sobre a Computação nas Nuvens. A seção 2.5 comenta sobre o *Windows Communication Fundation*, tecnologia utilizada no desenvolvimento deste trabalho. Por fim, a seção 2.6 explica o *framework PURE MVC* e sua arquitetura de funcionamento.

2.1 *RUP* e *UML*

O *RUP* é um *framework*[2] de processo de desenvolvimento de *software* desenvolvido para ser utilizado em diversas variações de tamanhos e tipos de projeto, adequando-se ao domínio, tecnologia e contexto no qual um projeto está presente [13].

As práticas recomendadas pelo *RUP* foram definidas de forma a aumentar as chances de sucesso de um projeto. No processo de desenvolvimento, o *RUP* é responsável pela definição de perfis de trabalho que definem a responsabilidade por cada tarefa, artefatos que representam os produtos que serão gerados durante o processo de desenvolvimento, as atividades que descrevem como os representantes de cada perfil trabalham para construir o sistema e os fluxos de atividades que servem de guia na execução das atividades [14]. A escolha dos elementos que são utilizados para compor o processo de desenvolvimento varia de acordo com o

[2] http://pt.wikipedia.org/wiki/Framework

projeto com o qual se está trabalhando, ou seja, não necessariamente se utiliza todos os processos definidos pelo *framework* em questão em um projeto.

Este *framework* propõe um clico de vida de projeto baseado no modelo em espiral [15] e faz com que o desenvolvimento de um software seja dividido em iterações, cada uma resultando em uma liberação de versão executável do sistema. Consequentemente esta metodologia cria um ciclo de descoberta, criação e implementação contínuas de requisitos a cada iteração do projeto, e esta é uma das grades vantagens de utilização deste processo de desenvolvimento.

Figura 1. Ciclo de vida *RUP*

Durante a modelagem de negócio, requisitos, análise e projeto de software, vários componentes gráficos podem ser gerados para deixar esta etapa mais completa e para isso a linguagem *UML* é utilizada. Em *UML* um sistema é descrito utilizando diferentes perspectivas, de negócio, caso de uso, de processos e implementação. Cada perspectiva é contemplada por diferentes ferramentas de modelagem, como diagramas de caso de uso, diagramas de atividade, diagramas de sequência, diagramas de colaboração e diagramas de classe [16]. Este trabalho aborda as ferramentas de modelagem mais relevantes ao projeto.

2.1.1 Casos de Uso

Os casos de uso de são responsáveis por moldarem o requisitos funcionais de um sistema. O diagrama responsável por representar graficamente os casos de uso é o diagrama de casos de uso. Este modelo possui uma notação gráfica simples com objetivo de facilitar a comunicação entre desenvolvedores e usuários, além disso é um modelo muito importante pois ele direciona as tarefas posteriores no clico de vida de desenvolvimento de um software. O diagrama de casos de uso de um sistema é composto por atores, casos de uso e o relacionamento entre eles [17].

Os casos de uso são responsáveis por especificar interações entre o sistema e os agentes externos, porém não descrevem comportamentos internos de um sistema. Os agentes externos que vão interagir com o sistema são denominados atores, estes não fazem parte do sistema e podem ser pessoas, organizações e até outros sistemas. Os atores interagem com os casos de uso utilizando relacionamentos como: inclusão, extensão e generalização.

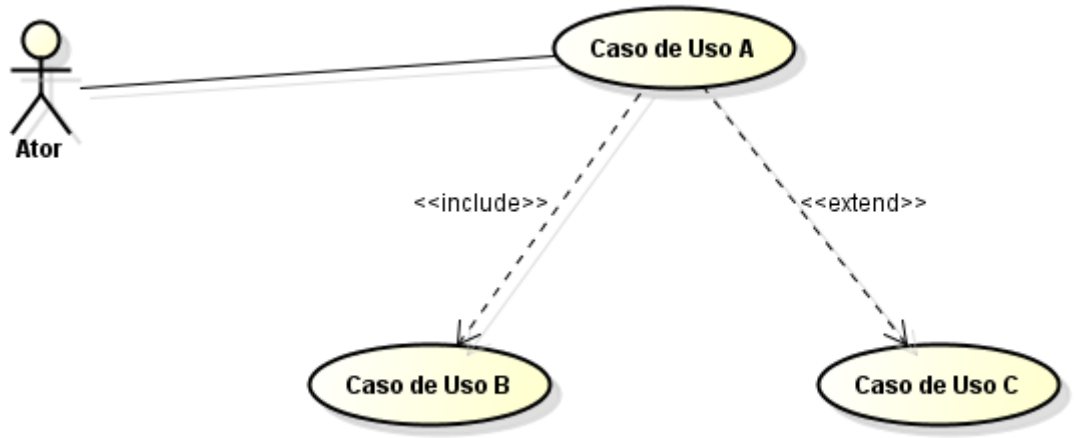


Figura 2. Exemplo de caso de uso

Além de um modelo gráfico de casos de uso, cada caso de uso possui uma descrição da sua funcionalidade, abrangendo seus fluxos principais, alternativos e de exceção. Esta descrição é baseada nas definições das regras de negócio do software. A Figura 3 mostra um exemplo de descrição de caso de uso.

Nome do Caso de Uso	Manutenção de Cliente
Atores	Usuários, Administrador
Dados Consumidos	Nome, Endereço, Telefone, Email, CPF
Dados Produzidos	Código do cliente
Prioridade	Alta
Fluxo Principal	
Ação do Ator	**Resposta do Sistema**
O usuário após autenticar-se, poderá dar manutenção(incluir, alterar, Excluir e Consultar) nos clientes	
Incluir, Alterar, Excluir ou Consultar	Apresentar formulário padrão
Fluxo alternativo	
Ação do Ator	**Resposta do Sistema**

Figura 3. Exemplo de descrição de caso de uso

2.1.2 Diagrama de Classe

O diagrama de classes é utilizado na construção de modelos de classes desde o nível de análise até o de especificação. O diagrama de classes é um modelo fundamental de uma especificação orientada a objetos. É um diagrama mais próximo da implementação de um software, descrevendo detalhes de seus componentes, como atributos e métodos de classes [17]. Ele representa as classes, interfaces e relacionamentos entre elas, além de servir como base para construção de outros diagramas. Exemplos de relacionamentos podem ser observados na figura a seguir.

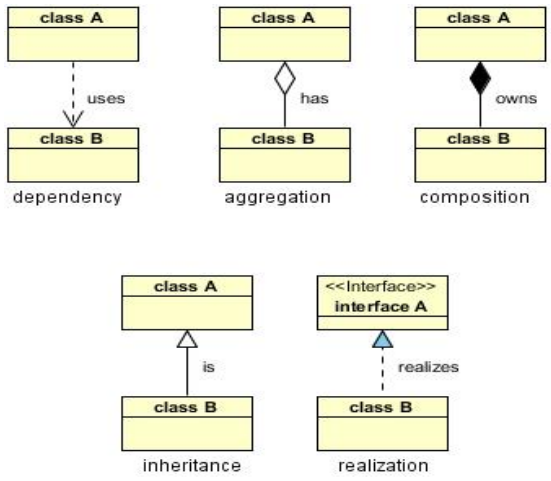


Figura 4. Exemplos de relacionamentos no diagrama de classes

Como o processo de desenvolvimento de um software com *RUP* é interativo, incremental e cíclico, é comum acontecer de uma etapa posterior observar um erro ou melhoria que pode ser realizada em uma etapa anterior. Durante o desenvolvimento do diagrama de classes é possível observar mudanças na etapa de requisitos, nos casos de uso, desta forma há um ciclo de colaboração entre as fases de desenvolvimento.

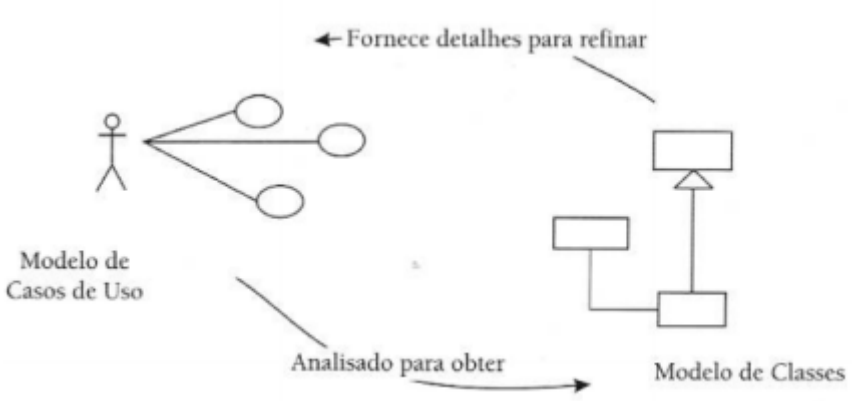


Figura 5. Exemplo de ciclo interativo no desenvolvimento de software

2.1.3 Diagrama de Sequência

O diagrama de sequência é classificado com um diagrama de interação, pois seu foco é mostrar a forma que grupos de objetos se relacionam. Este diagrama demonstra a sequência temporal de eventos que ocorrem em um determinado caso de uso. De forma clara, é possível observar a ordem que métodos de objetos são chamados para realizar as tarefas de um caso de uso [17].

Este diagrama utiliza objetos definidos nos diagramas de classe, é capaz de demonstrar graficamente os atores interagindo com as instâncias de objetos. O diagrama de sequência possui duas dimensões: vertical, que representa o tempo; e horizontal que representa atores, objetos e a troca de mensagens. A passagem temporal é realizada no sentido vertical de cima para baixo e o sentido é demonstrado pelas setas. A imagem abaixo demonstra uma exemplo de diagrama de sequência.

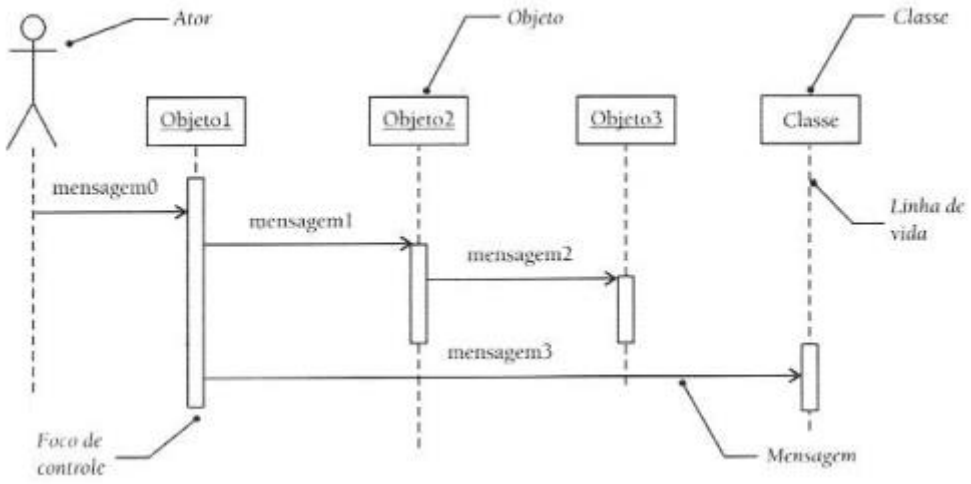


Figura 6. Exemplo de diagrama de sequência

Este diagrama possui grande valor quando utilizado por um desenvolvedor, sua sequência de ações descritas facilitam a etapa de programação do software, diminuindo as chances de erros e padronizando o sistema de forma a deixar clara à equipe o funcionamento a nível de código de um requisito.

2.2 Prontuário do Paciente

O prontuário do paciente é o principal ferramenta utilizada por profissionais e instituições de saúde com o objetivo de acompanhar a evolução de seus pacientes. A necessidade do uso de prontuários decorreu de uma mudança na compreensão da natureza da doença pelos profissionais de saúde, “pensada agora como um processo, e só passível de ser diagnosticada através da observação sistemática” [18]. Na atualidade, os prontuários são indispensáveis para profissionais de saúde. Esta ferramenta colabora consideravelmente para uma melhor gestão de informação hospitalar e do paciente, pois ela dá suporte ao processo de atenção à saúde, registro de ações médicas, às pesquisas e promoção do ensino e gerenciamento de serviços [18]. Além disso, sua organização tende a ser padronizada e ordenada de forma a simplificar e resumir informações de saúde consideradas complexas [2].

De acordo com [19], “Reforçar a importância do prontuário médico é procedimento óbvio, mas indispensável. A própria experiência dos Conselhos de

Medicina comprova que o prontuário é instrumento fundamental não só para contribuir com a qualidade de atendimento ao paciente como também – quando isto se faz necessário – para a defesa do médico em eventuais demandas judiciais e nos Conselhos de Medicina.”

2.2.1 Prontuário Problema Orientado

Com a utilização de prontuários eletrônicos, é natural que surjam padrões de utilização desta ferramenta. Na década de 1960 *Lawrence L. Weed* introduziu um novo conceito de prontuário, o *Problem-Oriented Medical Record (POMR)* [2].

Até o dado momento, registros de saúde eram organizados cronologicamente e, de acordo com a pessoa que manuseava o registro, eram organizados em seções e nomeadas como, por exemplo, “notas médicas”, “notas da enfermaria”, “relatórios laboratoriais”, entre outros. Eram informações de manutenção exaustivas e seguindo poucos ou nenhum padrão de preenchimento [2].

A variedade de problemas que os médicos lidavam no dia-a-dia levou a *Lawrence L. Weed* a desenvolver o *POMR*. A partir de então, estudantes e médicos puderam trabalhar de uma forma estruturada e rigorosa. Logo em seguida foi introduzido um formato de dados ao *POMR* chamado *SOAP*, um acrônimo para *Subjective Objective Assessment Plan* [25]. Nesta metodologia de prontuários, problemas identificados em um contato inicial são listado e enumerados, visitas subsequentes são documentadas, e notas no formato *SOAP* são geradas referenciando os números dos problemas encontrados. Este formato de dados já é utilizado globalmente por diversos tipo de profissionais de saúde e é bem reconhecido por mudar o foco do prontuário para o paciente [25].

Apesar da utilização do *POMR* não mudar a variedade de problemas enfrentados pelos médicos, ele permitiu uma abordagem mais organizada para o grau de complexidade enfrentado [2].

2.2.2 Prontuário Eletrônico

A utilização de prontuários trouxe diversas vantagens, mas não corrigem todos os problemas encontrados em instituições de saúde. Uma grande deficiência que ainda existe é a gerência de informações. Isso se observa em um atendimento ao paciente que em geral é multidisciplinar, onde muitos profissionais estão envolvidos na geração e manipulação de informações de saúde [18]. Uma solução para tornar este processo mais eficiente é a adoção de tecnologias da informação, mais precisamente um Prontuário Eletrônico do Paciente (PEP).

Este tipo de tecnologia tem sido disseminado internacionalmente em hospitais, como é possível notar no experimento realizado em três hospitais estudados em [20]. A utilização do PEP traz inovação administrativa baseada na introdução de novo processo de prestação de serviço. A qualidade e organização dos registros dos pacientes que o uso do PEP proporciona é notável, utilizando-o é possível armazenar grandes volumes de informação e, comparando à manipulação de informações de forma manual, facilita o acesso à informação, pois computadores podem cruzar, interligar e gerar dados utilizando registros dos pacientes com esforço menor [5] [6] [1].

2.2.3 PEP NUTES UPE

O Núcleo de Telessaúde da Universidade de Pernambuco foi responsável por desenvolver um projeto envolvendo prontuários eletrônicos utilizando o *POMR*. Durante este projeto foi desenvolvido o PEP no NUTES da UPE, um sistema *web* seguindo as diretrizes de um prontuário problema orientado e eletrônico. Este projeto foi iniciado como um projeto de extensão com a união da Faculdade de Ciências Médicas da UPE (FCM) e a Escola Politécnica de Pernambuco representada pelo curso de Engenharia da Computação. Posteriormente foi ligado ao Núcleo de Telessaúde da UPE onde o seu desenvolvimento foi finalizado. Atualmente, este sistema é utilizado no pelo NUTES de forma didática em disciplinas da Faculdade de Ciências Médicas (FCM) da UPE.

