UNIVERSIDADE FEEVALE

GUSTAVO TROTT

ESTUDO E APLICAÇÃO DE UM PROCESSO DE MDD UTILIZANDO IFML

Novo Hamburgo

2013

GUSTAVO TROTT

ESTUDO E APLICAÇÃO DE UM PROCESSO DE MDD UTILIZANDO IFML

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale

Orientador: João Batista Mossmann

Novo Hamburgo

2013

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

Aos amigos e às pessoas que convivem comigo diariamente, minha gratidão, pelo apoio emocional - nos períodos mais difíceis do trabalho.

RESUMO

Ao passar dos anos a modelagem de software tem crescido de forma exponencial. Cada vez mais empresas observam os benefícios de se utilizar a modelagem desde a concepção de um projeto, evitando problemas ao longo da execução. Criado pela OMG (Object Management Group) em 1997, a UML (Unified Modeling Language) consolidou-se como sendo a linguagem padrão de mercado e hoje é conhecida e utilizada em pelo menos 70% das organizações de desenvolvimento de software no mundo todo. Sabe-se que as empresas de desenvolvimento de software preocupam-se em aumentar a produtividade da equipe, bem como a qualidade do produto, e para isso novas soluções tem surgido. Dentre estas destaca-se as DSMs (Domain-specific modeling), linguagens de modelagem criadas para um domínio específico que permitem atingir alto nível de abstração. Possibilitando notações específicas de um domínio, ao contrário da UML que foi desenvolvida para atender diversos setores da engenharia de software e muitas vezes causa perda de eficácia no desenvolvimento. Outra abordagem que tem-se mostrado eficiente é o MDD (Model-Driven Development), desenvolvimento de software baseado em modelos. Automatizando boa parte do desenvolvimento do software através da leitura de uma modelagem previamente elaborada. Neste contexto, a proposta desse trabalho é realizar um estudo sobre a DSM voltada para sistemas web chamada WebML/IFML, estabelecer um processo de desenvolvimento e construir uma aplicação final a partir das entradas geradas utilizando-se MDD.

Palavras-chave: Modelagem de Software. DSM. MDD. IFML.

ABSTRACT

Over the years the modeling software has grown exponentially. Increasingly companies observe the benefits of using modeling from the design of a project, avoiding problems during the implementation. Created by OMG (Object Management Group) in 1997, the UML (Unified Modeling Language) has established itself as the standard language of the market and is now known and used in at least 70% of software development organizations worldwide. It is known that software development companies are concerned to increase team productivity and product quality, and new solutions has been emerged for this purpose. Among these stands out the DSMs (Domain-specific modeling), modeling languages created for a specific domain which help to achieve high level of abstraction. Enabling a domain specific notations, unlike UML that was developed to support various sectors of engineering software and often causes loss of development effectiveness. Another approach that has proved effective is the MDD (Model-Driven Development), software development based on models. Automating much of the development of software by reading a model previously developed. In this context, the purpose of this study is to conduct a study on the DSM-oriented web systems called WebML/IFML, establish a process to develop and build a final application from the entries generated using MDD.

Key words: Software Modeling. DSM. MDD. IFML.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DSM	*Domain-specific Language*
ER	Entidade e Relacionamento
HDM	*Hypermedia Design Model*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
IFML	*Interaction Flow Modeling Language*
JSP	*JavaServer Pages*
JSTL	*JavaServer Pages Standard Tag Library*
MDA	*Model Driven Architecture*
MDD	*Model-Driven Development*
MDE	*Model Driven Engineering*
OMG	*Object Management Group*
OOHDM	*Object Oriented Hypermedia Design Method*
PDF	*Portable Document Format*
PIM	*Platform-independent Model*
PSM	*Plataform-specific Model*
RMM	*Relationship Management Model*
UML	*Unified Modeling Language*
WAE	*Web Application Extension*
WEBAPP	*Web application*
WEBE	*Web engineering*
WEBML	*Web Modelling Language*
WML	*Web Modelling Language*
XML	*eXtensible Markup Language*
XSL	*Extensible Stylesheet Language*

SUMÁRIO

INTRODUÇÃO

A modelagem é uma técnica da Engenharia Civil aprovada e bem aceita. Através de um modelo de arquitetura o cliente pode visualizar como será sua casa antes mesmo da construção, ainda, os modelos matemáticos auxiliam os engenheiros na prevenção de desastres na construção ao longo dos anos. Contudo, a modelagem não se limita apenas a indústria de construção, modelos de automóveis, sistemas de telefonia e até mesmo roteiros na indústria cinematográfica requerem algum grau de modelagem que permitam uma melhor compreensão do sistema e facilitem a comunicação da ideia à outras pessoas (BOOCH, RUMBAUCH, JACOBSON, 2005).

Neste sentido, cada vez mais ouve-se falar em modelagem de software. A grande variedade de linguagens de programação bem como as lógicas complexas tornam necessária a utilização de linguagens gráficas que facilitem o entendimento dos sistemas e orientem as decisões das equipes. (BOOCH, RUMBAUCH, JACOBSON, 2005).

Sabendo-se desta necessidade, diferentes linguagens de modelagem começaram a surgir no final dos anos 80, mas a falta de um padrão acabava fragmentando os ambientes de trabalho e gerando uma abundancia de softwares para essa finalidade na indústria (FOWLER, 2005).

Diante deste contexto viu-se a necessidade de uma linguagem aberta e que unificasse as demais que haviam surgido, então em 1997 a OMG (*Object Management Group*), um grupo formado para estabelecer padrões para sistemas orientados a objetos, criou a UML (*Unified Modeling Language*), uma linguagem de modelagem não proprietária que criou uma família de notações gráficas que atendessem as demandas do mercado (FOWLER, 2005).

Segundo Watson (2008), em 1995 as ferramentas de modelagem eram utilizadas por uma pequena fração de projetos de software; em 2006 estima-se que mais de 10 milhões de profissionais da TI utilizavam UML e em 2008 mais de 70% das organizações de desenvolvimento de software já utilizavam a linguagem. A UML tornou-se a "linguagem franca" em desenvolvimento de software, permitindo que engenheiros possam exportar seus modelos para outras organizações e serem facilmente reaproveitados.

Conforme Lobo (2009), a modelagem se torna cada vez mais importante a medida que um sistema começa a crescer e se torna mais complexo. A falta de uma modelagem desde o início pode revelar diversos erros na etapa final do projeto. Outro ponto importante apontado por Lobo é que a equipe de desenvolvimento dificilmente contará com os mesmos membros e participantes até o final do projeto, nestes casos a modelagem será ainda mais

importante para aqueles que se juntarem a equipe ao longo do projeto, uma vez que compreender diagramas visuais é muito mais fácil do que decifrar códigos criados por outros profissionais.

Além disso, sabe-se que empresas de desenvolvimento de software estão sempre preocupadas em aumentar a produtividade da equipe, desenvolvendo mais aplicações em menos tempo, mas sem abrir mão da qualidade. O trabalho de Kelly e Tolvanen (2008) observa que a troca de linguagem de programação não proporciona um ganho significativo na produtividade da equipe. Dessa forma outros meios foram encontrados para obter melhores resultados.

Dentre esses destaca-se as DSMs (*domain-specific modeling*), linguagens de modelagem criadas para um domínio especifico, o que possibilidade um nível de abstração muito alto na especificação do sistema e maior eficácia no desenvolvimento da aplicação. Isso acontece porque uma solução DSM propõe possibilidades de automação e facilita a definição do caso com notações específicas para o ambiente em questão (KELLY, TOLVANEN, 2008).

Outra abordagem é o MDD (*model-driven development*), nesta os diagramas utilizados na modelagem deixam de ser apenas referência e passam a gerar entradas para a aplicação final. Segundo Mellor, Clark e Toyo (2003), o MDD oferece o potencial de automaticamente transformar modelos projetados em alto nível em aplicações prontas para execução. Selic (2003) afirma que as tecnologias atuais de conversão de modelo para código podem gerar códigos com a mesma performance e eficiência na utilização da memória que um programador convencional, porém alguns casos críticos ainda carecem de trabalho manual.

Uma das áreas de domínio que mais tem crescido nas últimas décadas é a Engenharia Web. Aplicações críticas disponibilizadas e acessadas através da internet são cada vez mais comuns e sua performance, confiabilidade e qualidade são de primordial importância, uma vez que o nível de exigência dos usuários tem crescido na mesma velocidade. Atualmente sistemas baseados na web envolvem planejamento, arquitetura, design, teste, garantia de qualidade, avaliação de desempenho, atualização contínua e manutenção (SUH, 2004).

Aplicações web possuem características diferentes de aplicações convencionais, como a navegação entre as páginas, ambiente imprevisível, componentes específicos, etc. Visando atender estas peculiaridades a academia e a indústria oferecem soluções de modelagem focadas em expressar as principais funcionalidades de um site em alto nível.

Neste contexto em 1998 um grupo de pesquisadores de base de dados do Dipartimento di Eletronica e Informazione da Universidade Politecnico di Milano, propôs a

WebML (*Web Modeling Language*) (WebML, 2013). A WebML foi uma das linguagens de modelagem pioneiras a trabalhar com hipermídia e *webdesign*. O objetivo inicial foi apoiar na concepção e implementação de sistemas com a necessidade de armazenar e acessar grandes quantidades de dados estruturados, como comércio eletrônico, sistemas institucionais de organizações públicas e privadas, bibliotecas digitais, portais corporativos e sites de comunidades. Utilizando uma notação original para expressar os recursos de navegação e composição de interfaces, a WebML se tornou uma abordagem completamente diferente de propostas anteriores (BRAMBILLA et al., 2007).

Um diferencial competitivo da WebML é a presença de uma linha comercial de desenvolvimento trabalhando em associado com a pesquisa acadêmica, patrocinado pela Politecnico di Milano, uma spin-off chamada Web Models trabalha desde 2001 no desenvolvimento de soluções que complementem a WebML (BRAMBILLA et al., 2007).

Recentemente, surgiram iniciativas com o objetivo de tornar a WebML uma linguagem padrão de mercado reconhecida pela OMG. Porém, para que isso fosse possível, surgiu uma nova linguagem chamada IFML (*Interaction Flow Modeling Language*) que foi desenvolvida com o apoio da OMG e em março de 2013 adotada como padrão (WEBRATIO 2013a) (IFML, 2013).

A IFML é uma linguagem totalmente inspirada nos conhecimentos adquiridos pela WebML, porém a linguagem sofreu algumas alterações como nome de elementos e representações visuais de notações que compõe o modelo (WEBRATIO, 2013b).

A ferramenta WebRatio, concebida em 2006, propõem um ambiente de desenvolvimento, suportando a modelagem com IFML e a capacidade de exportar código para aplicações em Java/JSP 2.0[1] utilizando MDD. Hoje a WebRatio está consolidada na indústria, utilizada em mais de 100 aplicações e superando quatro mil downloads por ano, além de ser utilizada em várias universidades e instituições para cursos de Engenharia Web. (BRAMBILLA et al., 2007). Ainda, através da ferramenta é possível aumentar a produtividade em até três vezes, conforme a fabricante de produtos de TI Acer (WEBRATIO, 2013a).

---

[1] A tecnologia JavaServer Pages (JSP) permite facilmente criar conteúdo web utilizando componentes tanto estáticos quanto dinâmicos (ORACLE, 2010).

A proposta desse trabalho é realizar um estudo sobre a linguagem de modelagem WebML/IFML, estabelecer um processo de desenvolvimento e modelar uma aplicação web utilizando as notações da linguagem e, construir a aplicação final a partir das entradas geradas utilizando-se MDD.

OBJETIVOS

Diante do que já foi exposto o objetivo geral desse trabalho é estudar os conceitos oriundos das técnicas de MDD associando a linguagem de modelagem visual WebML/IFML. Ao final, estabelecer um processo e desenvolver uma aplicação web utilizando como base códigos gerados a partir do processo estabelecido, investigando os pontos positivos e negativos do processo, bem como a produtividade do mesmo.

Dentre os objetivos específicos destaca-se:

- Apresentar conceitos relacionados ao MDD;
- Estudar a linguagem de modelagem visual WebML/IFML;
- Definir um processo de desenvolvimento aderente ao MDD e a WebML/IFML;
- Designar uma ferramenta de modelagem que ofereça as notações da WebML/IFML;
- Modelar um sistema web real utilizando a linguagem WebML/IFML;
- Desenvolver e avaliar a aplicação construída conforme as premissas do MDD.

ESTRUTURA DO TRABALHO

Este trabalho está dividido em seis capítulos. No primeiro capítulo são apresentados os conceitos importantes da engenharia de software, bem como uma visão geral a respeito de processos de desenvolvimento que privilegiam modelos. No segundo capítulo é feito um estudo sobre a engenharia web, analisando atributos e características das aplicações e uma proposta de framework que auxilie na construção dos mesmos. No capítulo seguinte, são apresentados características a respeito de linguagens de modelagem e uma comparação entre linguagens específicas para aplicações web. No quarto capítulo é apresentada a linguagem de modelagem escolhida para o experimento proposto neste trabalho. Já o quinto capítulo contempla todo o processo de desenvolvimento detalhadamente. No último capítulo é

realizada uma avaliação acerca da estratégia utilizada, além de uma comparação de esforço ao utilizar-se MDD.

1 ENGENHARIA DE SOFTWARE

Embora existam semelhanças entre o desenvolvimento de software e a fabricação de *hardware*, as duas atividades são fundamentalmente diferentes. Apesar de ambas as atividades objetivarem a construção de um "produto" e dependerem de um bom projeto para obter alta qualidade, a relação entre pessoas envolvidas e trabalho realizado é completamente diferente (PRESSMAN, 2011).

O *hardware* apresenta taxas de defeitos altas no início de sua vida, porém os defeitos são corrigidos e a taxa cai para um nível estável. A medida que o tempo passa e os componentes de *hardware* sofrem os efeitos cumulativos de poeira, impactos, temperaturas e outros males ambientais e a taxa de defeitos volta a subir até que o *hardware* fique inutilizável (PRESSMAN, 2011).

O software, não sofre com os males ambientais, entretanto, durante sua vida ele passará por alterações e consequentemente novos erros serão introduzidos. A medida que isso acontece a taxa de defeitos se acentua e antes mesmo que o produto chegue a uma linha estável de erros, novas alterações serão solicitadas e novamente e a taxa aumentará. Ao passar dos anos o software não irá desgastar, mas acabará se deteriorando devido a modificações (PRESSMAN, 2011).

No sentido de trabalhar a dificuldades de um projeto de software surge a disciplina de Engenharia de Software que estuda processos, métodos e práticas que formam um leque de ferramentas e oportunidades de trabalho para viabilizar um ambiente de boas práticas e possibilitar aos profissionais desenvolverem softwares de qualidade (PRESSMAN, 2011).

Neste contexto, uma etapa importante para a garantia de qualidade do software é o cuidado na modelagem, segundo Lobo (2009), a importância de construir modelos ideais está relacionada à grande necessidade de se construir softwares de qualidade. Dessa forma, ciente de que a Engenharia de Software propicia diferentes processos de desenvolvimento, alguns desses evidenciam e privilegiam modelos, tais processos são conhecidos como: MDE, MDA e MDD, do inglês *model-driven engineering*, *model-driven architecture, model-driven development*, a seguir os subcapítulos 2.3, 2.4 e 2.5 dessa monografia descrevem tais processos. Entretanto, para um melhor embasamento é necessário definir formalmente o que é modelo e meta-modelo.

1.1 MODELO

Entende-se por modelo, uma abstração arquitetada com uma série de elementos formais que descrevem algo a ser construído com algum objetivo previamente definido (MELLOR, CLARK, TOYO, 2003).

A utilização de modelos torna possível realizar uma série de análises, tanto em relação ao problema que está sendo modelado, bem como no produto que está sendo desenvolvido/concebido. Dessa forma, a partir do uso adequado de modelos pode-se obter, algumas características importantes para o processo de desenvolvimento, tais como (MELLOR, CLARK, TOYO, 2003):

- Melhora a comunicação de ideias entre pessoas/máquinas envolvidas no processo;
- Permite verificação de integridade do produto ainda em tempo de projeto;
- Admite a concepção e geração de casos de teste;
- Viabiliza indicadores ligados a custo, esforço de desenvolvimento e qualidade;
- Permite a criação/utilização de padrões de projeto.