O PEP do NUTES é dividido em fichas médicas que representam os dados dos paciente que são descritos pelos profissionais de saúde, é um conjunto

organizado e qualificado de dados seguindo os moldes de um prontuário problema orientado. Cada ficha do PEP do NUTES possui uma página no sistema representada por uma aba, a ordem das abas foi definida para que fizesse mais sentido ao profissional de saúde durante a anamnese. Existem dois tipos básico de usuário no sistema, o usuário assistente administrativo que possui acesso ao cadastro de pacientes e é responsável por mantê-los, enquanto o outro usuário é o profissional de saúde, que tem acesso ao sistema completo e é responsável por manter os dados de saúde dos pacientes. Abaixo segue uma imagem da página de uma ficha denominada Ficha Médica.

Figura 7. Ficha Ficha Médica do PEP do NUTES da UPE

2.3 *Microsoft® HealthVault©*

HealthVault [11] é uma plataforma atuante na área de saúde que foi desenvolvida pela *Microsoft*. Esta plataforma foi lançada em 2007 e atua como produto final ao usuário e também como recurso para desenvolvedores.

O usuário final do *HealthVault* pode armazenar os dados de saúde na plataforma e também pode criar e manter perfis de saúde de familiares. Esta plataforma possui uma loja de aplicativos que os usuários podem aderir, são aplicações que se utilizam dos dados do sistema para tratar, geralmente, assutos específicos ao usuário. Além disso a plataforma possui diversos dispositivos compatíveis que podem ser integrados, por exemplo: monitores de fequência cardíaca, monitores de pressão sanguínea, entre outros. Os dados destes dispositivos podem ser sincronizados com a plataforma e serem compartilhados com as aplicações ou outros participantes da plataforma.

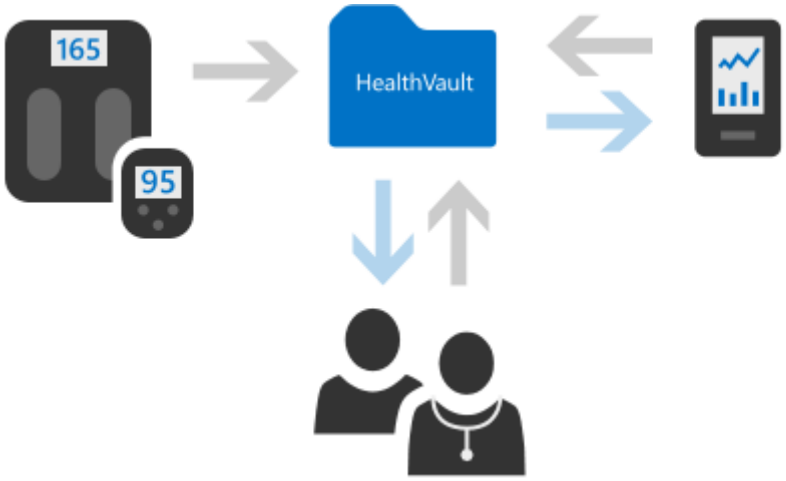


Figura 8. Arquitetura de comunicação do *HealthVault*

A plataforma disponibiliza meios de desenvolver aplicações que funcionam em conjunto com o *HealthVault*, dando a oportunidade de desenvolvedores criarem sua aplicação e terem acesso aos usuários e dados de saúde armazenados no sistema da *Microsoft.*

Esta plataforma aos poucos está se popularizando e assim surgem exemplos de sistemas que utilizam o *HealthVault* e atuam na área de saúde, como o citado em [21].

2.3.1 Conceitos

Como forma de representar as informações de saúde dos pacientes, a plataforma introduz termos específicos para os tipos de dados comportados. Existem quatro principais tipos de dados definidos: *Record*, *Person*, *Thing*, *Thing Type*[3].

O *Record* representa um conjunto de informações de saúde, ele é responsável por agrupar e organizar outros tipos de dados.

O *Person* representa um indivíduo, no caso um usuário, com registros de saúde. Ele possui controle total dos seus registros e pode compartilhá-los com outros indivíduos.

O *Thing* são dados que compõem um *Record*, podem ser medições de pesos, aferições de pressão arterial, documentos, entre outros dados suportados no *HealthVault*.

O *Thing Type* é a definição de cada *Thing*, por exemplo em uma pressão arterial tem-se os valores sistólicos e diastólicos.

Todos estes dados estão definidos pela plataforma e representados na forma de *Extensible Markup Language (XML)* [12]. A Figura abaixo mostra um exemplo de *XML* extraído do *HealthVault*.

[3] http://msdn.microsoft.com/en-US/healthvault/dn783353

```
<?xml version="1.0" encoding="utf-16"?>
<blood-pressure>
  <when>
    <date>
      <y>2006</y>
      <m>1</m>
      <d>1</d>
    </date>
    <time>
      <h>9</h>
      <m>30</m>
      <s>0</s>
      <f>0</f>
    </time>
  </when>
  <systolic>115</systolic>
  <diastolic>76</diastolic>
  <pulse>80</pulse>
  <irregular-heartbeat>false</irregular-heartbeat>
</blood-pressure>
```

Figura 9. Exemplo de *XML* do *Thing de* pressão sanguínea

2.3.2 Arquitetura

O *HealthVault* provê a possibilidade de toda informação de um aplicativo desenvolvido para a plataforma ser armazenada diretamente nos servidores da *Microsoft*, sem haver necessidade armazenamento local. Quando há a necessidade de dados específicos para uma aplicação, e que não estejam ainda definidos pelo sistema, existe a possibilidade de uma aplicação criar um tipo próprio e ainda assim serem armazenados nos servidores da *Microsoft*. O novo tipo pode ser permanentemente adicionado ao *HealthVault*, caso seja de uso geral e realmente útil à maioria das aplicações (como os tipos de dados já existentes são considerados). A adição permanente de um novo tipo ao sistema deve ser feita por um contato direto com a empresa e é sujeito à avaliação, geralmente é um processo bem demorado.

Apenas um novo tipo de dado é permitido por aplicação, é um limitação explicitada da *Microsoft*. Contudo, a própria empresa, na documentação do *HealthVault*, explica formas de trabalhar com mais de um tipo de dados, mesmo a plataforma identificando todos os tipos como um só.

O *HealthVault* possui duas formas de acesso aos dados da plataforma, o acesso *online* e *offline*. Na utilização *online* o usuário acessa diretamente o aplicativo e toda interação de dados depende do usuário e a utilização ativa da aplicação. No acesso *offline*, após a devida permissão dada pelo usuário ao aplicativo para acesso *offline*, é possível acessar a qualquer momento as informações e realizar leitura e escrita de acordo com as permissões concedidas,

independentemente se o usuário está ou não utilizando a aplicação. A forma na qual o paciente permite acesso *offline* ao seus dados às aplicações é denominada *Patient Connect*.

A *Microsoft* disponibiliza o *HealthVault.NET Software Development Kit (HealthVault SDK)* para ser utilizado juntamente com o *.NET Framework* como forma de facilitar o acesso à plataforma. Desta forma, grande parte das configurações e chamadas dos serviços do *HealthVault* são simplificadas na linguagem de programação utilizada, tornando o desenvolvimento transparente para o desenvolvedor.

```
OfflineWebApplicationConnection personConnection = new OfflineWebApplicationConnection(
    applicationId,
    healthServiceUrl,
    patientConnection.PersonId);

HealthRecordAccessor accessor = new HealthRecordAccessor(
    personConnection,
    patientConnection.RecordId);
```

Figura 10. Exemplo de código C# utilizando o *HealthVault SDK*

2.4 Computação nas Nuvens

A computação nas nuvens é um conjunto de recursos computacionais de serviços e armazenamento. É um modelo de computação onde serviços são fornecidos sob medida durante um período de tempo, em outras palavras, são alugados sistemas e equipamentos em servidores remotos responsáveis por realizar as tarefas designadas pelos usuários. Comumente os recursos são particionados e compartilhados entre usuários, isto faz com que o serviço possui um custo menor e conforme a demanda aumenta os recursos aumentam. Ao consumir um serviço de computação nas nuvens o usuário se livra dos trabalhso de configuração e implementação do ambiente, não precisa se preocupar com a manutenção de equipamentos e já possui toda uma segurança da informação definida [29].

A computação nas nuvens é dividida em três grandes conjuntos: infraestrutura, plataforma e aplicação. A infraestrutura, *Infrastrucuture As A Service*,

é onde são implementadas máquinas virtuais e executar múltiplos sistemas operacionais em uma máquina, compartilhando os recursos de hardware. A plataforma, *Platform As A Service*, é responsável por prover plataformas de desenvolvimento já previamente configuradas com objetivo de agilizar parte da implementação para os usuários. Por fim, a aplicação, *Application As A Service*, é responsável por prover a aplicação para o usuário final. Aplicações dependem de plataformas que dependem de infraestrutura, todos estão interligados. A Figura 11 mostra a arquitetura da Computação nas Nuvens.

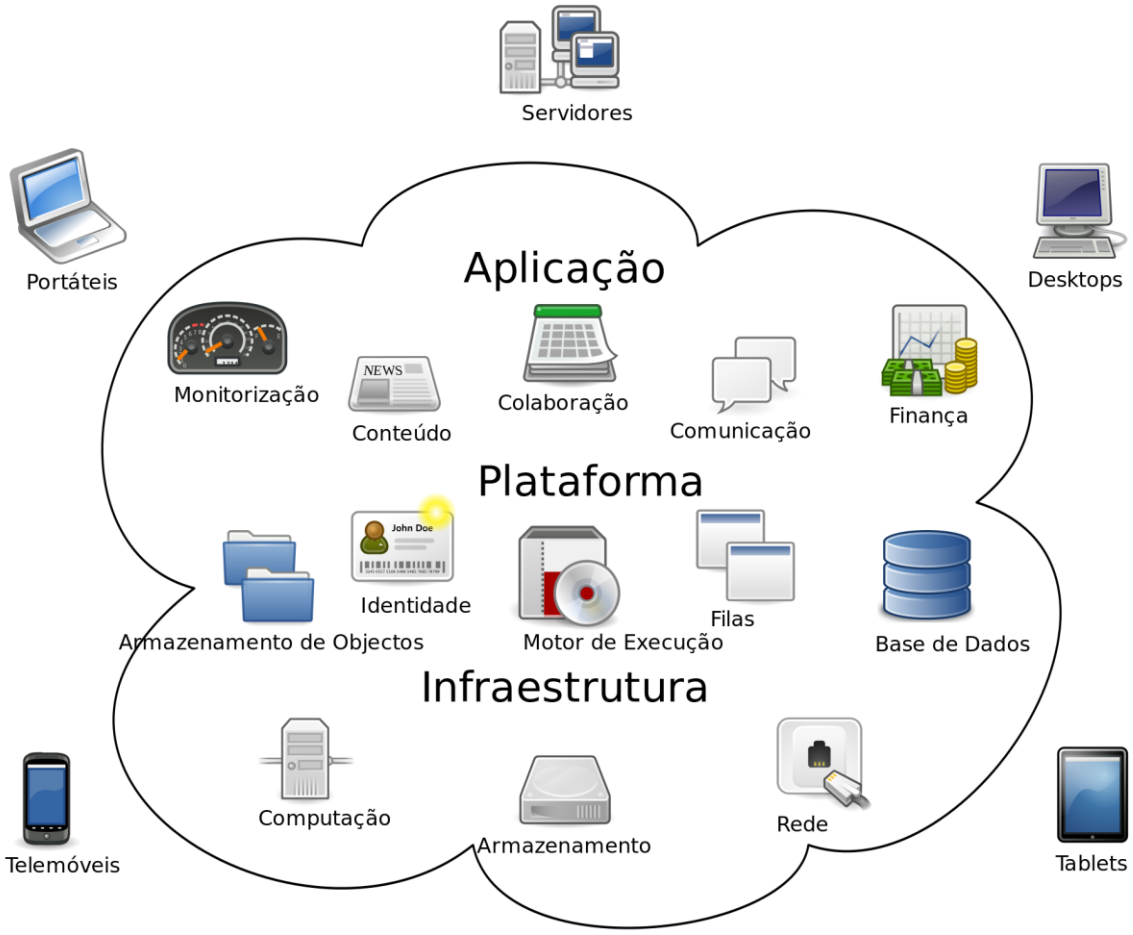


Figura 11. Arquitetura da Computação nas Nuvens

2.5 Windows Communication Foundation

Windows Communication Foundation (WCF) é um *framework* da Microsoft para o desenvolvimento de sistemas orientados a serviço[4]. Este *framework* proporciona facilidades para o desenvolvimento de serviços web abstraindo complexas tarefas como: segurança, serialização, controle de transações, entre outras. O *WCF* foi utilizado no desenvolvimento deste trabalho também por ser uma tecnologia *Microsoft*, assim como o *HealthVault*, e trouxe facilidade de integração entre as plataformas. Este projeto é disponibilizado nas nuvens utilizando este *framework* da *Microsoft*.

Como o foco deste trabalho não é discutir sobre *WCF*, as informações sobre o mesmo podem ser encontradas em [22].

2.6 PURE MVC

O desenvolvimento de *software* nos provê várias formas de desenvolvimento, entre elas existe a oportunidade de lidar com a complexidade de *software*. Esta etapa está ligada diretamente à manutenibilidade de um *software*, a fase de elaboração da solução arquitetônica do *software*. Padrões de projeto foram criados para especificar problemas comuns em aplicações, padronizar e facilitar soluções. Mesmo utilizando padrões, somente isso não garante uma solução bem colocada. Este é um problema que padrões de arquitetura ajudam a resolver [23].

Uma das mais populares arquiteturas utilizadas por desenvolvedores é um padrão chamado *Model-View-Controller (MVC)*. Esta proposta de arquitetura divide um sistema em três grandes camadas como o nome sugere: *Model* que é responsável por manter os dados e informações relevantes do sistema, *View* que é responsável pela interface de utilização e interação do sistema com usuários e agentes externos, e *Controller* que é responsável pelas regras de negócio definidas pelo sistema [23]. A imagem abaixo ilustra um exemplo de arquitetura *MVC*.

[4] http://pt.wikipedia.org/wiki/Service-oriented_architecture

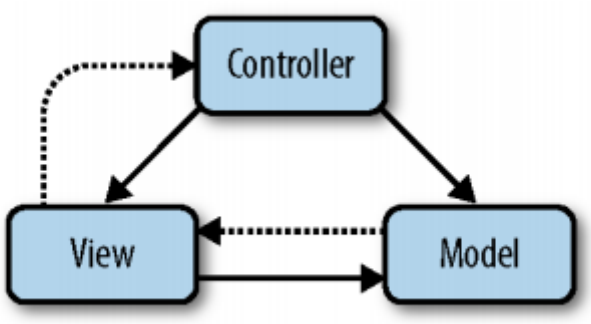


Figura 12. Arquitetura *MVC*.

Com a popularização desta arquitetura, muitos *frameworks* foram desenvolvidos seguindo os conceitos de *MVC*. Um desses *frameworks* criados foi o *PURE MVC*. Inicialmente o *PURE MVC* foi criado para a linguagem de programação *ActionScript 3*[5], mas com sua boa aceitação, logo outras linguagens receberam suas versões [24].

O *PURE MVC* é um *framework* leve para criação de aplicações baseado no *MVC*. Com a junção de alguns padrões de projeto o *PURE MVC* construiu solução para desenvolvedores, alguns dos padrões utilizados foram: *Singleton, Façade, Mediator, Command, Proxy e Observer*.