No contexto da Engenharia de Software, entende-se por modelo como uma forma reduzida de representar um sistema. Removendo ou escondendo detalhes que são irrelevantes para um determinado ponto de vista, isto permite compreender a essência, do objeto modelado, mais facilmente (SELIC, 2003).

Dentre as vantagens da utilização de modelos, destaca-se o alto nível de abstração, tornando-o inteligível para um usuário final, ressaltando uma das características apontadas anteriormente - Melhora a comunicação de ideias entre pessoas/máquinas envolvidas no processo (MELLOR, CLARK, TOYO, 2003).

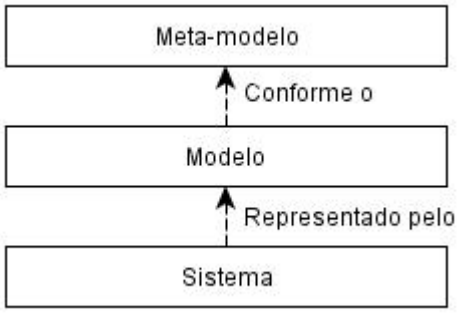
1.2 META-MODELO

Meta-modelos (do grego, meta que significa “depois”) são modelos construídos para descrever a semântica de um processo guiado por modelos (MELLOR, CLARK, TOYO, 2003). Pode-se dizer que é um objeto que define o próprio modelo que será empregado em alguma etapa no processo de desenvolvimento de software.

Segundo a OMG (2013), um meta-modelo é o modelo de um modelo, sendo assim, este conceito pode ser recursivamente aplicado, onde cada novo meta-modelo gerado, pode

ser especificado por algum outro. A Engenharia de Software, já contempla uma série de processos de desenvolvimento que privilegiam modelos de software, intui-se com isso, conceber metodologias que transformem modelos em sistemas computacionais, conforme a Figura 1.1.

Figura 1.1 - Os dois relacionamentos básicos de um meta-modelo



Fonte: Adaptado de http://www.ie.inf.uc3m.es/ggenova/Warsaw/Part3.pdf

Assim, para fazer da automação uma realidade, modelos devem ter significados definidos, utilizando uma linguagem com sintaxe e semântica bem definidos (MELLOR, CLARK, TOYO, 2003). Dessa forma, paralelemente a Engenharia de Software, a disciplina de Engenharia Baseada em Modelo, estuda as melhores práticas no contexto dos processos de desenvolvimento guiado por modelos.

1.3 MDE (ENGENHARIA BASEADA EM MODELO)

De modo geral, como dito anteriormente, modelos são representações simplificadas da realidade. Na engenharia, modelos são utilizados para definir de forma rápida, simplificada e objetiva características de produtos que serão posteriormente implementados (LOBO, 2009).

Porém, a simples criação de um modelo não é garantia de que o software será construído com rapidez e qualidade, para isso é necessário o entendimento da equipe quanto aos seguintes quesitos (LOBO, 2009):

- Método/Processo adotado no ambiente de desenvolvimento de software;
- Controle de qualidade de software;
- Utilização de arquitetura de software adequada ao projeto;
- Controle nos processos;
- Fidelidade aos padrões.

Os modelos são criados para documentar e, assim, facilitar as fases de desenvolvimento de software, pois possibilitam uma visão global e rápida do sistema e,

permitem que toda a equipe se oriente através dos modelos. Entretanto, um modelo de software mal definido ou inadequado pode causar um caos nos resultados do projeto (LOBO, 2009).

Para melhor entender-se o conceito e a empregabilidade do MDE, é importante entender o contexto histórico que antecede o surgimento da mesma, abordado no subcapítulo 2.3.1, a seguir define-se MDE.

1.3.1 Contexto histórico da MDE

Nas últimas cinco décadas, pesquisadores de software e desenvolvedores tem criado abstrações para que seja possível focar no que é importante para o desenvolvimento do software, deixando de lado o que não é relevante, como por exemplo, rotinas que controlam CPU, memória e dispositivos de rede. Buscando sempre criar um escudo para estas complexidades de ambiente (SCHIMIDT, 2006).

Neste contexto, linguagens de alto nível surgiram para preservar o desenvolvedor da linguagem de máquina, já sistemas operacionais como OS/360 e Unix, para proteger os desenvolvedores do controle direto do hardware (SCHIMIDT, 2006).

Ainda, surgiram linguagens orientadas a objeto como C++, Java ou C#. Todas elas aumentaram ainda mais o nível de abstração, o que possibilitou aos desenvolvedores focarem no que realmente importa para o domínio de negócio do sistema. Essas linguagens ainda proporcionaram classes e bibliotecas reutilizáveis (SCHIMIDT, 2006).

Apesar das evoluções, muitos problemas ainda persistem, por exemplo, garantir as funcionalidades de um software em novas/diferentes plataformas, o que requer um tempo considerável para que o código seja migrado de plataforma, muitas vezes manualmente, além de atualizações constantes direcionadas a cada uma das plataformas (SCHIMIDT, 2006).

Este e outros problemas acabam forçando os desenvolvedores a implementar soluções inadequadas que duplicam códigos, violam princípios da arquitetura, além de complicar a evolução do sistema e a garantia da qualidade (SCHIMIDT, 2006).

Diante deste contexto histórico, viu-se a oportunidade, e a necessidade, de se criar um elemento de abstração que permita a modelagem de um sistema, bem como a transformação dessa modelagem em sistemas computacionais independente de plataforma.

1.3.2 Conceito de MDE

Segundo Lobo (2009), a Engenharia Dirigida por Modelos propõe uma metodologia que visa contornar os problemas relacionados ao desenvolvimento de um sistema computacional multiplataforma, bem como estimular a qualidade e, reutilização de elementos que representam conceitos ligados diretamente com o domínio de um problema. Dessa forma, a MDE, emprega os seguintes conceitos:

- **Linguagens de modelagem de domínio específico:** As quais formalizam a estrutura da aplicação, o comportamento, e requisitos de um domínio em particular. São descritos utilizando-se meta-modelos que especificam a relação entre os conceitos do domínio em questão e as restrições associadas a estes conceitos (LOBO, 2009);
- **Máquinas de transformação e geradores:** Analisam os modelos e então geram artefatos, como código do aplicativo, descritores XML[2] (*eXtensible Markup Language*) ou outros modelos alternativos. O processo de conceber artefatos através dos modelos assegura coerência entre a especificação e a implementação da aplicação. O processo de transformação automatizado é frequentemente referenciado como "correto-por-construção", oposto do convencional "construído-por-correção" (LOBO, 2009).

Assim, entende-se por MDE como uma disciplina que estuda diferentes maneiras de aplicar as Linguagens de Domínio Específico e, transformar os modelos gerador por essas em novos artefatos, sejam sistemas ou outro modelos.

1.4 MDA (ARQUITETURA DIRIGIDA PELOS MODELOS)

A MDA é uma iniciativa da OMG (2013) que busca aumentar a produtividade e reuso através da separação em duas estruturas lógicas, onde a primeira se ocupa do domínio do problema e, a segunda da lógica que envolve plataforma a qual a aplicação estará sendo executada (SPARX SYSTEMS, 2007).

A MDA concentra-se inicialmente na funcionalidade e comportamento do aplicativo (domínio do problema), sem preocupar-se com a tecnologia da plataforma que o aplicativo irá

[2] XML é uma linguagem de marcação que permite armazenar e transportar dados utilizando-se uma organização extensível (W3C, 2013a).

rodar futuramente. Desta forma, não é necessário repetir o processo de definição de uma aplicação ou funcionalidades do sistema, este será gerado uma única vez (OMG, 2013).

A especificação MDA consiste em um modelo base independente de plataforma denominado PIM – *Platform Independent Model*. Associado a este encontra-se diferentes modelos específicos de cada plataforma, intitulados de PSM – *Platform Specific Model* (OMG, 2013). Sendo assim:

- **PIM (modelo independente de plataforma):** Descrevem a estrutura e funções do sistema, mas não especifica a implementação do mesmo. Contém informações suficientes para conduzir um ou mais modelos de plataforma específica (SPARX SYSTEMS, 2007);
- **PSM (modelo de plataforma específica):** Código fonte, arquivos de configurações, XML e outras saídas específicas de uma plataforma (SPARX SYSTEMS, 2007).