Todos os componentes do *PURE MVC* são interligados por uma *Façade*, que possui a responsabilidade de registrar, dar acesso e remover os componentes. A *Façade* se comunica com objetos que representam o *Model*, *View* e o *Controller*, delegando a cada um sua responsabilidade. A comunicação da camada *View* com a camada *Controller* é feita através de notificações. Um *Mediator* responsável por um componente da *View* e através da utilização do padrão *Observer* notifica os *Commands* para executarem uma ação. Geralmente os *Commands* são responsáveis por entrar em contato com a camada *Model* através da *Façade* e assim tem acesso aos *Proxies*. Além disso os *Commands* também podem notificar outros *Commands*. Um *Proxy* mantém os dados da aplicação, após fazer as alterações pedidas pela camada *Controller* podem notificar a *View* e assim permitir a

[5] http://www.adobe.com/devnet/actionscript/learning.html

atualização na apresentação dos dados ao usuário final. O diagrama abaixo ilustra esta arquitetura descrita.

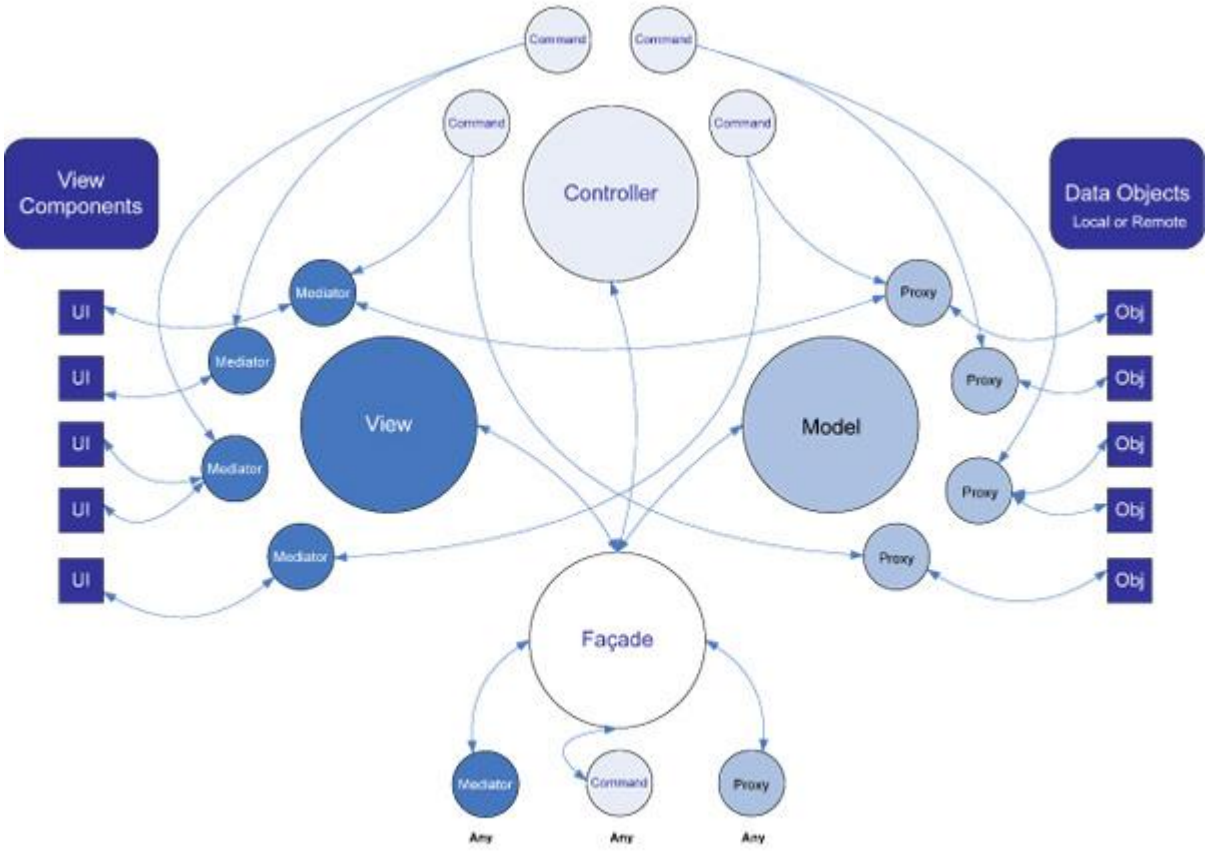


Figura 13. Arquitetura *PURE MVC*.

2.7 Banco de Dados

Um banco de dados é um conjuto de dados persistentes, um local onde os dados necessários ficam armazenados para serem utilizados em determinadas funcionalidades de um sistema [27]. Seu nome originou-se do termo em inglês *Databanks*, que posteriormente se tornou *Database*.

Os dados mantidos em um banco de dados geralmente representam aspectos do mundo real, em outras palavras, um banco de dados é uma coleção lógica e coerente de dados com um significado no contexto utilizado. Um conjunto aleatório não pode ser considerado um banco de dados, pois um banco de dados é

construído e preenchido com finalidade específica e possui um grupo específico de usuários [28].

O gerenciamento de um banco de dados é realizado por um *software* chamado Sistema de Gerenciamento de Banco de dados (SGBD) [27]. O objetivo deste *software* é disponibilizar recursoso para definir, assegurar, criar, manipular, e manter dos dados de uma bando de dados [28]. A Figura 14 mostra uma exemplo de arquitetura de um SGBD.

Figura 14. Exemplo de arquitetura de um SGBD.

A *Structured Query Language (SQL)* foi criada no início da década de 1970, para o protótipo de um banco de dados *System R* [27]. Inicialmente com o nome de *SEQUEL (Structured English QUEry Language)*, *SQL* hoje é a linguagem padrão para Sistemas de Gerenciamento de Bancos de Dados Relacionais (SGBDR) [28]. A *SQL* é composta por subdivisões e dentre as mais importantes estão *Data Definition Language (DDL)* e *Data Manipulation Language (DML)* que são responsáveis por definir e manipular os dados de um banco de dados, respectivamente. Utilizando a *DML* é possível realizar inclusões, consultas, exclusões e alteração de dados e para isso os comandos *INSERT, SELECT, UPDATE* e *DELETE* são usados. A *DDL* define associações e tabelas utilizando os comandos CREATE, DROP e ALTER.

Neste trabalho foi utilizada o banco de dados *SQLite* [26] é um banco de dados que não possui um SGBD e mantém os dados em um arquivo único. É uma ferramenta mais simples que os bancos de dados mais tradicionais e geralmente disponibilizada na forma de uma biblioteca de linguagem de programação. A decisão de utilizá-lo parte de alguns pontos: o sistema possui a maior parte dos dados guardados em uma fonte externa, os dados as serem guardados são de baixa complexidade e em maior parte ligados à ação de Login do sistema, e diminuir a complexidade do sistema evitando um SGBD.

Capítulo 3

PEP *Vault*

Neste capítulo será abordado o processo usado para desenvolver este projeto. Na seção 3.1 o sistema PEP *Vault* é apresentado e a sua arquitetura é explicada. Na seção 3.2 é mostrada a fase de análise de requisitos, mostrando as dificuldades e os diagramas gerados. Por último, na seção 3.3 é mostrada a análise de projeto, exemplificando os principais diagramas e a arquitetura de funcionamento do sistema.

3.1 Visão Geral e Arquitetura

PEP *Vault* é o nome do sistema desenvolvido neste trabalho com base no sistema legado PEP do NUTES UPE. Este sistema possui o foco em disponibilizar as funcionalidades do sistema legado na forma de serviços, possibilitando que outros sistemas possam consumir os serviços do PEP *Vault* e utilizá-lo da forma que melhor for adequada. Estes serviços devem ser disponibilizados na forma de *web services*, para isso foi utilizado o *WCF* como base de desenvolvimento. A Figura 15 mostra a arquitetura do PEP V*ault* juntamente com a utilização do *HealthVault*.

O PEP *Vault* utiliza o *HealthVault* para ter acesso aos dados do paciente. Todo dado de saúde é manuseado pelo PEP *Vault* e armazenado no sistema da *Microsoft*, fazendo um trabalho de integração entre os sistemas. A imagem acima demonstra a utilização do sistema desenvolvido através de aplicações intermediárias e responsáveis por prover a interface com o usuário.

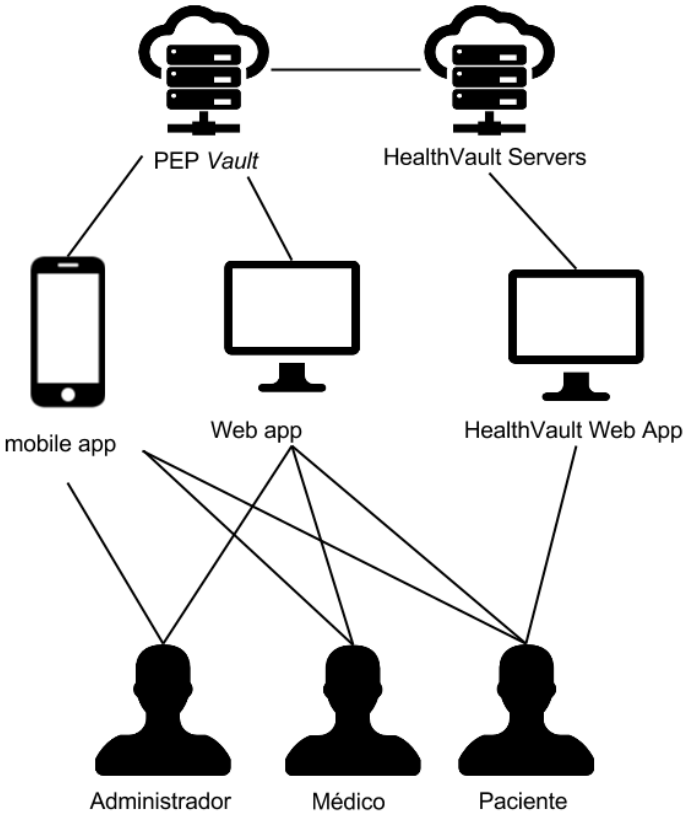


Figura 15. Arquitetura PEP *Vault.*

3.2 Análise de requisitos

Nesta etapa foram criados artefatos de requisitos com base no sistema legado, esta é uma etapa importante e que afeta todas as posteriores. Foram gerados modelos representativos dos requisitos e as descrições com base nos requisitos.

Um dos problemas da migração do PEP do NUTES UPE é o fato de não haver documentação sobre o sistema desenvolvido. Todo o sistema anterior foi concebido sem haver também versões ou interações, diferente do que é pregado pelo *RUP*. Tendo em vista este problema, na primeira etapa do desenvolvimento foi feita uma análise do sistema legado e a geração de requisitos baseados na utilização do sistema. Todas as telas e usuários foram analisados, porém os principais e essenciais requisitos funcionais foram descritos para o desenvolvimento da nova plataforma, o PEP *Vault*.

3.2.1 Requisitos

Ao todo foram gerados oito requisitos funcionais descrevendo as funcionalidades do PEP *Vault*. As principais funcionalidades do antigo sistema foram mantidas, as de menor utilização e relevância foram excluídas. O gerenciamento de usuários foi um ponto mantido e melhorado para o novo contexto. Dentre os requisitos mais importantes, podemos citar o Gerenciar Pacientes que teve que ser adaptado ao contexto do *HealthVault*, pois os pacientes não são mais mantidos na própria plataforma e sim através da integração entre os sistemas. O requisito abaixo descreve o novo funcionamento.

RF02 – Gerenciar Pacientes

Descrição	O cadastro do paciente deve ser realizado integrado com o sistema *HealthVault* da *Microsoft*, é deste sistema que os Pacientes vem ao sistema PEP *Vault*. O cadastro no PEP *Vault* significa o paciente dar acesso de leitura e escrita de suas informações pessoais e de saúde que estão contidas no *HealthVault*. É necessário um cadastro prévio no *HealthVault* e após isso a realização de uma comunicação entre os sistemas. A aquisição de novos Pacientes deve ser realizada diretamente entre o PEP *Vault* e o próprio paciente, de forma automatizada. O processo é iniciado no PEP *Vault* e realizando um encaminhamento para o *HealthVault*. Após a liberação de acesso aos dados do paciente o sistema deve ser capaz de utilizá-lo. A remoção do acesso do PEP aos dados do paciente é realizada pelo próprio, utilizando o *HealthVault*.
Prioridade	Essencial

Figura 16. Requisito Funcional 2, Gerenciar Pacientes.

3.2.2 Casos de uso

A elaboração de casos de uso foi utilizada para definir os requisitos do sistema e servir de base para as etapas futuras. Foram definidos três atores para o sistema, o Paciente, o Administrador e o Médico. A Figura 17 descreve cada uma das funções (chamadas de responsabilidades).

Nome	Descrição	Responsabilidades
Paciente	Paciente	Se cadastrar no sistema
Administrador	Administrador do sistema	Criar contas de usuários
Médico	Profissional de saúde que atenderá o paciente	Gerenciar Pacientes e fichas médicas dos pacientes.

Figura 17. Atores do sistema PEP *Vault*

Pelo fato do PEP *Vault* utilizar a plataforma *HealthVault*, todo usuário Paciente também deve ser capaz de utilizar a plataforma da *Microsoft*. Por isso foram gerados diagramas de casos de uso para ambas as plataformas.

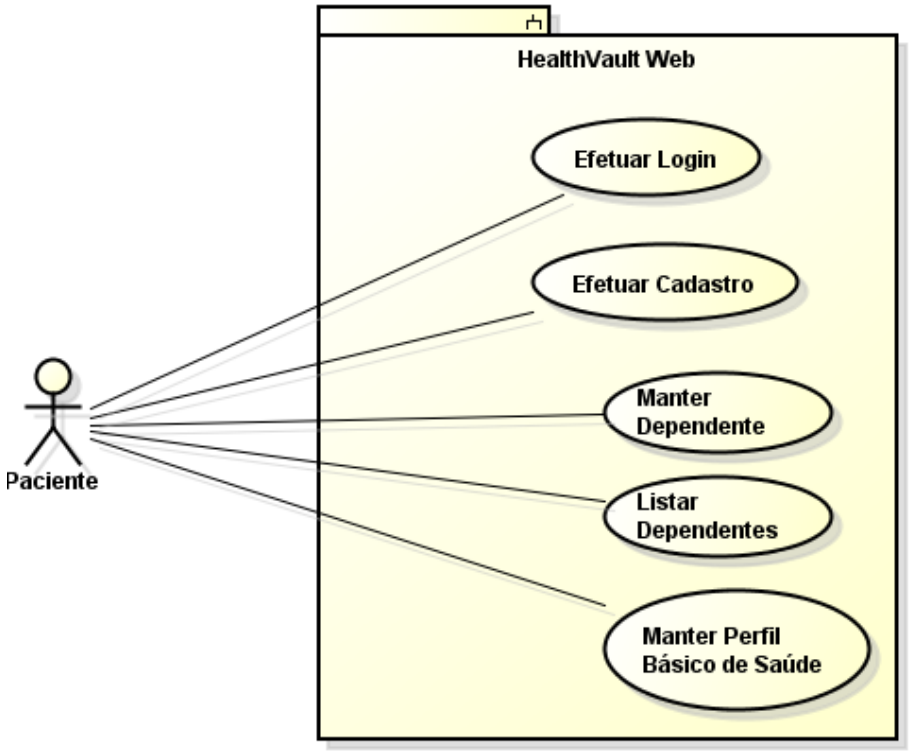


Figura 18. Casos de uso do *HealthVault*.