A capacidade do MDA em transformar um PIM e vários PSM simplifica a implementação do sistema em diferentes plataformas, uma vez que o desenvolvedor deve preocupar-se em modelar as funções de negócio uma única vez utilizando-se de um modelo PIM e posteriormente utilizar esse mesmo modelo para gerar código para diferentes linguagens, gerando artefatos por exemplo para Java e C#.

Para uma melhor compreensão, a Figura 1.2 exemplifica a relação de multiplicidade entre os elementos PIM e PSM.

Figura 1.2 - Diagrama de transformação MDA

Fonte: Adaptado de (SPARX SYSTEMS, 2007)

1.5 MDD (DESENVOLVIMENTO DIRIGIDO POR MODELOS)

O MDD é um processo de desenvolvimento de software que privilegia o modelo, neste primeiro cria-se um modelo que representa as funcionalidades de um sistema. Em

seguida, este modelo é traduzido em uma linguagem compilável (ATKINSON, KUHNE, 2003).

Neste contexto, pode-se definir que todo desenvolvimento é dirigido por modelos, uma vez que uma aplicação desenvolvida em linguagens convencionais (linguagens de alto nível), como Java, por exemplo, deve ser primeiro transformada em linguagem de máquina, para então ser executada pelo computador. Em geral, essa transformação é feita por compiladores ou máquinas virtuais que interpretam os modelos de uma linguagem em específico (MELLOR, CLARK, TOYO, 2003).

Assim, entende-se, linguagens de alto nível, como modelos – abstrações da linguagem de máquina. Entretanto, normalmente emprega-se o termo de MDD, para desenvolvimento guiado por modelos gráficos visuais.

A automação, oriunda da transformação de um modelo em linguagem compilável, não anula a necessidade da criatividade. Apenas concentra a criatividade no objetivo, ou seja, no domínio do problema (MELLOR, CLARK, TOYO, 2003).

Além disso, o MDD pode automatizar a transformação de um modelo para uma linguagem compilável, conforme dito, ou então para um novo modelo de destino. Cada um dos modelos, origem e destino, são expressos através de elementos que representam os diferentes comportamentos de um sistema. Exemplo, o modelo destino pode definir os dados para acesso remoto a algum objeto, mesmo que isso não esteja no escopo do modelo de origem.

Uma das vantagens da utilização do MDD é a padronização dos elementos construídos, uma vez que os modelos são criados para determinado contexto e utilizam-se de ferramentas que auxiliam no processo da utilização de um modelo. Com isso, intui-se a economia de tempo, significa muito ganho de produtividade, além da qualidade dos códigos gerados, uma vez que estes são produzidos pelas máquinas de transformação (WALLS, RICHARDS, 2004 apud SILVEIRA, 2012, p. 23).

Para entender-se melhor o processo de transformação utilizado pelo MDD, é importante compreender os conceitos de funções de mapeamento, modelo e meta-modelo.

1.5.1 Funções de mapeamento

A transformação entre dois modelos escritos em diferentes linguagens de modelagem é possível através das funções de mapeamento. Essas funções estão definidas nos meta-modelos das linguagens e fornecem informações a respeito do modelo elaborado (MELLOR, CLARK, TOYO, 2003).

Os meta-modelos podem ser expressos utilizando-se o *OMG Meta-Object Facility*, que prevê um padrão para criação das funções que permitem consultar, visualizar e transformar modelos (MELLOR, CLARK, TOYO, 2003).

Na tentativa de evitar que as funções de mapeamento fiquem associadas a apenas um meta-modelo, existem iniciativas de criação de meta-modelos "*SILOS*" que vinculam diferentes domínios (MELLOR, CLARK, TOYO, 2003). Essa padronização fornece um significativo impulso, porque coleta as melhores práticas, incentiva a reutilização e facilita a interconexão entre ferramentas complementares utilizadas no processo de MDD. (SELIC, 2003).

2 ENGENHARIA WEB

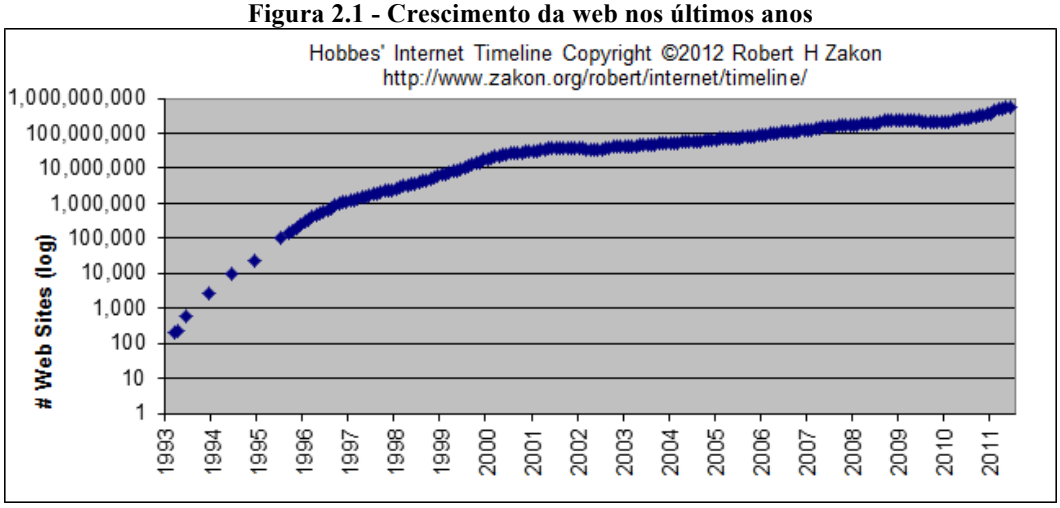
A Web tem se tornado uma tecnologia indispensável para os negócios, comércio, comunicação, educação, engenharia, entretenimento, finanças, governo, indústria, mídia, medicina, política, ciências, e transporte. Para nomear apenas algumas áreas que impactam sua vida. Isso tem mudado a maneira que compramos produtos (e-commerce), conhecemos pessoas (encontros online), entendemos o mundo (portais), buscamos nossas notícias (web mídia), expressamos opiniões (blogs), se distrai (tudo, desde música online até cassinos online), e vai para a escola (educação a distância) (PRESSMAN, LOWE, 2009) (tradução nossa).

Para que as funcionalidades citadas acima fiquem disponíveis para o usuário, é necessário a utilização de uma aplicação web (WebApp). Esta aplicação será a responsável por receber a informação requisitada pelo usuário final, estruturá-la e posteriormente enviar de volta para o usuário em forma de um pacote de apresentação, ou seja, a aplicação web deve ficar disponível para receber requisições e responde-las com as informações solicitadas (PRESSMAN, LOWE, 2009).

Seguindo o contexto acima, uma aplicação web irá retornar um pacote de apresentação, porém esse pacote não encontra-se em uma forma legível para um usuário comum que está utilizando o sistema, ao contrário disso, esse pacote de apresentação reúne instruções e parâmetros de como o sistema deve ser apresentado na tela, de forma que torne a utilização do sistema uma experiência simples e agradável. Para que essas instruções e parâmetros se transformem na tela final, é necessário a utilização de um navegador, no inglês também conhecido como *browser*, o qual tem a função de receber códigos em texto da aplicação, interpretar de forma que identifique a disposição dos elementos na tela e posteriormente apresentar para o usuário a tela desenhada (PRESSMAN, LOWE, 2009).

Além das funcionalidades citadas anteriormente, existe uma forte tendência de cada vez mais funcionalidades de variados setores do mercado estarem disponíveis através da internet. Pode-se evidenciar este fato observando o número de servidores web, que hospedam uma ou mais aplicações, que estão operando pelo mundo todo. Enquanto 18 anos atrás, no inicio da internet, existiam cerca 100 servidores no total, ao final de 2011 este número já ultrapassava 500 milhões, conforme demonstrado pelo gráfico na Figura 2.1.

Figura 2.1 - Crescimento da web nos últimos anos



Fonte: http://www.zakon.org/robert/internet/timeline/

Sabendo-se que aplicações web necessitam necessariamente de servidores e são estimulado a partir de requisições feitas por usuários, conclui-se que estas diferem das aplicações criadas para ambientes locais em vários aspectos, para entender melhor estas diferenças e particularidades o próximo capítulo irá abordar o assunto Aplicações Web.

2.1 APLICAÇÕES WEB

Na inicio da *World Wide Web*, um *website* não passava de uma série de arquivos de hipertextos vinculados entre si, composto por textos e gráficos simples.

Com o passar dos anos, houve uma grande evolução nas tecnologias, tanto no lado do cliente (Javascript[3], Css[4]), como no lado do servidor (linguagens robustas e dinâmicas). Desta forma, *websites* deixaram de serem apenas simples páginas para tornarem-se software de computadores tão poderosos quanto os convencionais (que rodam em ambiente local), integrando-se com outros sistemas e até mesmo base de dados do governo.

[3] JavaScript é um código de programação que pode ser inserido em páginas de HTML (W3C, 2013c).

[4] CSS (*Cascading Style Sheets*) são folhas de estilo que definem como elementos HTML devem ser exibidos no aspecto visual (W3C, 2013b).

Ainda neste contexto, uma aplicação web pode, por muitas vezes, vir a substituir as aplicações existentes previamente, que foram concebidas para rodar em ambiente local. Porém, ao desenvolver uma aplicação web, que também pode ser chamada de WebApp, existem uma série de atributos que devem ser observados e relevados para garantir a qualidade e viabilidade da aplicação. Conforme (PRESSMAN, LOWE, 2009), pode-se destacar os seguintes atributos:

- **Intensidade de rede:** Toda WebApp é acessada pelos usuários através de uma rede;
- **Simultaneidade:** Muitos usuários podem acessar a aplicação ao mesmo tempo, sendo que muitas vezes a ação de um usuário pode causar impacto na do outro;
- **Carga imprevisível:** O número de usuários varia a cada dia. Uma WebApp deve estar preparada para atender vários eventos simultaneamente;
- **Performance:** Uma WebApp deve responder as requisições rapidamente, caso contrário o cliente pode desistir de esperar e abandonar a aplicação;
- **Disponibilidade:** A expectativa é que a aplicação fique disponível 100% do tempo, existem usuários acessando o sistema 24 horas por dia, 7 dias por semana;
- **Orientado a dados:** WebApps são utilizadas para disponibilizar textos, gráficos, áudios, vídeos e também informações compartilhadas com outros sistemas internos (por exemplo o sistema financeiro);
- **Conteúdo sensível:** A qualidade do conteúdo é determinante para o sucesso da aplicação;
- **Evolução contínua:** WebApps requerem continuidade, algumas aplicações chegam a ser atualizadas diariamente, sem data marcada;
- **Imediatismo:** Existe uma necessidade de disponibilizar produtos para o mercado rapidamente. Sendo assim, as WebApps devem ser planejadas e desenvolvidos em poucos dias;
- **Segurança:** WebApps são acessadas por pessoas de todo o mundo. Então uma forte segurança deve ser pensada para proteger as informações internas;

- **Estética:** Uma boa interface e design agradável são imprescindíveis para o sucesso de uma WebApp. Em uma aplicação de vendas o design é tão importante para o sucesso quanto a funcionalidade.

Além de dar a devida importância para os atributos de uma aplicação web, é importante identificar a categoria da aplicação. Essa importância se da pelo fato de que cada categoria combina o seu foco em informação, funcionalidade e comportamento de uma maneira diferente e, desta forma, um engenheiro Web irá focar em diferentes aspectos como interface, conteúdo, navegação e funcionalidade. Por exemplo, em uma aplicação de funcionalidades complexas, tal como um site de leilão, é mais provável a necessidade da modelagem funcional completa. Enquanto uma aplicação que envolve informações, como um site de notícias, provavelmente vai exigir um foco na modelagem de conteúdo e modelos de iteração que descrevem como os usuários irão interagir com o conteúdo (PRESSMAN, LOWE, 2009).

Segundo Pressman e Lowe (2009), as aplicações habituais devem evoluir através das seguintes categorias:

WebApps de informação

Esse tipo de aplicações objetivam a exibição de informações acerca de serviços e produtos. É utilizada uma navegação simples para o usuário e o conteúdo é restrito para leitura, sem que o usuário possa interagir ou customizar as informações recebidas.

WebApps de download

Esse tipo de aplicação é voltada a disponibilização de arquivos para os usuários. Pode ser utilizada, por exemplo, para disponibilizar arquivos do tipo PDF (*Portable Document Format*) com especificações descrevendo algum produto ou serviço.

WebApps customizáveis

Aplicações customizáveis possibilitam a existência de diferentes versões para uma mesma página. A customização é feita de acordo com o perfil do usuário, ou seja, uma mesma aplicação pode existir uma versão voltada ao público doméstico, outra versão voltada ao público comercial e uma terceira criada especialmente para usuários técnicos. Sendo que a exibição será direcionada a partir do momento que o perfil do usuário for identificado.

WebApps de interação

Esse tipo de aplicação é criada de forma que privilegie a colaboração dos usuários para a construção do conteúdo da página. É frequentemente utilizada em produtos ou serviços que possuem uma comunidade em torno de sua existência e surge a necessidade de possibilitar o envolvimento desta comunidade através de chat, perguntas e respostas, depoimentos, avaliações, entre outros.

WebApps de parâmetros do usuário

Aplicações de parâmetros do usuário são compostas por formulários que possibilitam que o usuário preencha e envie informações para a aplicação. É utilizada, por exemplo, em páginas que possuem um formulário de contato onde o usuário pode facilmente enviar sua mensagem em vez de ter que redigir um e-mail através de uma aplicação externa.

WebApps orientadas a transação

Aplicações orientadas a transações são carregadas mediante informações que são recebidas através de interações do usuário. Por exemplo, uma aplicação que recebe as entradas do usuário providas através de um formulário de contato, analisa as informações e pode executar alguma ação, como uma resposta automática, mediante as entradas que foram submetidas.

WebApps orientadas a serviço

Esse tipo de aplicação é voltada a disponibilização de informações de acordo com as entradas do usuário. Pode ser utilizado, por exemplo, para exibir informações a respeito da região do usuário, uma vez que a entrada foi a descrição do espaço geográfico em que ele se encontra. Para isso um serviço é utilizado para obter diretamente as informações desejadas.

WebApps de portais

Aplicações do tipo portal, são desenvolvidas para prover conteúdo completo e informações relevantes a respeito de um ou mais assuntos. A medida que a página de um serviço ou produto vai crescendo, surge a necessidade de prover mais informações a respeito de um domínio de negócio, uma vez que mais usuários passam a utilizar e as dúvidas passam a ser recorrentes.

WebApps voltada a acesso a base de dados

Este tipo de aplicação é necessária quando surge a necessidade de consultar dados externos às informações da página. Uma mesma aplicação pode consultar várias base de dados ao mesmo tempo, podendo inclusive incrementá-las com dados submetidos pelos

usuários. Um exemplo de acesso a base de dados externa seria a consulta sobre a relação de clientes e produtos.

WebApps de *warehouse*

Aplicações deste tipo são as que extraem informações úteis ao negócio de diferentes fontes, armazenam em uma base de dados e disponibilizam para os clientes. Geralmente as bases de dados de uma *warehouse* possuem uma larga escala e podem ser utilizadas, por exemplo, para armazenar informações a respeito de fornecedores, normas, etc.

2.2 FRAMEWORK WEBE

Sabendo-se que WebApps possuem atributos específicos e categorias próprias, a utilização da engenharia de software convencional pode não ser a mais adequada ao se desenvolver uma aplicação com características de web, neste sentido uma alternativa foi criada buscando reduz os riscos e aumenta a probabilidade de sucesso da aplicação. A alternativa é a Engenharia Web (WebE), a qual propõem um framework ágil para auxiliar na construção de WebApps de qualidade (PRESSMAN, LOWE, 2009).

Um framework estabelece uma base para o processo completo da engenharia Web. Reunindo uma série de atividades replicáveis para qualquer projeto WebApp, independentemente do tamanho ou complexidade. Além de fornecer uma série de *umbrella activities* (gerenciamento de risco, garantia da qualidade e gerenciamento de conteúdo) que são aplicáveis durante todo o processo da WebE (PRESSMAN, LOWE, 2009).

A característica de ser ágil, é importante uma vez que as demandas, estratégias e regras mudam rapidamente, o gerenciamento das demandas exigem respostas quase instantâneas e os clientes mudam de ideia a cada entrega. Sabendo de todas essas premissas, as equipes de WebE devem enfatizar *agility*, ou seja, devem ser rápidos em suas ações e decisões (PRESSMAN, LOWE, 2009).