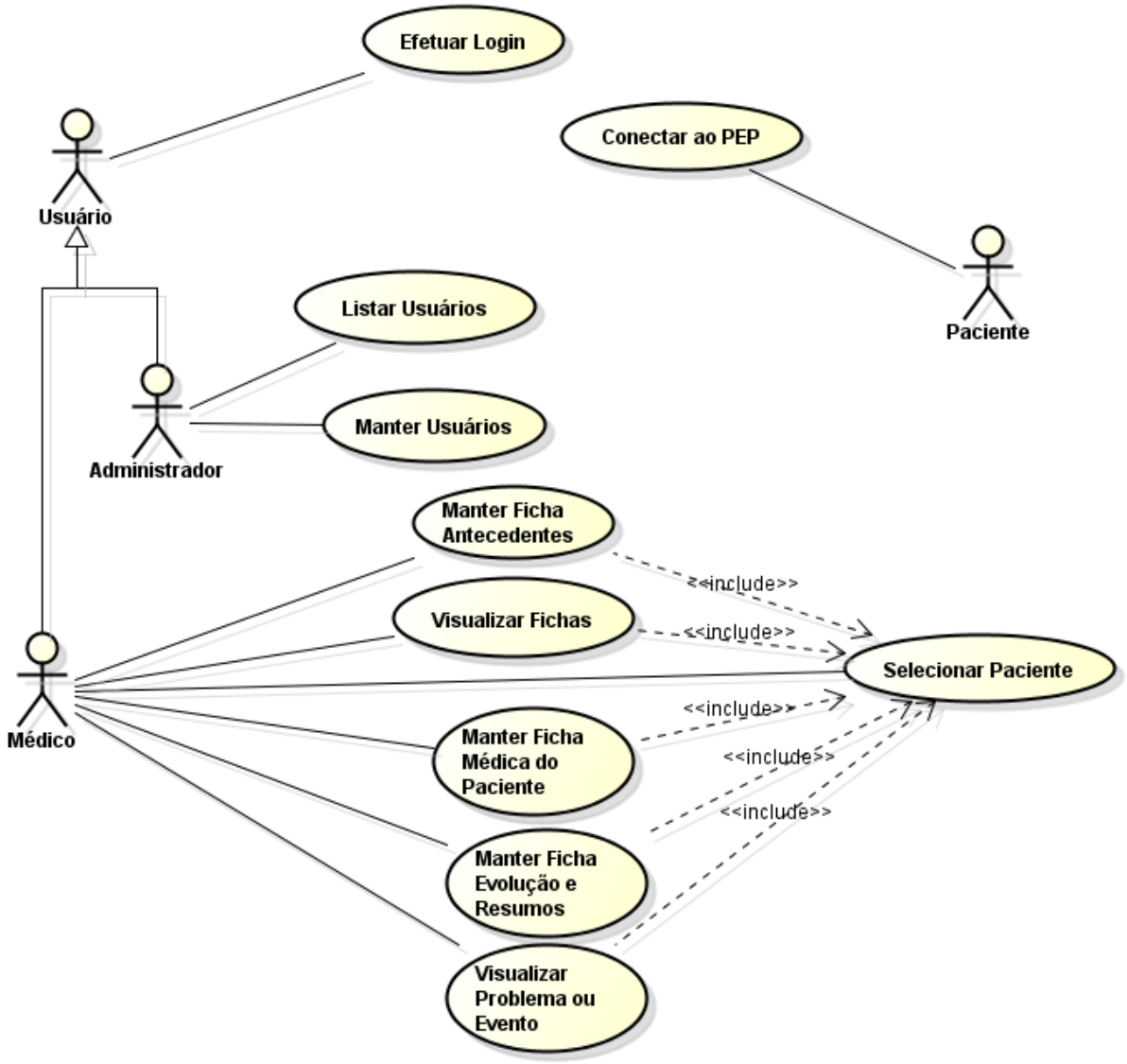


Figura 19. Casos de uso do PEP *Vault.*

Cada caso de uso gerado possui sua descrição de fluxo de execução. São descritos os fluxos: principal, alternativos e exceções. Pode-se observar o exemplo de descrição do caso de uso Conectar ao PEP, na imagem seguinte.

UC004 - Conectar ao PEP
Atores: Paciente
Breve Descrição Realiza a integração de um paciente do *HealthVault* ao PEP *Vault*.
Fluxo de Eventos **Fluxo Básico** 1. O usuário acessa a área de cadastro de pacientes; 2. O usuário preenche os campos; 3. O usuário confirma a ação uma de conectar ao PEP *Vault*; 4. O sistema gera o código de integração com o *HealthVault*; 5. O sistema redireciona o Paciente ao *HealthVault*. 6. O usuário efetua o login no HealthVault e realiza o processo de integração com o PEP; 7. O sistema consegue ter acesso ao Paciente. **Fluxos Alternativos** N/A
Exceções: N/A
Precondições Possuir um cadastro no HealthVault.
Pós-condições Um novo paciente é integrado ao sistema.
Pontos de Extensão N/A

Figura 20. Caso do uso 4, Conectar ao PEP.

O diagrama de análise também foi gerado e pode ser visto no apêndice A juntamente com as descrições de todos os casos de uso.

3.3 Análise e Projeto

A fase de projeto vem para concretizar melhor os artefatos gerados na fase de requisitos. Após uma análise dos casos de uso é possível começar a arquitetar a solução mais a nível de projeto para o PEP *Vault*. É nesta fase que entra a utilização do modelo arquitetural proposto pelo *PURE MVC* que foi apresentado no capítulo 2. Também nesta fase é adaptado o projeto aos padrões *WCF*, para que as funcionalidades do *web service* sejam atendidas. Os artefatos envolvidos nesta etapa interferem diretamente durante a programação do sistema, são definidas as classes e relacionamentos entre os componentes que serão criados no código fonte do *software*.

3.3.1 Arquitetura

A aplicação PEP *Vault* foi construída na forma de serviços *web*, desta forma, toda aplicação que consiga se comunicar com este tipo de serviço poderá utilizá-la. O *framework WCF* foi desenvolvido para facilitar a criação de aplicações orientadas a serviço, como comentado no capítulo 2, e devido a este motivo foi utilizado na construção dos serviços desta aplicação. Toda a complexa tarefa de realizar a comunicação entres clientes e servidores relacionadas a *web services* é provida pelo *WCF*, deixando tarefas que não são objetivos do projeto fora do escopo e fazendo com que o projeto seja mais concentrado da solução de prontuário eletrônico. Porém como a aplicação é contruída em cima do *WCF*, são necessárias algumas etapas de modelagem.

Ao modelar o *web service*, é necessário definir os contratos disponíveis para uso externo, ou seja, as chamadas de funções que vão ser consumidas por aplicações de terceiros, estes contratos são chamados contratos de operação. Um contrato de operação consiste em definir uma função, suas entradas de dados e sua saída de dados, para que os clientes que irão consumir o serviço sigam estas diretrizes definidas. Estas funções expostas devem garantir que os casos de uso sejam incorporados e manter-se fiel à regra de negócio. No PEP *Vault, estes* contratos foram definidos utilizando interfaces, seguido as diretrizes do *WCF*, e todos os contratos definidos foram incorporados à uma fachada responsável por

concentrar todas funções de acesso ao sistema. Desta forma, essa fachada é a porta de entrada ao sistema e está diretamente ligada à aplicação *PURE MVC* do sistema.

Foi criada uma interface de definição de contrato de serviço do *web service* para cada tipo de dado manipulado pela aplicação. As interfaces criadas foram para o Usuário, Paciente, além de cada uma para cada dado de saúde do sistema. Por exemplo, para a entidade do usuário existe uma interface *IUserService* que detalha as operações disponíveis.

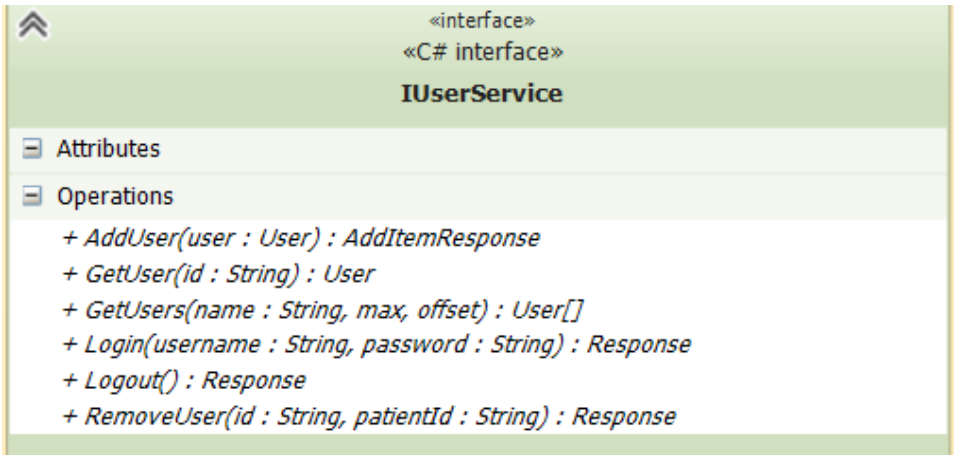


Figura 21. Contrato de serviço de usuários, *IUserService*.

As funções de manipulação de dados descritas na interface são implementadas pela fachada de serviço chamada *ServiceFacade*, com isso é esta entidade responsável por receber as requisições clientes quando consomem o serviço. Cada dado novo definido no sistema, e que será enviado ou recebido durante a comunicação com o *web service*, é definido como um contrato de dados. Esta definição será necessária para que os clientes consigam entender os dados recebidos e possa tambpem enviar dados ao servidor.

Na Figura 22 é possível observar todos os contratos definidos para o sistema; também é possível notar duas classes, *Response* e *AddItemResponse*, que são utilizadas como saída de alguns métodos e por isso são exemplos de contratos de dados.

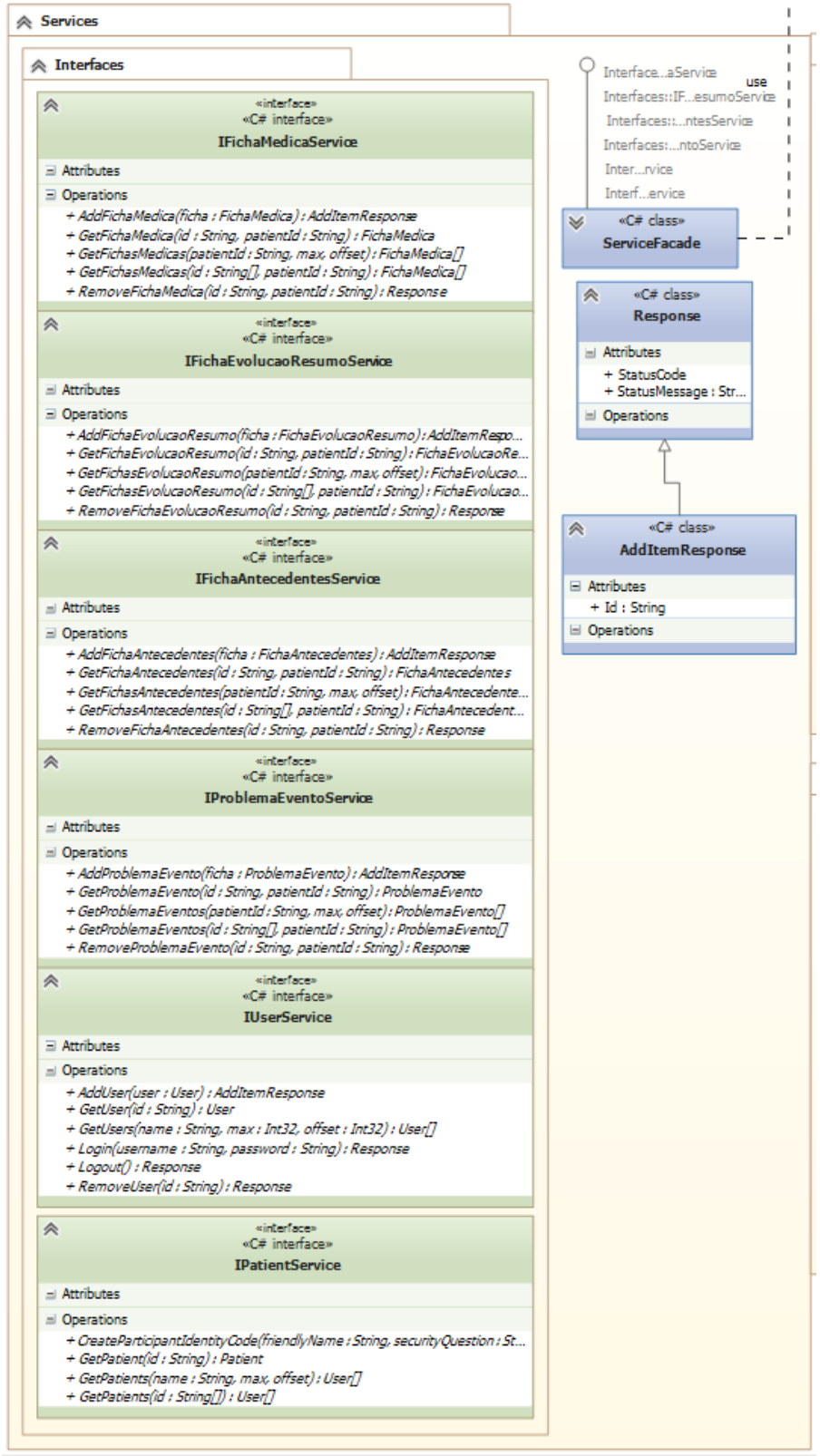


Figura 22. Contratos de serviço do sistema.

Após as definições dos serviços da aplicação parte-se para a representação do negócio do sistema aplicando os conceitos da arquitetura do *framework PURE MVC*. Como é uma arquitetura em camadas, cada camada foi criada no projeto, com exceção da camada *View* que é responsável pela interação com agentes externos ao sistema. Neste projeto, esta camada é representada pelos contratos de operações definidos, pois como não existe interface gráfica de interação com o usuário, ao invés disso, existe interação com sistemas por meio de serviços.

A arquitetura do PEP *Vault* utilizando *PURE MVC* teve maior foco nas camadas *Model* e *Controller*, desta forma foram utilizados *Proxies* e *Commands* para nestas camadas e interligando-os através de uma *Façade*, a fachada *MVC* como foi explicado no capítulo 2. Todo acesso direto a dados com intuito de leitura é feito diretamente ao *Proxy* responsável por estes dados, sejam locais ou remotos. Toda ação ou manutenção de dados é feita utilizando um *Command*, é nele que reside a regras de negócio do sistema. A alteração de dados frequentemente é realizado utilizando um *Command* e este acessa um *Proxy*.

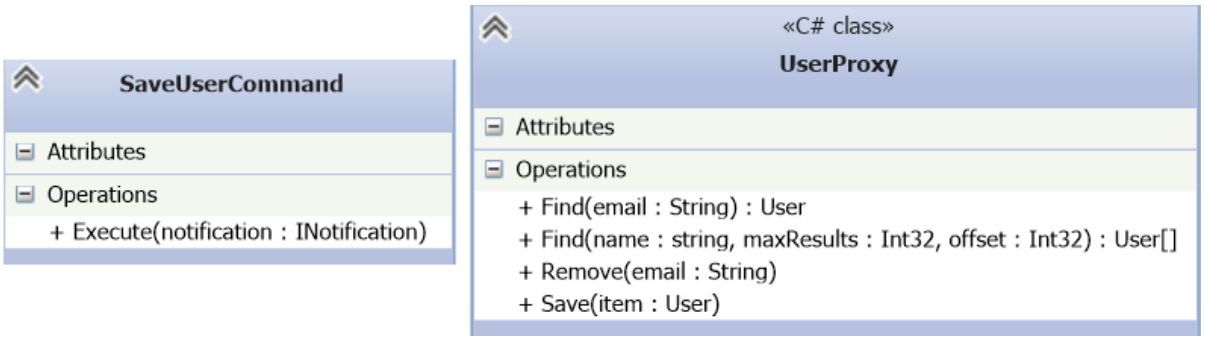


Figura 23. Exemplo de *Command* e *Proxy*.

3.3.2 Diagramas de sequência

Além dos modelos gerados acima, foram gerados também diagramas de sequência das principais funcionalidades. Estes diagramas contêm o a sequência temporal de ações, como foi explicado no capítulo 2, e visam ajudar no desenvolvimento das atividades mais complexas. Um exemplo pode ser visto na Figura 24 do diagrama de sequência do Login no sistema.

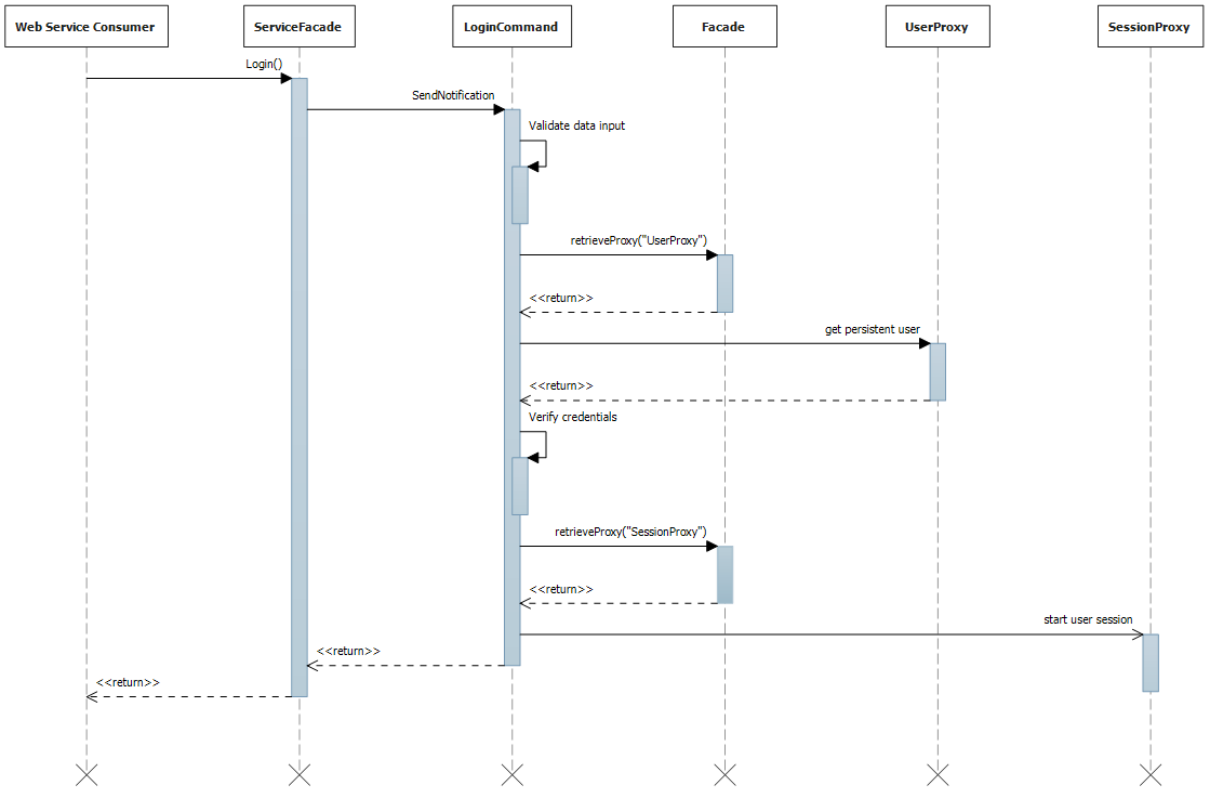


Figura 24. Diagrama de sequência do Login.

A integração do PEP *Vault* com o *HealthVault* é uma das partes mais importantes e o principal motivo do desenvolvimento deste projeto. A entrada de pacientes no sistema é realizada através de uma integração, é necessário que sejam informados dados do paciente, o nome, uma pergunta secreta, uma resposta à pergunta e um identificador do paciente para essa integração. Com todos os dados disponíveis é possível gerar uma chave chamada *Identity Code* que é responsável pela integração entre os sistemas.

Após gerar esta chave, o usuário que quer participar do PEP *Vault* deve se encaminhar ao site do *HealthVault* e utilizar esta chave como início do processo. No *HealthVault*, após a validação dessa chave, as mesmas perguntas informadas no início do processo são feitas para confirmar a identidade do paciente. Após tudo confirmado, o paciente aceita os termos de utilização do PEP *Vault* e seus dados ficam disponíveis. É importante notar que a responsabilidade do PEP *Vault* é de gerar este *Identity Code*, todo o resto do processo é feito pelo paciente e no *HealthVault*. Todo este processo descrito foi comentado no capítulo 2 é o acesso *offline*, o nome dado a este acesso é *Patient Connect*, pois necessita da interação

direta do paciente para dar as permissões sobre seus dados. A imagem seguinte demonstra esse processo através de um diagrama de sequência.

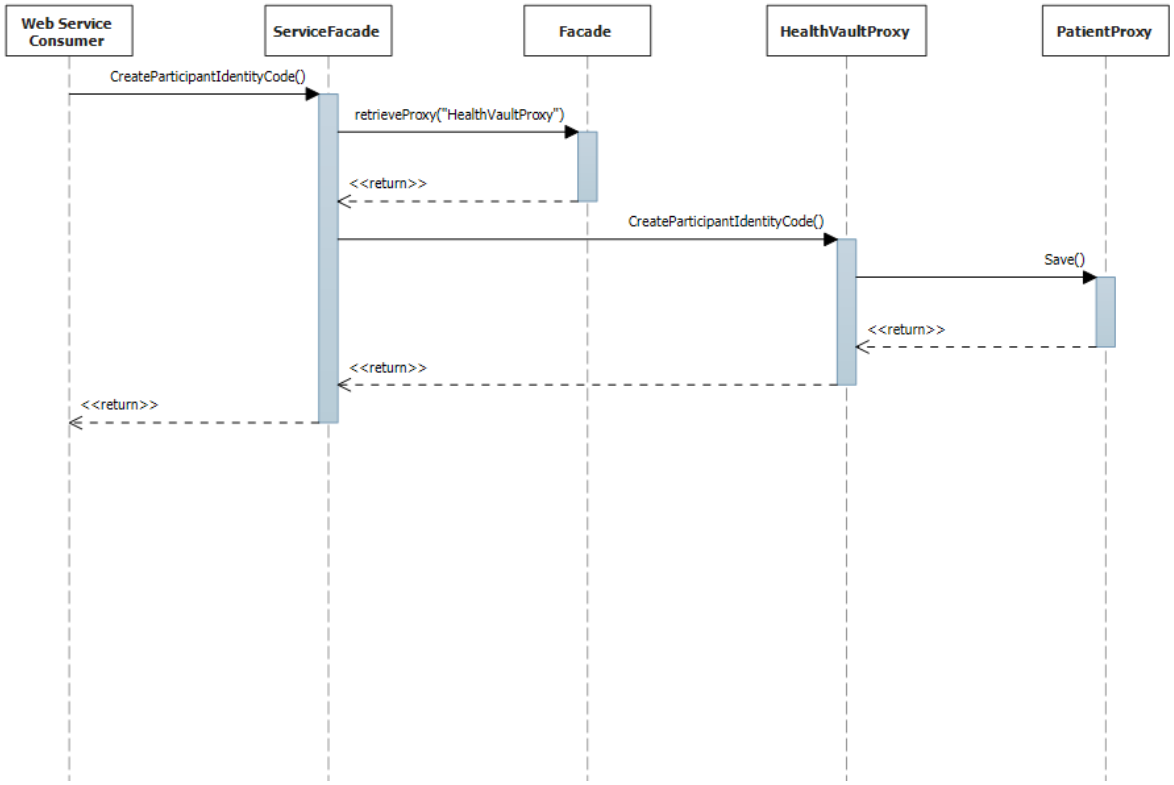


Figura 25. Diagrama de sequência Conectar ao PEP *Vault*.

Tendo em vista este processo de acesso aos dados do paciente, foram definidos dois estados para os pacientes que podem ser vistos na máquina de estados abaixo.

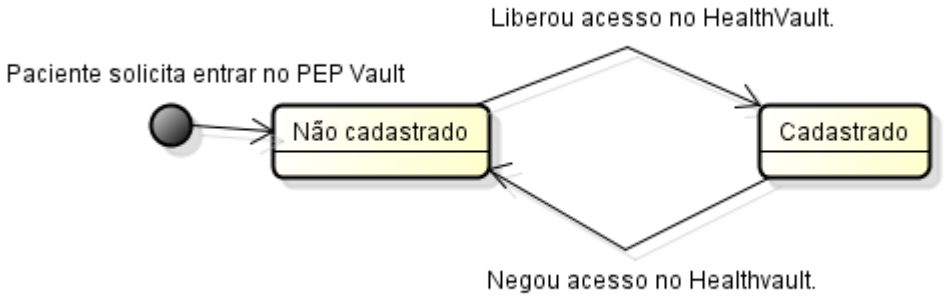


Figura 26. Máquina estados do Paciente.

É possível notar que é uma máquina de estados sem um estado final, isso se justifica pelo fato de que o paciente a qualquer momento pode remover as permissões dadas ao aplicativo e também voltar a qualquer momento.

Após o PEP *Vault* receber permissões de escrita nos dados do paciente integrado, é possível trabalhar com os dados específicos do prontuário e relacionar estes dados com o paciente. Todo dado a ser inserido no *HealthVault* é feito utilizando uma classe do *HealthVault SDK* chamada *ApplicationSpecific*. Como já foi explicado no capítulo 2, todos os dados presentes do HealthVault são descritos na forma de *XML* e a classe *ApplicationSpecific* é a responsável por transformar os objetos em *XML* no formato específico do *HealthVault*. A *Microsoft* define um *Thing*, dados que compõem um *Record*, para dados específicos de aplicações externas. No corpo do *XML* deste *Thing* existe uma *tag* onde os dados específicos vão estar concentrados, esta *tag* possui o nome de *"app-specific"*. A Figura 27 mostra um exemplo de dado específico no formato *XML* do *HealthVault*.

```
<thing>
  <thing-id version-stamp="878de6f9-485b-47e6-af84-0a2a7fccfd33">04401e5a-e810-49eb-a0d5-07b983ec94a4</thing-id>
  <type-id name="Informações específicas do aplicativo">a5033c9d-08cf-4204-9bd3-cb412ce39fc0</type-id>
  <thing-state>Active</thing-state>
  <flags>0</flags>
  <eff-date>2014-11-14T04:06:50.864Z</eff-date>
▼<created>
    <timestamp>2014-11-14T04:06:50.507Z</timestamp>
    <app-id name="pineapple">53ad69b5-126f-41a0-9bb7-431ceb02b062</app-id>
    <person-id name="Afif Fikani">bf571fd9-99b1-4cb6-8c96-186d9965a5a7</person-id>
    <access-avenue>Offline</access-avenue>
    <audit-action>Created</audit-action>
  </created>
▼<updated>
    <timestamp>2014-11-14T04:06:50.507Z</timestamp>
    <app-id name="pineapple">53ad69b5-126f-41a0-9bb7-431ceb02b062</app-id>
    <person-id name="Afif Fikani">bf571fd9-99b1-4cb6-8c96-186d9965a5a7</person-id>
    <access-avenue>Offline</access-avenue>
    <audit-action>Created</audit-action>
  </updated>
▼<data-xml>
  ▼<app-specific>
      <format-appid>53ad69b5-126f-41a0-9bb7-431ceb02b062</format-appid>
      <format-tag>ff94ff58-078d-4565-ba40-0a601f592c47</format-tag>
      <summary>FichaMedicaPaciente</summary>
    ▼<FichaMedicaPaciente>
        <PatientId>c496c35b-d935-42a6-b5fc-a09edd5f75a5</PatientId>
        <ReferralReason>Afif</ReferralReason>
      </FichaMedicaPaciente>
    </app-specific>
    <common/>
  </data-xml>
</thing>
```

Figura 27. *Thing* específico para a aplicação PEP *Vault.*

A tarefa de adicionar um dado específico requer integração entre os sistemas, para isso a plataforma da *Microsoft* necessita que se faça uma conexão e neste caso é

uma conexão *offline*. Para realizar esta conexão é preciso ter o id do paciente. Com a conexão realizada, é possível utilizar o *SDK* para adicionar novos dados específicos da aplicação. Este processo é descrito no diagrama de sequência da Figura 28.

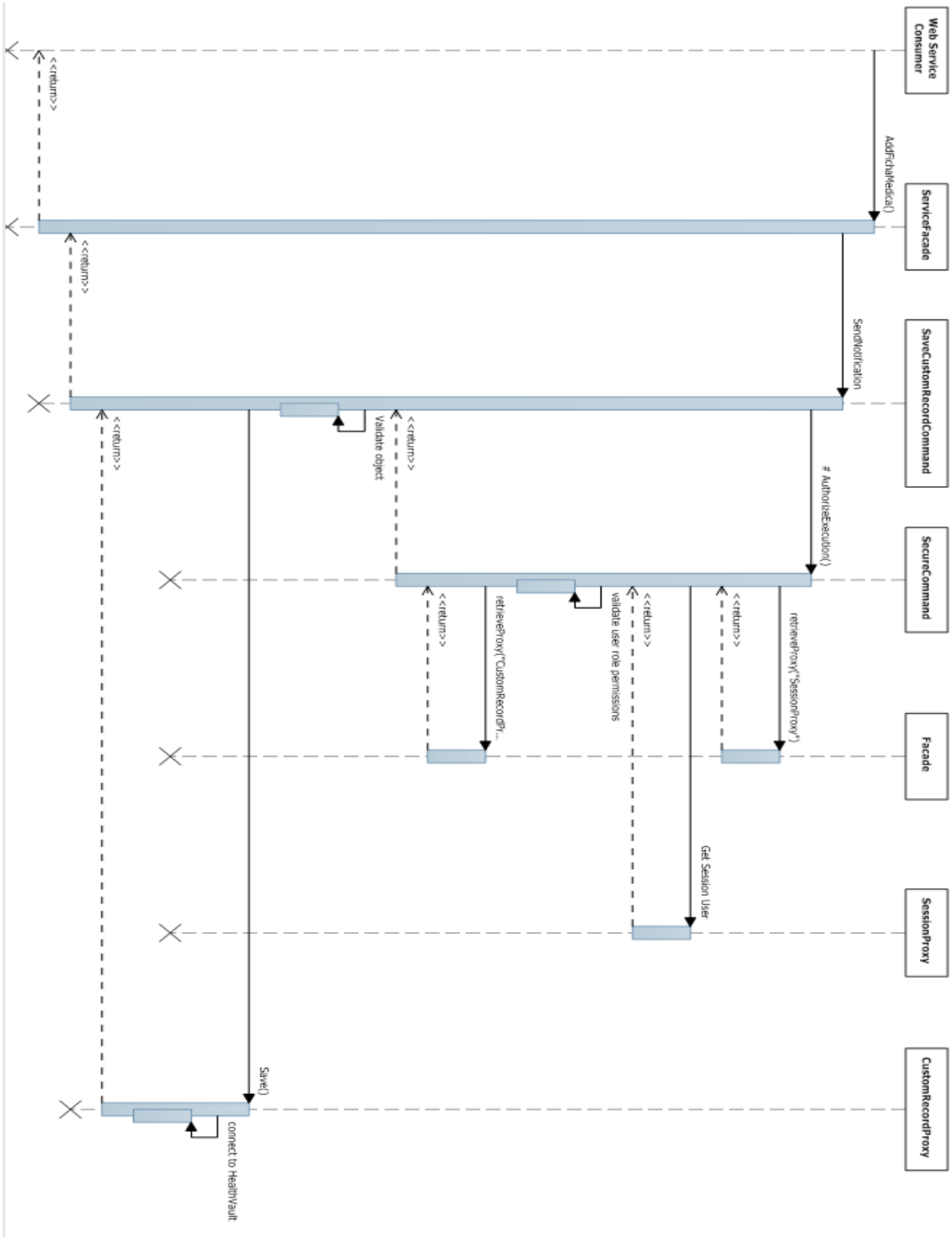


Figura 28. Diagrama de sequência adicionar Ficha Médica.