Segundo Pressman e Lowe (2009) um framework genérico auxilia nas seguintes atividades:

Comunicação: Envolve uma forte interação e colaboração com o cliente e abrange o levantamento de requisitos e outras atividades relacionadas.

Planejamento: Engloba as ações, tarefas técnicas, riscos, recursos que serão necessários, produto de trabalho, e estabelece um cronograma.

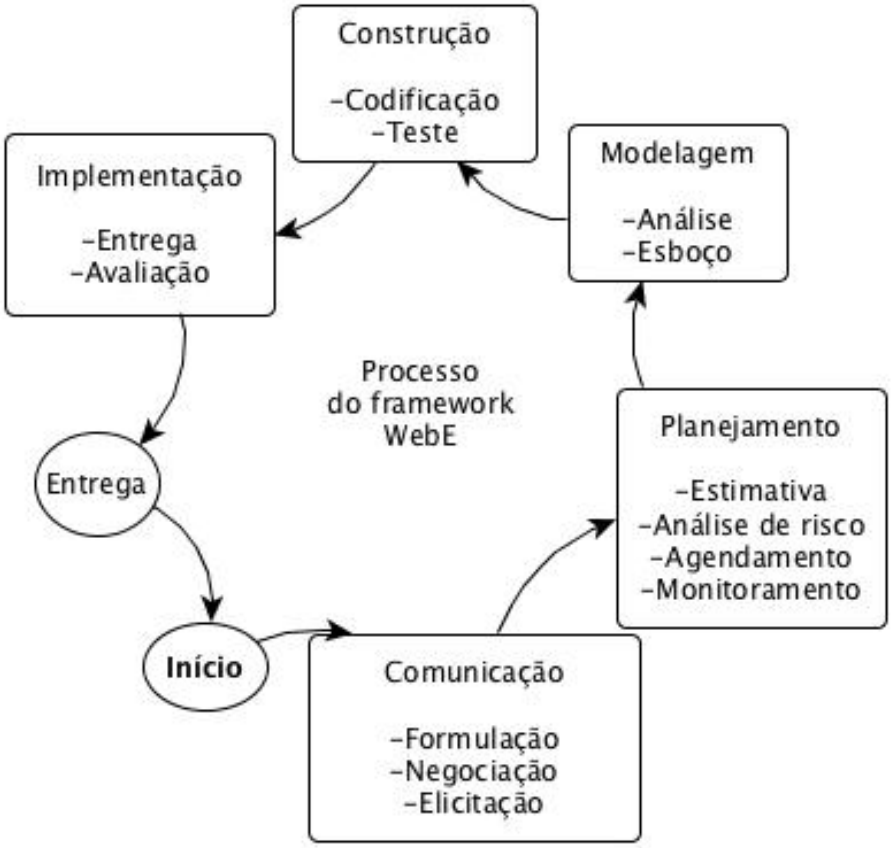
Modelagem: Engloba a criação de modelos que irão auxiliar os desenvolvedores e ajudar o cliente a entender os requisitos.

Construção: Combina geração de HTML (*HyperText Markup Language*), XML, Java e outros códigos com testes necessários para identificar erros no código.

Implementação: Entrega as novas funcionalidades para o cliente, que avalia e fornece um feedback.

Para entender o fluxo de um framework WebE, a 2.2 ilustra o processo sinalizando a ordem de cada atividade, além disso cita algumas tarefas habituais que compõem cada atividade.

Figura 2.2 - Processo do framework WebE



Fonte: Adaptado de (PRESSMAN, LOWE, 2009)

3 LINGUAGENS DE MODELAGEM

Entende-se por uma linguagem de modelagem, uma série de notações, termos e símbolos que tem por objetivo representar graficamente um conjunto de características de um sistema e o relacionamento entre os componentes do mesmo, de forma rápida, clara e objetiva (LOBO, 2009).

Estas notações utilizadas por uma linguagem de modelagem, podem representar elementos genéricos que irão abranger variados domínios de negócio e/ou plataformas de desenvolvimento, como também podem focar exclusivamente em um domínio e, desta forma, aumentar a qualidade do modelo ao contar com elementos dedicados ao domínio do problema. Nas sessões a baixo serão abordados mais detalhes sobre as duas possibilidades de linguagens, generalistas ou de domínio específico.

3.1 LINGUAGENS DE MODELAGEM GENERALISTAS

Conforme Pressman e Lowe (2009), uma linguagem de modelagem generalista é concebida de forma que as suas notações possam abranger um maior número de situações e domínios, sendo assim, uma linguagem generalista é capaz de produzir diferentes modelos para diferentes domínios e situações. Algumas características que contemplam estas linguagens são levantadas por Kelly e Tolvanen (2008):

- Criadas para auxiliar no desenvolvimento posterior;
- Compatíveis com diferentes domínios;
- Permitem especificar, visualizar e documentar o produto final.

Entre as linguagens generalistas existentes no mercado, a UML (*Unified Modeling Language*), criada pela OMG em 1997, consolidou-se como sendo a linguagem padrão de mercado (FOWLER, 2005). Conforme Lobo (2009), esta linguagem permite flexibilidade e fácil customização, sendo, desta forma, capaz de criar modelos abstratos para qualquer software, independente do domínio do problema.

Ainda neste contexto, Kelly e Tolvanen (2008) reforçam a importância da linguagem UML para a indústria do software, uma vez que esta enfatizou a importância da modelagem para o processo de desenvolvimento de um sistema. Assim como Watson (2008) afirma que antes do surgimento da UML uma pequena fração dos projetos de software utilizavam ferramentas de modelagem e hoje mais de 70% da indústria utiliza.

Porém, a utilização da UML auxilia durante a especificação, visualização e documentação, mas não oferece uma grande ajuda na automação do desenvolvimento de um software e, por muitas vezes, não gera um aumento de produtividade (KELLY, TOLVANEN, 2008). Para auxiliar neste contexto, surgem as linguagens de modelagem dedicadas a um domínio específico, chamadas DSM (*domain-specific modeling*).

3.2 DSM (LINGUAGENS DE MODELAGEM PARA DOMÍNIO ESPECÍFICO)

Conforme Pressman e Lowe (2009), uma linguagem de modelagem para domínio específico é concebida de forma que suas notações sejam especializadas para funções de um domínio em específico, desta forma, há uma tendência que para cada novo domínio surja uma nova linguagem especializada, diferentemente das linguagens generalistas estudadas anteriormente, que procuram abranger o maior número de domínios possível. Algumas características que contemplam estas linguagens são levantadas por Kelly e Tolvanen (2008):

- Elevam o nível de abstração (em um nível acima do que linguagens generalistas);
- Otimizadas para automação (conversão de modelo para código);
- Foca em um domínio menor.

Entre as vantagens da utilização de uma DSM, destaca-se o aumento da produtividade da equipe e, para isso, três estratégias são apontadas por Kelly e Tolvanen (2008) devem ser utilizadas:

- Trabalhar em um alto nível de abstração, de forma que os analistas possam criar sistemas mesmo sem o conhecimento técnico da linguagem final, assim o foco pode ser exclusivamente nos protótipos e não na implementação;
- Vincular o protótipo com a implementação, garantindo a fidelidade das funcionalidades modeladas com as funcionalidades do sistema final;
- Criar ferramentas altamente eficientes, de forma que o desenvolvedor perceba ganhos na sua utilização e utilize-a sempre.

Como visto anteriormente, sabe-se que existem linguagens de modelagem para os mais diversos domínios. Entretanto, é importante ressaltar o domínio que está sendo

abordado nessa monografia, que é o desenvolvimento de aplicações web. Sendo assim, a seguir será abordado o estudo de WMLs (*Web Modelling Languages*).

3.2.1 Linguagens de modelagem web

Visando aprimorar o processo de desenvolvimento de aplicações web, muitas linguagens para este domínio tem surgido nos últimos anos. Estas linguagens procuram auxiliar durante o desenvolvimento, manutenção e evolução das aplicações, para isso elas devem fornecer funções sofisticadas que auxiliem na compreensão, documentação, visualização, comunicação e construção de arquitetura importante, design detalhado e decisões de implementação (GU, HENDERSON-SELLERS, LOWE, 2002).