3.3.3 Banco de dados

Utilizando um banco de dados, o PEP *Vault* permite que dados sejam armazenados e acessados de forma mais robusta e rápida. Foi decidido o uso do *SQLite* como banco de dados do projeto, como foi explicado no capítulo 2.

O PEP *Vault* guarda dados do paciente referentes ao processo de entrada no sistema, basicamente os dados do processo do *Patient Connect*. Todo o os dados do login também são guardado em banco de dados local sem integrações com outros sistemas. Levando em consideração a arquitetura do sistema, isso quer dizer que os *Proxies* responsáveis pelos dados usuário e do paciente implementam uma conexão com o banco *SQLite* e persistem este dados no banco de dados. A última etapa da Figura 25 ilustra parte deste processo em um diagrama de sequência e a Figura 29 mostra o fluxo de dados e onde cada dado está mantido, todos os dados referentes à saude são guardados utilizando uma integração com o *HealthVault* e os dados que são referentes ao Login e o início do processo *PatientConnect* sao guardados no banco de dados local.

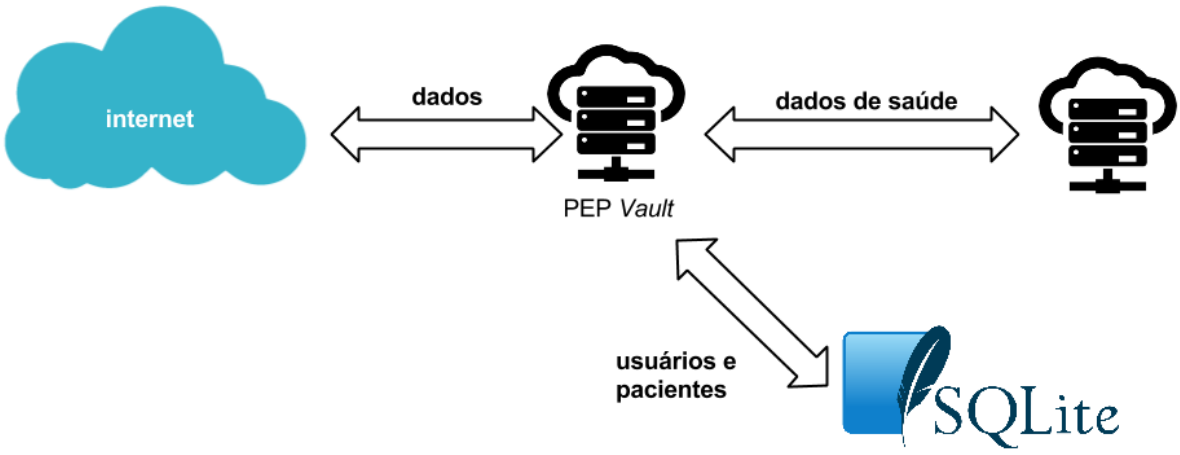


Figura 29. Fluxo de dados no PEP *Vault*.

Como é demonstrado na Figura 29, no banco de dados local estão presentes apenas as entidades usuário e paciente, todas as outras informações estão mantidas no *HealthVault*. Todos os relacionamentos entre os dados de saúde e os

pacientes são realizados pelo sistema da *Microsoft* e o acesso à esses relacionamentos é ralizado através do uso do *SDK*.

Capítulo 4

Conclusão e Trabalhos Futuros

Com o desenvolvimento do PEP *Vault* se avança uma etapa da evolução do PEP do NUTES UPE e é uma nova abordagem permitindo que muitos caminhos possam ser seguidos a partir deste ponto. Existem agora novas oportunidades de novos sistemas utilizando o PEP *Vault* como base, pois trata-se de uma plataforma na área de saúde e utilizando a metodologia *POMR*.

Na visão do desenvolvedor, são grandes as facilidades realizadas pelo PEP *Vault*. Com a implementação do projeto, a regra de negócio envolvendo *POMR* já fica disponível para ser consumida. O foco do desenvolvimento passa a ser na interface com o usuário final e na integração com o sistema por meio de *web services*.

Pela forma como o serviço foi projetado há diversas possibilidades de utilização da plataforma. Não há restrições quanto à tecnologia que irá consumir o PEP *Vault*, desde que consiga consumir *web services*, podendo assim atingir plataformas como a móvel e outros dispositivos capazes de utilizar a *internet*. Este é um avanço importante do projeto, pois abre portas para as plataformas móveis que são de grande valia para futuros projetos.

Como trabalhos futuros, pretende-se adicionar os dados de saúde do PEP do NUTES não contemplados durante este projeto. Transformar o PEP *Vault* em uma plataforma mais completa, capaz de aceitar registro de aplicações que o utilizam como base. Utilizar os dados deste projeto de forma a acrescentar e ajudar profissionais de saúde, tornando o PEP em uma central de dados de saúde. Melhorar a arquitetura do sistema afim de lançar o sistema de forma oficial e disponibilizar a aplicação na loja de aplicativos da Microsoft.

Com a conclusão do PEP *Vault*, tem-se como trabalho futuro criar uma aplicação *web* de prontuário eletrônico que use a plataforma deste projeto, capaz de ser utilizada comercialmente em clínicas médicas. Além disso, uma aplicação móvel também é prevista, aproveitando que o foco do desenvolvimento poderá ser maior na interface com o usuário, sem maior preocupações com as regras já implementadas. Desta forma, tem-se como trabalho futuro criar uma aplicação mais completa e multiplataforma para ser comercializada entre profissionais de saúde em qualquer lugar e momento.

Como limitações deste trabalho é possível citar a utilização da plataforma *HealthVault*. Todos os pacientes que vão utilizar o PEP *Vault* devem antes possuir um cadastro no sistema da *Microsoft*, então isso quer dizer que é preciso inserir o sistema da *Microsoft* em um ambiente antes de inserir o PEP *Vault*. Além disso, o processo de entrada de clientes no PEP *Vault* é através do processo nomeado *PatientConnect* e não é um processo tão simples, exige um conhecimento básico de informática. Este fato também dificulta a inserção deste projeto em ambientes de saúde pública, tendo em vista a falta de acesso à tecnologia da informação pela sociedade mais carente que é atendida na saúde pública.

Um outro problema encontrado durante o desenvolvimento foi com a integração com o *HealthVault*, apesar da documentação ser vasta e de fácil acesso, alguns detalhes mais específicos de programação não estão bem explícitos. Pode-se citar novamente o processo de *PatientConnect*, só foi conseguido realizar este processo utilizando projetos de exemplos, ou seja, foi feita uma análise do código destes projetos e posteriormente realizado o desenvolvimento no PEP *Vault*.

Bibliografia

[1] JING-SONG LI et al. **Design and Development of EMR Supporting Medical Process Management**. Journal of Medical Systems, 2012, Vol.36(3), pp.1193-1203

[2] FEINSTEIN, A R. **The problems of the "problem-oriented medical record".** Annals of internal medicine, 1973, Vol.78(5), pp.751-62

[3] KOZAK, ELIZABETH et al. **Deciphering the physician note**. Journal of General Internal Medicine, 1994, Vol.9 (1), pp.52-54.

[4] ROCKART, J. F. **Medical record storage: a method for determining level of activity**. Health services research, 1972, Vol.7(4), pp.276-87

[5] Oliveira, S. V. W. B. et al. **Use And Development Of Health Information Systems: The Experience Of An Organizational Unit Responsible For The Technological Services At A Public Hospital**. School of Economics, Business and Accounting of Ribeirao Preto – USP, São Paulo, Brazil

[6] EYSENBACH, GUNTHER. **Medicine 2.0: Social Networking, Collaboration, Participation, Apomediation, and Openness**. Journal of Medical Internet Research, 2008, Vol.10(3)

[7] KRUCHTEN, PHILIPPE. **The Rational Unified Process: An Introduction. Addison-Wesley**. 2003.

[8] LALLCHANDANI, JAIPRAKASH. **A Dynamic Slicing Technique for UML Architectural Models**. IEEE Transactions on Software Engineering [0098-5589] yr:2011 vol:37 iss:6 pg:737 -771

[9] D. GASEVIC, N. KAVIANI, AND M. HATALA, **On Metamodeling in Megamodels, Model Driven Eng. Languages and Systems**. Springer, Sept. 2007.

[10] T. GIRBA, J.-M. FAVRE, AND S. DUCASSE, **Using Meta-Model Transformation to Model Software Evolution.** Electronic Notes in Theoretical Computer Science, vol. 137, no. 3, pp. 57-64, Sept. 2005

[11] *HealthVault*. Disponível em: < http://msdn.microsoft.com/en-US/healthvault >. Acesso em: 19/11/2014.

[12] *HealthVault Data Types*. Disponível em: < https://developer.healthvault.com/DataTypes >. Acesso em: 19/11/2014.

[13] KRUCHTEN, P. **Agility with the RUP**. Cutter IT Journal, The Journal of Information Technology Management, [S.l.], v.14, n.12, p. 27 – 33, Dec. 2001.

[14] CINTRA, C. C. **A implementação de um processo de engenharia de requisitos baseado no Processo Unificado da Rational (RUP) alcançando nível 3 de Maturidade da Integração de Modelos de Capacidade e Maturidade (CMMI) incluindo a utilização de práticas de métodos ágeis**. 2006. 160 f. Dissertação - Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação. 2006.

[15] OEHM, B. **A Spiral Model of Software Development and Enhancement**. INTERNATIONAL WORKSHOP ON SOFTWARE PROCESS AND SOFTWARE ENVIRONMENTS, 1985, Trabuco Canyon, US. [S.l. : s.n.], 1985.

[16] KARETSOS, S. et al. **Modeling an e-government observatory for rural SMEs using UML with RUP**. Operational research [1109-2858], yr:2011 vol:11 iss:1 pg:59 -75.

[17] BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Editora Campus. 2006, p.: 45, 46.

[18] MASSAD, E. et al. **O prontuário eletrônico do paciente na assistência, informação e conhecimento médico**. São Paulo: H. de F. Marin, 2003.

[19] OSELKA, G . **Prontuário Médico**. Revista da Associação Médica Brasileira, Directory of Open Access Journals (DOAJ) 2002, Vol.48(4), p.286.

[20] FARIAS, J. S. **Inovação em hospitais do Brasil e da Espanha: a percepção de gestores sobre o prontuário eletrônico do paciente**. BBR : Brazilian Business Review, 2012, Vol.9(3), p 25.

[21] Liao, L. **A Novel Web-enabled Healthcare Solution on HealthVault System**. Journal of Medical Systems, 2012, Vol.36(3), pp.1095-1105.

[22] ***Windows Communication Foundation***. Disponível em : < http://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx >. Acessado em 22/11/2014.

[23] HALL, C. ***ActionScript Developer's Guide to PureMVC***. O'Reilly Media, 2011. P 264.

[24] ***PURE MVC***. Disponível em: < http://puremvc.org/content/view/67/178/ >. Acessado em 22/11/2014.

[25] Gagan, M. J. **The SOAP format enhances communication: the SOAP format provides a clear and concise way of documenting patient information**. Kai Tiaki: Nursing New Zealand. 15.5 (June 2009): p 15.

[26] Jeon, S. et al. **A recovery method of deleted record for SQLite database**. Personal and ubiquitous computing [1617-4909] yr:2012 vol:16 iss:6 p 707 - 715.

[27] DATE, C. J. **INTRODUÇÃO A SISTEMAS DE BANCOS DE DADOS**. 7. ed. Rio de Janeiro: Elsevier, 2001.

[28] ELMASRI, Ramez; NAVATHE, Shamkant B. **SISTEMAS DE BANCO DE DADOS.** 4. ed. São Paulo: Addison Wesley, 2006.

[29] OGURA, D. R. **Uma Metodologia Para Caracterização De Aplicações Em Ambientes De Computação Nas Nuvens.** 2011, 124f, Dissertação – Escola Politécnica da Universidade de São Paulo. Programa de Pós-Graduação em Engenharia Elétrica.

Apêndice A
Requisitos do PEP *Vault*.

Identificação dos Requisitos

Por convenção, a referência a requisitos é feita através do identificador do requisito seguido da descrição da funcionalidade, de acordo com o esquema abaixo:

[Identificador do requisito – Descrição da funcionalidade]

Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”, “importante” e “desejável”.

Essencial é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.

Importante é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.

Desejável é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

Requisitos Funcionais (RF)

Os itens a seguir descrevem os requisitos funcionais do projeto Associações. Cada RF possui por padrão uma prioridade de disponibilidade.

RF01 – Gerenciar Usuários

Descrição	O sistema deve possuir um Usuário Administrador já presente no sistema. Este usuário é responsável por gerenciar outros usuários.
Prioridade	Essencial

RF02 – Gerenciar Pacientes

Descrição	O cadastro do paciente deve ser realizado integrado com o sistema *HealthVault* da *Microsoft*, é deste sistema que os Pacientes vem ao sistema PEP *Vault*. O cadastro no PEP *Vault* significa o paciente dar acesso de leitura e escrita de suas informações pessoais e de saúde que estão contidas no *HealthVault*. É necessário um cadastro prévio no *HealthVault* e após isso a realização de uma comunicação entre os sistemas. A aquisição de novos Pacientes deve ser realizada diretamente entre o PEP *Vault* e o próprio paciente, de forma automatizada. O processo é iniciado no PEP *Vault* e realizando um encaminhamento para o *HealthVault*. Após a liberação de acesso aos dados do paciente o sistema deve ser capaz de utilizá-lo. A remoção do acesso do PEP aos dados do paciente é realizada pelo próprio, utilizando o *HealthVault*.
Prioridade	Essencial

RF03 – Acessar histórico de fichas médicas

Descrição	O sistema deve permitir o acesso a todas as fichas médicas já cadastradas para um paciente. Fichas médicas finalizadas não podem ser modificadas. As fichas existentes não podem ser fisicamente excluídas do sistema, apenas logicamente. As fichas médicas são dados definidas pelo PEP, sendo assim, não são dados já existentes no sistema HealthVault. Todas as fichas devem ser integradas e guardadas no HealthVault
Prioridade	Desejável

RF04 – Visualizar Problemas ou Eventos

Descrição	O sistema deve permitir o acesso ao histórico de Problemas ou Eventos reportados por profissionais de saúde. Cada item da lista de Problemas ou Eventos deverá levar o usuário à sua respectiva ficha de Evolução e Resumos.
Prioridade	Essencial

RF05 – Cadastrar Ficha Médica do Paciente

Descrição	O sistema deve permitir o preenchimento de uma nova Ficha Médica do Paciente. Textos de instruções de preenchimento devem ser mostrados como auxilio ao usuário.
Prioridade	Essencial

RF06 – Cadastrar Ficha Antecedentes

Descrição	O sistema deve permitir o preenchimento de uma nova Ficha Antecedentes. Textos de instruções de preenchimento devem ser mostrados como auxilio ao usuário.
Prioridade	Essencial

RF07 – Cadastrar Ficha Evolução e Resumos

Descrição	O sistema deve permitir o preenchimento da Ficha Antecedentes. Textos de instruções de preenchimento devem ser mostrados como auxilio ao usuário. Cada ficha preenchida em Evolução e Resumo deve ser mostrada como um problema ou evento na tela de Problemas ou Eventos.
Prioridade	Essencial

RF08 – Logar no sistema

Descrição	O acesso ao sistema deve ser através de *Login* de usuário. Cada usuário acessará as áreas específicas de seu tipo. As fichas de saúde devem ser de acesso restrito aos profissionais de saúde. O cadastro de usuários do PEP deve ser de uso restrito do Administrador.
Prioridade	Essencial

Especificação dos Casos de Uso

Atores

Nome	Descrição	Responsabilidades
Paciente	Paciente	Se cadastrar no sistema
Administrador	Administrador do sistema	Criar contas de usuários

Médico	Profissional de saúde que atenderá o paciente	Gerenciar Pacientes e fichas médicas dos pacientes.

UC001 - Efetuar Login
Atores: Administrador, Médico.
Breve Descrição Ocorre quando um usuário acessa o site e não existe sessão iniciada. Desta forma é necessário realizar o *Login* no sistema.
Fluxo de Eventos **Fluxo Básico** 1. O usuário acessa o sistema; 2. O usuário preenche os campos de *Login* e clica no botão de *Login*; 3. O sistema valida as credenciais entradas. 4. O acesso do usuário ao sistema é liberado. **Fluxos Alternativos** **N/A**
Exceções: 1. Dados do usuário incorretos; 2. O usuário corrige os dados; 3. O sistema valida as credenciais entradas. 4. O acesso do usuário ao sistema é liberado.

Precondições

Não ter sessão iniciada previamente.

Pós-condições

O usuário está logado.

Pontos de Extensão

N/A

UC002 - Manter Usuário

Atores: Administrador

Breve Descrição

Ocorre quando há necessidade de criar, modificar ou remover usuários do sistema.

Fluxo de Eventos

Fluxo Básico

1. O usuário acessa a área de gerenciamento de usuários;
2. O usuário realiza uma ação de criar novo usuário;
3. As alterações são salvas;

Fluxos Alternativos

1. O usuário acessa a área de gerenciamento de usuários;
2. Usuário busca por usuário já existente;
3. O usuário realiza uma ação entre: Remover e Modificar;
4. As alterações são salvas;

Exceções:

1. Usuário adicionou usuário com identificador único já existente em outro usuário;
2. Usuário corrige valores errados;
3. As alterações são salvas;

Precondições

UC001.

Pós-condições

A alteração de usuário está salva.

Pontos de Extensão

N/A

UC003 - Buscar Pacientes

Atores: Médico.

Breve Descrição

Realiza a busca no banco de dados de pacientes.

Fluxo de Eventos

Fluxo Básico

1. Usuário acessa área de pacientes;
2. Preenche o nome no campo de busca;
3. O sistema mostra os pacientes encontrados;

Fluxos Alternativos

1. Usuário acessa área de pacientes;
2. Preenche o identificador no campo de busca;
3. O sistema mostra os pacientes encontrados;

Exceções:
N/A

Precondições UC001.
Pós-condições N/A
Pontos de Extensão N/A

UC004 - Conectar ao PEP

Atores: Paciente

Breve Descrição

Realiza a integração de um paciente do *HealthVault* ao PEP *Vault*.

Fluxo de Eventos

Fluxo Básico

1. O usuário acessa a área de cadastro de pacientes;
2. O usuário preenche os campos;
3. O usuário confirma a ação uma de conectar ao PEP *Vault*;
4. O sistema gera o código de integração com o *HealthVault*;
5. O sistema redireciona o Paciente ao *HealthVault*.
6. O usuário efetua o *Login* no HealthVault e realiza o processo de integração com o PEP;
7. O sistema consegue ter acesso ao Paciente.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

Possuir um cadastro no HealthVault.

Pós-condições

Um novo paciente é integrado ao sistema.

Pontos de Extensão

N/A

UC005 - Selecionar Paciente

Atores: Médico.

Breve Descrição

O usuário seleciona um paciente para realizar uma consulta e criações de fichas.

Fluxo de Eventos

Fluxo Básico

1. O usuário indica o identificado do paciente;
2. O paciente é selecionado.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC003.

Pós-condições

O paciente é selecionado para a consulta.

Pontos de Extensão

N/A

UC006 - Visualizar Fichas

Atores: Médico.

Breve Descrição

O usuário seleciona uma ficha previamente feita e analisa as informações.

Fluxo de Eventos

Fluxo Básico

1. O acessa a página da ficha que deseja observar;
2. O usuário clica em uma ficha listada.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC004.

Pós-condições

N/A

Pontos de Extensão

N/A

UC007 - Visualizar Problema ou Evento

Atores: Médico.

Breve Descrição

O usuário seleciona um problema ou evento anterior para analisar as informações

Fluxo de Eventos

Fluxo Básico

1. O acessa a página de Problemas ou Eventos;
2. O usuário clica em um Problema ou Evento listado;
3. A ficha é mostrada na tela de Evolução e Resumos.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC005.

Pós-condições

N/A

Pontos de Extensão

N/A

UC008 - Manter Ficha Médica do Paciente

Atores: Médico.

Breve Descrição

O usuário manipula a ficha Médica do Paciente criando novas informações para o sistema.

Fluxo de Eventos

Fluxo Básico

1. O acessa a página da Ficha Médica do Paciente;
2. O usuário preenche os campos com as informações do paciente;
3. Usuário salva a ficha.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC005.

Pós-condições

Nova ficha é criada para o paciente.

Pontos de Extensão

N/A

UC009 - Manter Ficha Antecedentes

Atores: Médico.

Breve Descrição

O usuário manipula a ficha Antecedentes criando novas informações para o sistema.

Fluxo de Eventos

Fluxo Básico

1 O acessa a página da Ficha Antecedentes;
2 O usuário preenche os campos com as informações do paciente;
3 Usuário salva a ficha.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC005.

Pós-condições

Nova ficha é criada para o paciente.

Pontos de Extensão

N/A

UC010 - Manter Ficha Evolução e Resumos

Atores: Médico.

Breve Descrição

O usuário manipula a ficha Evolução e Resumos criando novas informações para o sistema.

Fluxo de Eventos

Fluxo Básico

1 O acessa a página da Ficha Evolução e Resumos;
2 O usuário preenche os campos com as informações do paciente;
3 Usuário salva a ficha.

Fluxos Alternativos

N/A

Exceções:
N/A

Precondições

UC005.

Pós-condições

1 Nova ficha é criada para o paciente.
2 Nova entrada referente à nova ficha é criada na página de Problema ou evento.

Pontos de Extensão

N/A

Diagrama de casos de uso

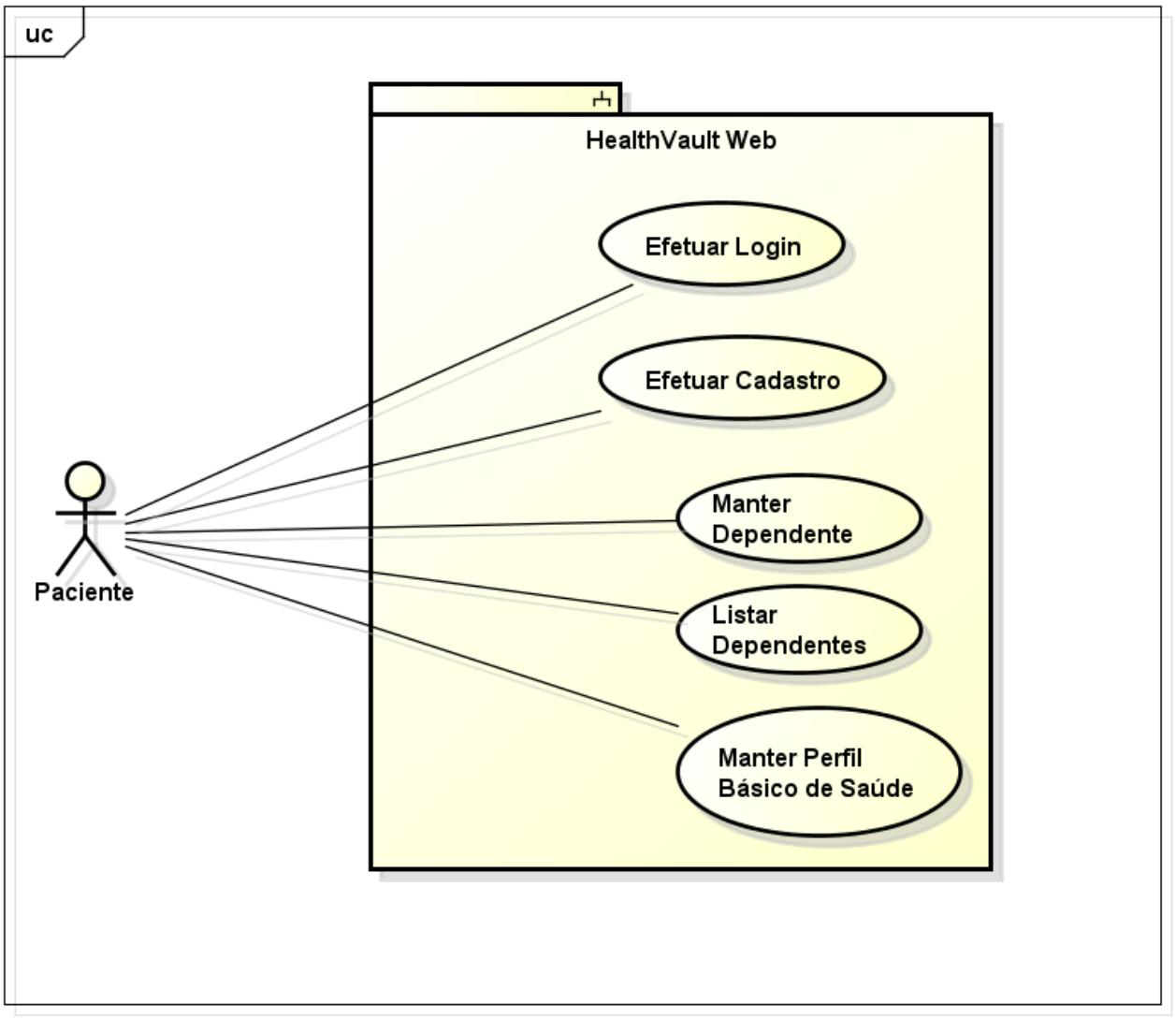


Figura 30. Diagrama de casos de uso *HealthVault*.