A fim de comparar as principais linguagens de modelagem web, Gu, Henderson-Sellers e Lowe (2002) reúnem uma série de requisitos que podem ser identificados para medir a capacidade de linguagem e, assim, medir sua eficiência frente as outras. Estes requisitos serão descritos abaixo e dividem-se em três categorias, requisitos de arquitetura funcional, requisitos de arquitetura de informação e requisitos gerais do domínio:

3.2.1.1 Requisitos de arquitetura funcional

As características funcionais de um sistema web impõem uma série de exigências sobre as linguagens de modelagem, de modo que esta consiga representar as funcionalidades de modo eficaz. Nos tópicos abaixo serão apresentados alguns dos requisitos desta arquitetura.

3.2.1.1.1 Capacidade de modelar integração e conectividade

É crucial para as linguagens de modelagem serem capazes de representar a integração de sistemas e recursos de alto níveis de arquitetura, os tópicos abaixo citam algumas situações onde a representação da integração e conectividade se fazem importantes:

- Em muitas organizações o sistema Web recém desenvolvido precisa trabalhar em conjunto com sistemas de negócio já existentes na organização e geralmente desenvolvidos em linguagens diversas. O desafio da integração dos sistemas web com estes sistemas legados de forma contínua requerem suporte da linguagem de modelagem;
- A integração de componentes é grande parte dependente das especificações de interface, as WMLs precisam fornecer capacidade de modelar e documentar componentes de interface de forma precisa;

- É desejável que as WMLs demonstrem a conexão e navegabilidade a partir das definições de interface;
- Sistemas web precisam se conectar a uma variedade de fontes de informação e serviços, tanto dentro como fora dos limites organizacionais. A representação dos mecanismos de conectividade e acesso devem ser apoiadas pela linguagem de modelagem.

3.2.1.1.2 Capacidade de suportar um padrão de modelagem

Pesquisas e experiências práticas indicam fortemente a utilização de padrões, sempre que possível, em sistemas de software orientados a objeto. Como formatos para captura, armazenamento, distribuição e reutilização de experiências e melhores práticas, os benefícios potenciais de padrões incluem:

- Prestação de orientação a desenvolvimento de software, assim as decisões corretas de projeto são feitas com o mínimo de esforço. Isto pode ajudar a atender prazos apertados de projetos de desenvolvimento;
- Como os padrões apropriados são comprovados e comumente aceitos, eles costumam atender um alto padrão de qualidade. O uso de padrões durante o desenvolvimento de software pode ajudar a aumentar a qualidade total do sistema web;
- Uma vez que os padrões são normalmente compartilhados entre desenvolvedores de software, especialmente do domínios de aplicações semelhantes, o uso de padrões pode realmente aumentar a interconexão dos sistemas web.

3.2.1.1.3 Capacidade de representar conceitos de forma tecnologicamente neutra

Uma vez que as tecnologias envolvidas nos sistemas web mudam rapidamente, é desaconselhável projetar uma arquitetura de sistema de uma maneira específica para uma tecnologia, sendo assim, modelo tecnologicamente neutro é muito necessário no desenvolvimento de sistemas web. Como consequência, WMLs precisam fornecer a capacidade de representar estes conceitos de arquitetura e design de uma forma tecnologicamente neutra.

3.2.1.1.4 Capacidade de modelar funcionalidades sofisticadas do sistema

Além de sites simples, sistemas web também são derivados de sistemas de processamento de transações convencionais. Estes sistemas suportam processos críticos de

negócio e fluxos de trabalho que são uma parte importante do modelo de negócio central da organização. Estas funções de negócio cruciais estão cada vez mais sendo implementadas através de interfaces web para aplicativos legados, ou através de novos sistemas baseados na web. Portanto, uma WML precisa fornecer a capacidade de representar estas funções e seus artefatos de design relacionados. Exemplos incluem a modelagem de fluxos de trabalho de negócios, informações e acompanhamento de pedidos, processamento de transações, etc.

3.2.1.2 Requisitos de arquitetura de informação

Arquiteturas de informação da web geralmente cobrem aspectos como conteúdo e como ele é gerenciado, estruturação e acesso à informação. Além de contextualização do usuário, design e apoio para navegação do ponto de vista de informação e questões de apresentação. Nos tópicos abaixo serão apresentados alguns dos requisitos desta arquitetura.

3.2.1.2.1 Capacidade de conceitos para modelar em nível de apresentação

Comparado com os sistemas convencionais de software (que são executados em ambiente local), o modelo de apresentação do sistema web tem suas próprias características únicas, que incluem:

- Funções mais completas e sofisticadas para o nível de apresentação, como consequência da necessidade do aumento da qualidade de interface dos usuários;
- Vários tipos de mídia a nível de apresentação, tais como texto, gráfico, vídeo e áudio;
- WMLs serão utilizados não apenas por profissionais de TI, mas também, por exemplo, designers gráficos, produtores de multimídia, autores e analistas de mercado.

Neste sentido, WMLs devem ser capazes de apoiar em uma modelagem de forma que contemple estas características únicas.

3.2.1.2.2 Capacidade de modelar a estrutura de navegação e comportamento

A navegação de um sistema web pode ser contextual ou não contextual. Enquanto ela precisa ser apoiada por uma ontologia bem estruturada de informações na web, a navegação contextual exige mais consideração, tais como regras e restrições relacionadas com o contexto, bem como informações que devem ser passadas ao longo da navegação.

3.2.1.2.3 Capacidade de modelar interações do usuário com a informação

Comparado aos sistemas convencionais de software (que são executados em ambiente local), sistemas web tendem a proporcionar iterações muito mais sofisticadas com seus usuários. Portanto, WMLs devem fornecer a capacidade de modelar essas iterações diferentes de uma forma completa, inequívoca e flexível para que as iterações possam ser capturadas, entendidas, projetadas, implementadas e mantidas.

3.2.1.2.4 Capacidade de modelar funções de usuários e grupos de usuários

O sucesso de um sistema Web é largamente dependente da satisfação do usuário, o que é conseguido, por exemplo, fornecendo funcionalidades sofisticadas, assim como interface e arquitetura de navegação bem estruturada e fáceis de utilizar. Uma técnica fundamental para alavancar todas essas características desejáveis juntas e entregá-las para as pessoas certas, na hora certa, no formato certo é através da personalização. WMLs devem ser capazes de fornecer recursos, incluindo a modelagem de definição e manutenção de perfis de usuário, a definição de grupos de usuários, a conexão entre os usuários e grupos de usuários, e a ligação entre perfis de usuários com grupos e iterações do usuário.

3.2.1.2.5 Capacidade modelar conteúdo

A gestão eficaz de conteúdo é um fator crítico de sucesso de desenvolvimento de sistemas Web. Gestão de conteúdo envolve geralmente a elaboração da estrutura do conteúdo, na maior parte expressadas no formato de base de dados de esquemas, hierarquias de documentos e hiperligações. A tendência potencial é avançar para uma norma internacionalmente aceita de definição de estrutura de conteúdo. As tendências atuais sugerem que o XML é um padrão provável, pelo menos no futuro previsível.

3.2.1.3 Requisitos Gerais

Além de suportar os requisitos de arquitetura funcional e arquitetura funcional, também é fundamental que uma WML forneça algumas capacidades genéricas, como a vinculação dos aspectos funcionais e de informação de uma forma consistente e coesa. Estes requisitos genéricos são apresentados nos tópicos a seguir.

3.2.1.3.1 Capacidade de modelar conceitos de negócio do domínio

Pesquisas e experiência práticas sugerem que há uma ligação muito forte entre modelos de negócios e arquitetura técnica em sistemas Web. Sabendo-se disso, a qualidade da arquitetura técnica dependerá em grande parte da compreensão do modelo de negócio pelos desenvolvedores. Para facilitar e documentar esse entendimento, WMLs precisam fornecer capacidade de conceitos de domínio de negócio, por exemplo, processos de negócios,

entidades empresariais, fluxos de trabalho, regras de negócios, juntamente com os papéis e responsabilidades dos usuários.