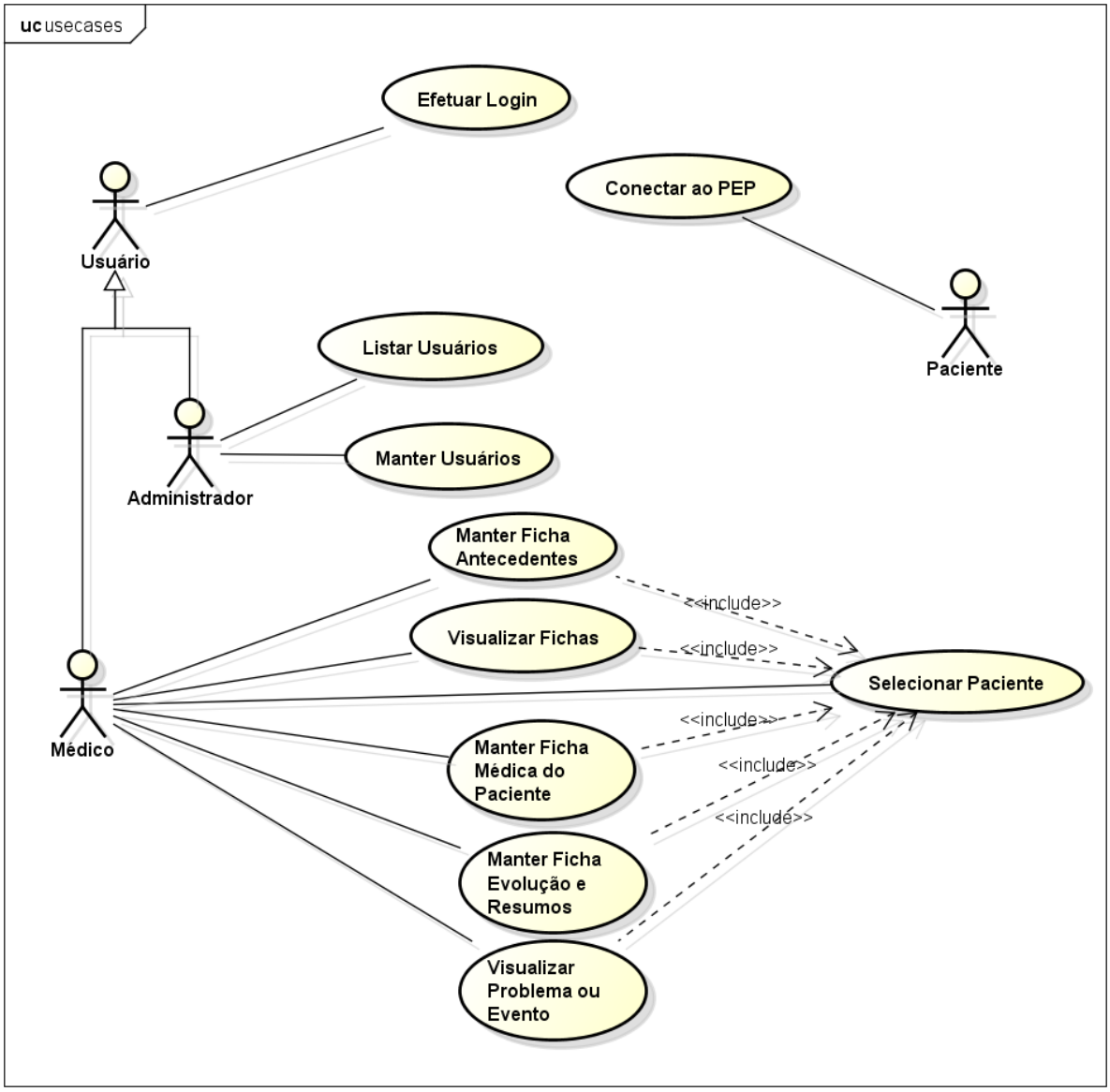


Figura 31. Diagrama de casos de uso do PEP *Vault.*

Diagrama de análise

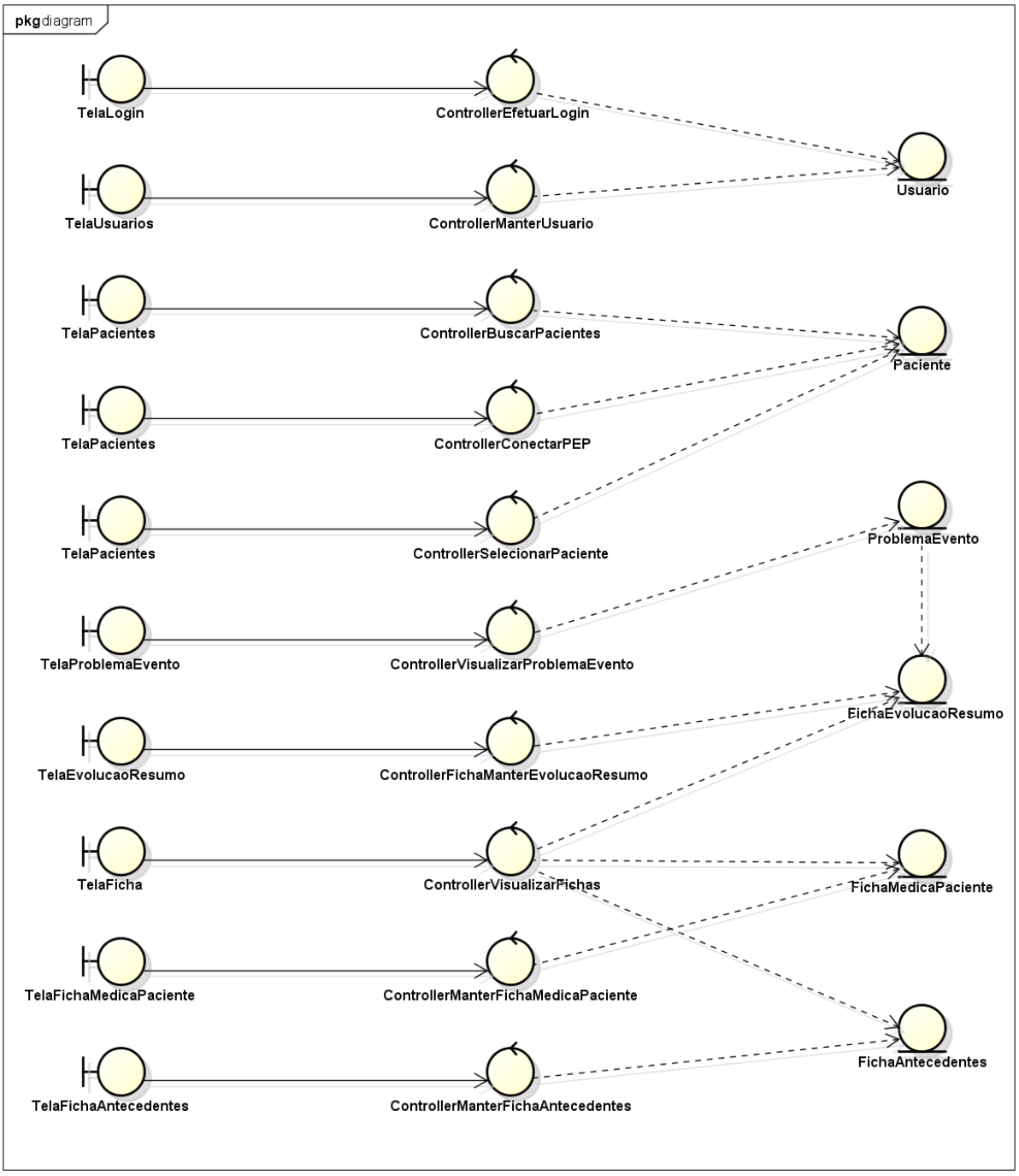
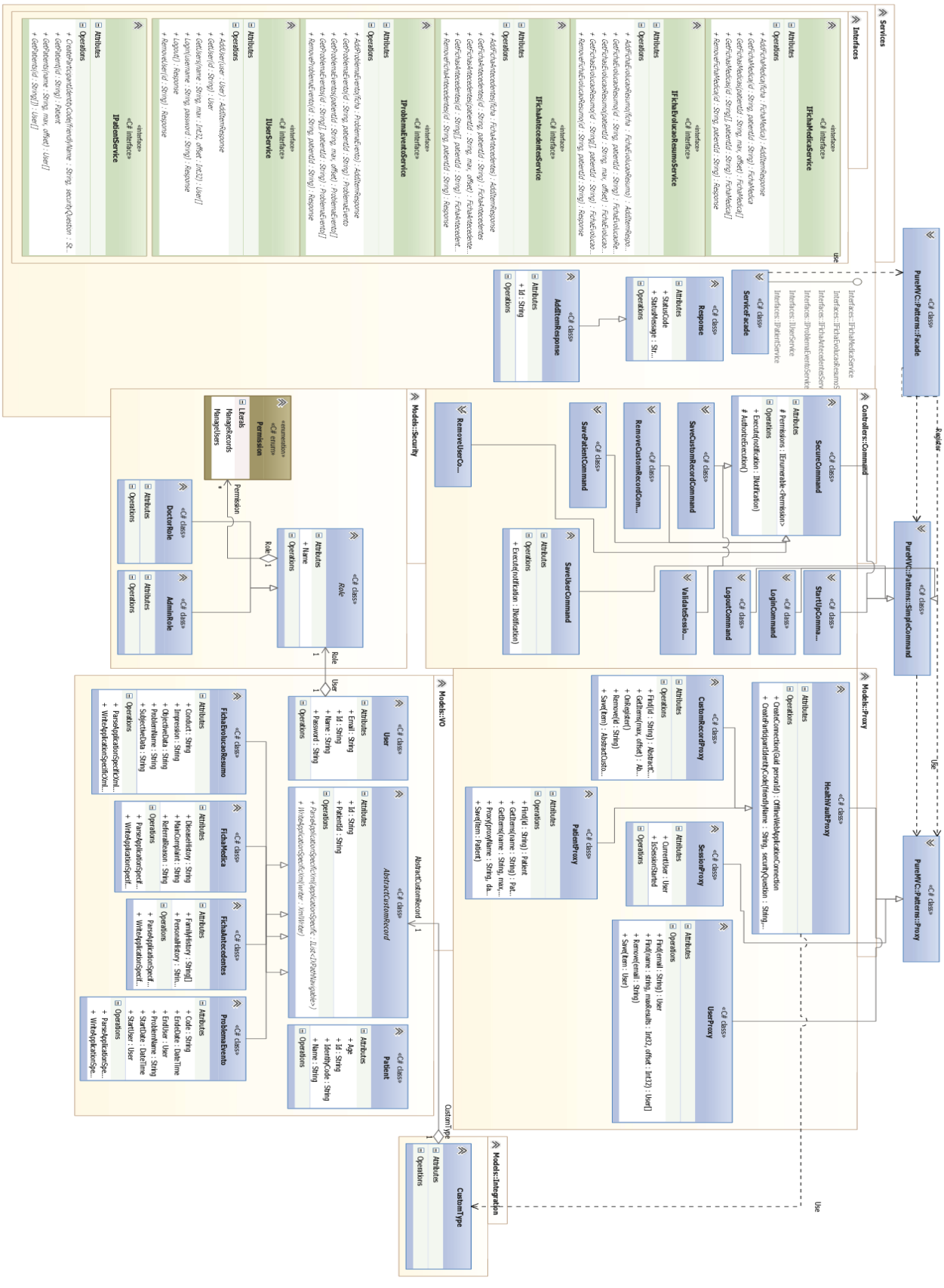


Figura 32. Diagrama de análise

Diagrama de classes



Diagramas de Sequência

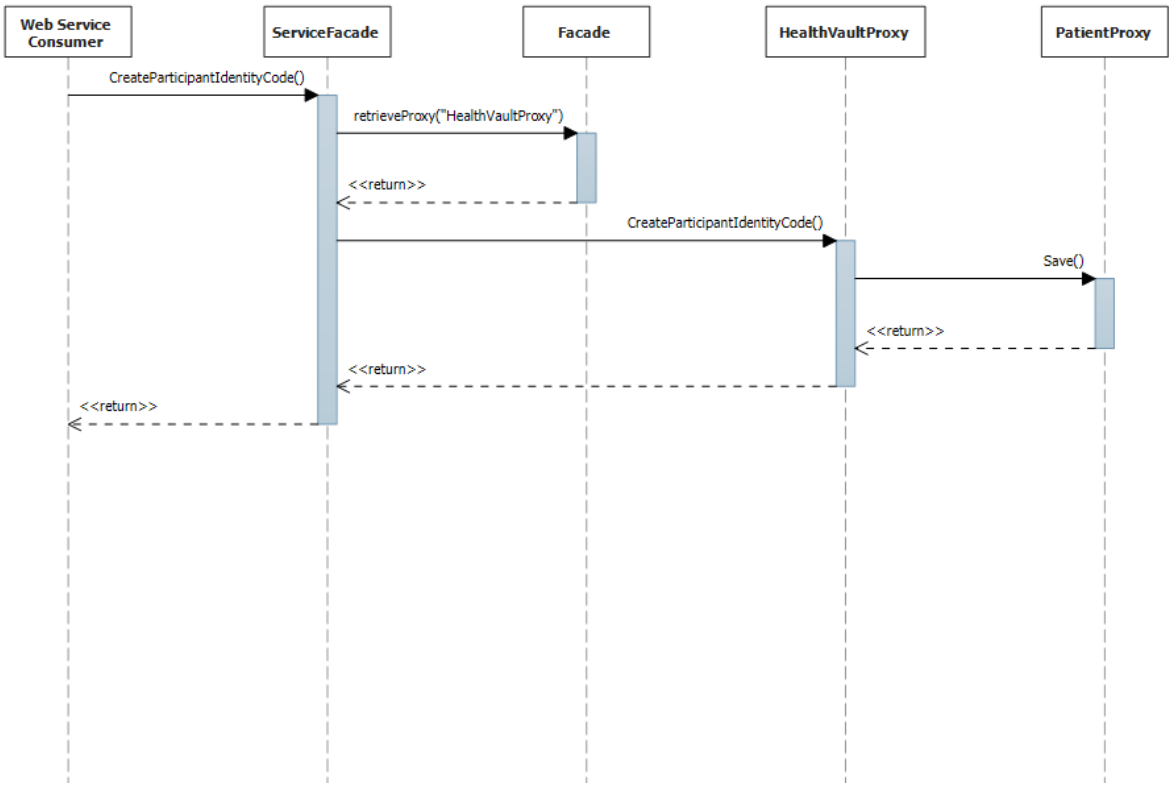


Figura 33. Diagrama de sequência Conectar ao PEP *Vault*.

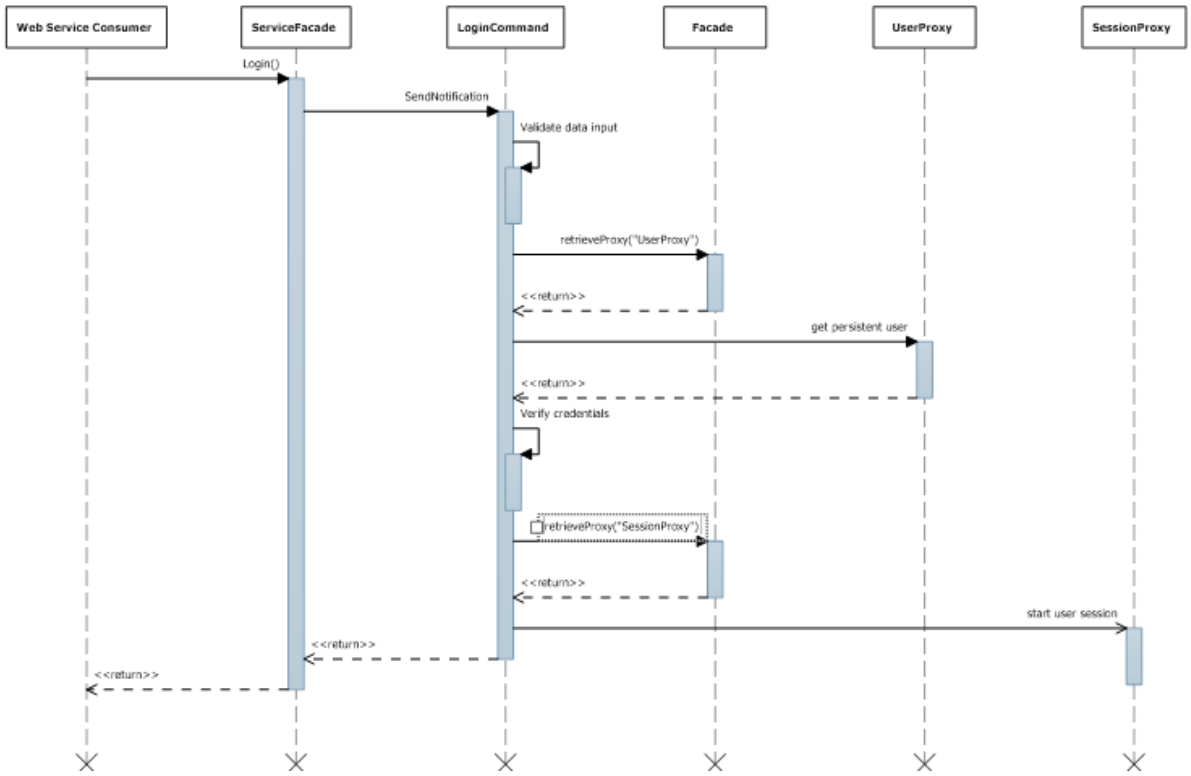


Figura 34. Diagrama de sequência do Login.

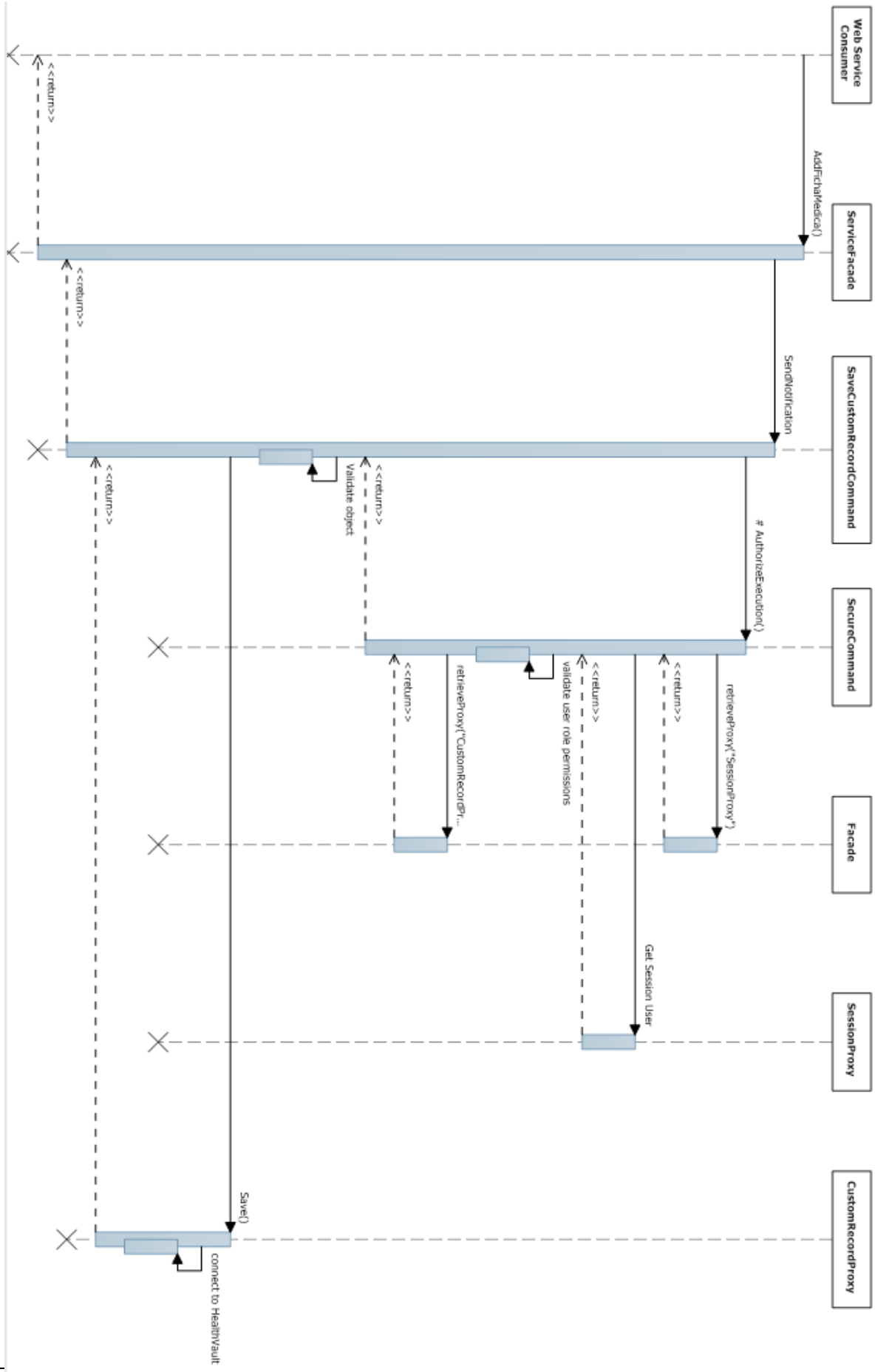

Web Service Consumer
ServiceFacade
SaveCustomRecordCommand
SecureCommand
Facade
SessionProxy
CustomRecordProxy
AddFichaMedica()
<<return>>
Send@Notification
Validate object
AuthorizeExecution()
retrieveProxy("CustomRecordPr...
validate user role permissions
retrieveProxy("SessionProxy")
Get Session User
Save()
connect to HealthVault

Figura 35. Diagrama de sequência adicionar Ficha Médica.

Diagramas de Estado