3.2.1.3.2 Capacidade de vincular modelo de negócio com a arquitetura técnica

Uma vez entendido e documentado, o modelo de negócio precisa ser efetivamente traduzido em uma arquitetura técnica para que a funcionalidade desejada do sistema possa ser implementada e entregue. Para dar suporte a este requisito, WMLs precisam fornecer a capacidade de identificar a ligação entre o modelo de negócios e a arquitetura técnica.

3.2.1.3.3 Capacidade de vincular arquitetura de informação com a arquitetura funcional

A integridade e coesão entre arquitetura da informação e arquitetura funcional são fundamentais para que o sistema web respeite as necessidades do negócio de forma eficaz. Sendo assim, uma WML deve não apenas apoiar na modelagem de ambas arquiteturas, mas ainda permitir uma integração delas de forma coesa.

3.2.1.3.4 Capacidade de manter a integridade do sistema

A integridade do sistema pode ser comprometida por:

- A complexidade em grande escala, conteúdo e sistemas web com funções ricas;
- As mudanças nos requisitos de negócio causadas pela incerteza do cliente;
- As rápidas mudanças de tecnologias;
- A evolução contínua da funcionalidade e da estrutura do sistema;
- A combinação de várias disciplinas e áreas tecnológicas.

No entanto, a integridade do sistema é um dos atributos críticos de qualidade que o desenvolvimento do sistema web precisa alcançar. WMLs podem apoiar neste objetivo fornecendo definições semânticas de ambos os elementos do modelo e as inter-relações entre eles.

3.2.1.3.5 Capacidade de apoiar o entendimento e comunicação

Para facilitar a comunicação durante o desenvolvimento e manutenção de sistemas web, WMLs precisam fornecer a capacidade de representar aspectos do sistema em diferentes níveis de abstração e de diferentes pontos de vista.

Níveis de abstração

Durante o clico de vida de um sistema, os modelos podem ser utilizados para apoiar diferentes tarefas que serão executados por diferentes equipes que possuem habilidades restritas. A seguir algumas tarefas que podem exigir um diferente nível de abstração do modelo:

- Engenharia de requisitos;
- Análise de sistemas;
- Design, estruturação;
- Apresentação de informações;
- Integração de componentes;
- Especificação de hardware;
- Manutenção.

Uma implicação desta exigência é que WMLs precisam, além de proporcionar a possibilidade de definição de modelos em diferentes níveis de abstração, mas também apoiar a coesão de todo o modelo, proporcionando interconexões coordenadas entre esses níveis de abstração.

Diferentes pontos de vista

Com o rápido aumento de complexidade de sistemas Web, é importante que uma WMLs forneça diferentes artefatos de modelagem que suportem vários pontos de vista. Ao olhar para o sistema a partir de diferentes perspectivas específicas, os desenvolvedores podem quebrar um sistema web grande e complexo em partes ou camadas que são mais fáceis de gerenciar. A seguir alguns pontos de vista possíveis:

- Modelo de negócio e processos de negócio;
- Apresentação e design de interface de usuário;
- Navegação;
- Gerenciamento de conteúdo da informação;
- Os requisitos funcionais e aspectos de design associados;
- Integração;
- Requisitos não-funcionais e seu impacto no projeto do sistema;
- Estrutura do sistema de nível de implementação;
- Personalização e gerenciamento de perfil de usuário.

É desejável que quando um aspecto da arquitetura é alterado, o impacto dessa mudança possa ser identificado e destacado nos aspectos relacionados, ou seja, a manutenção da inter-relação entre os pontos de vista da modelagem.

3.2.1.3.6 Capacidade de ser um processo independente

Os processos de desenvolvimento de software e metodologias convencionais não podem ser utilizados cegamente, sem modificações e aprimoramento. No entanto, abordagens da engenharia de software existentes mostram valor no processo de desenvolvimento de sistemas web, necessitando apenas serem selecionados e ajustados para lidar com as características únicas de sistemas web.

Para permitir esta utilização, WMLs precisam ser capazes de adaptar seus artefatos de modelagem, para que estes possam ser utilizados para um processo personalizado. Para que isto seja possível é necessária alguma independência, apesar de manter a coesão em relação ao sistema todo.

3.2.1.3.7 Capacidade de apoio na gestão do ciclo de vida do sistema

Para a maioria dos sistemas de software, incluindo sistemas Web, o estágio de desenvolvimento é apenas o início de seus ciclos de vida. Além disso, sistemas web demonstram evolução refinada durante seu ciclo de vida, desta forma, a manutenção desempenha um papel cada vez mais importante. WMLs devem facilitar o gerenciamento do ciclo de vida do software, produzindo artefatos de modelagem que são adequados não apenas nas fases de projeto e desenvolvimento, mas também para a manutenção e evolução da arquitetura do sistema web.

3.2.1.4 Comparação entre linguagens

Entre as linguagens de modelagem web que surgiram na última década, Gu, Henderson-Sellers e Lowe (2002) selecionaram algumas para um estudo e avaliação detalhada. De acordo com os requisitos relevantes para o domínio que foram estudados, é possível medir a capacidade através de uma analise de pontos fortes e fracos das linguagens de acordo com a Tabela 3.1.

Abaixo segue uma breve descrição sobre as linguagens que foram estudadas e avaliadas:

OOHDM (*Object Oriented Hypermedia Design Method*): Essa linguagem voltada para a construção de aplicações hipermídia baseada no paradigma de orientação a objetos com notações parecidas com a UML (ROSSI, 1996 *apud* BIANCHINI, 2008, p. 20).

WebML (*Web Modelling Language*): Essa linguagem aborda uma notação para especificação de aplicações Web complexas (CERI et al., 2000b *apud* BIANCHINI, 2008, p. 25).

WAE (Conallen) (*Web Application Extension*): Essa linguagem aborda um conjunto de novos elementos que estendem a linguagem de modelagem UML (CONALLEN, 2002 *apud* BIANCHINI, 2008, p. 31).

W2000: Essa é uma linguagem baseada em dois componentes previamente criados, UML e HDM (*Hypertext Design Model*), e aborda aspectos funcionais e navegacionais (BARESI et al., 2001a *apud* BIANCHINI, 2008, p. 29).

HDM-lite/Autoweb: Essa linguagem descende dos modelos ER (Entidade e Relacionamento) e HDM e foca no desenvolvimento de aplicações Web (FRATERNALI, PAOLINI, 2000).

Com base nas linguagens citadas, abaixo segue os resultados obtidos da avaliação feita por Gu, Henderson-Sellers e Lowe (2002).

Tabela 3.1 - Comparação de capacidades entre linguagens de modelagem para WebApp.

Capacidade de	**OOHDM**	**WebML**	**WAE - Conallen**	**W2000**	**HDM-lite/ Autoweb**
Requisitos de arquitetura funcional					
Modelar integração e conectividade	Não	Não	Parc.	Parc.	Parc.
Suportar um padrão de modelagem	Sim	Não	Parc.	Parc.	Não
Representar conceitos de forma tecnologicamente neutra	Parc.	Sim	Parc.	Sim	Sim
Modelar funcionalidades sofisticadas do sistema	Não	Não	Parc.	Não	Não
Requisitos de arquitetura de informação					
Modelar conceitos em nível de apresentação	Sim	Sim	Não	Não	Sim
Modelar a estrutura de navegação e comportamento	Sim	Sim	Parc.	Parc.	Sim
Capacidade de modelar interações do usuário com a informação	Parc.	Sim	Parc.	Sim	Sim
Modelar funções de usuários e grupos de usuários	Não	Sim	Parc.	Sim	Sim
Modelar conteúdo	Sim	Sim	Parc.	Sim	Sim
Requisitos Gerais					
Modelar conceitos de negócio do domínio	Parc.	Parc.	Parc.	Parc.	Parc.
Vincular modelo de negócio com a arquitetura técnica	Parc.	Parc.	Parc.	Parc.	Parc.
Vincular arquitetura de informação com a arquitetura funcional	Não	Não	Não	Parc.	Não
Manter a integridade do sistema	Não	Parc.	Não	Parc.	Parc.
Apoiar o entendimento e comunicação	Sim	Sim	Parc.	Sim	Sim

Capacidade de	**OOHDM**	**WebML**	**WAE - Conallen**	**W2000**	**HDM-lite/ Autoweb**
Ser um processo independente	Não	Sim	Sim	Não	Não
Apoio na gestão do ciclo de vida do sistema	Parc.	Parc.	Parc.	Parc.	Parc.

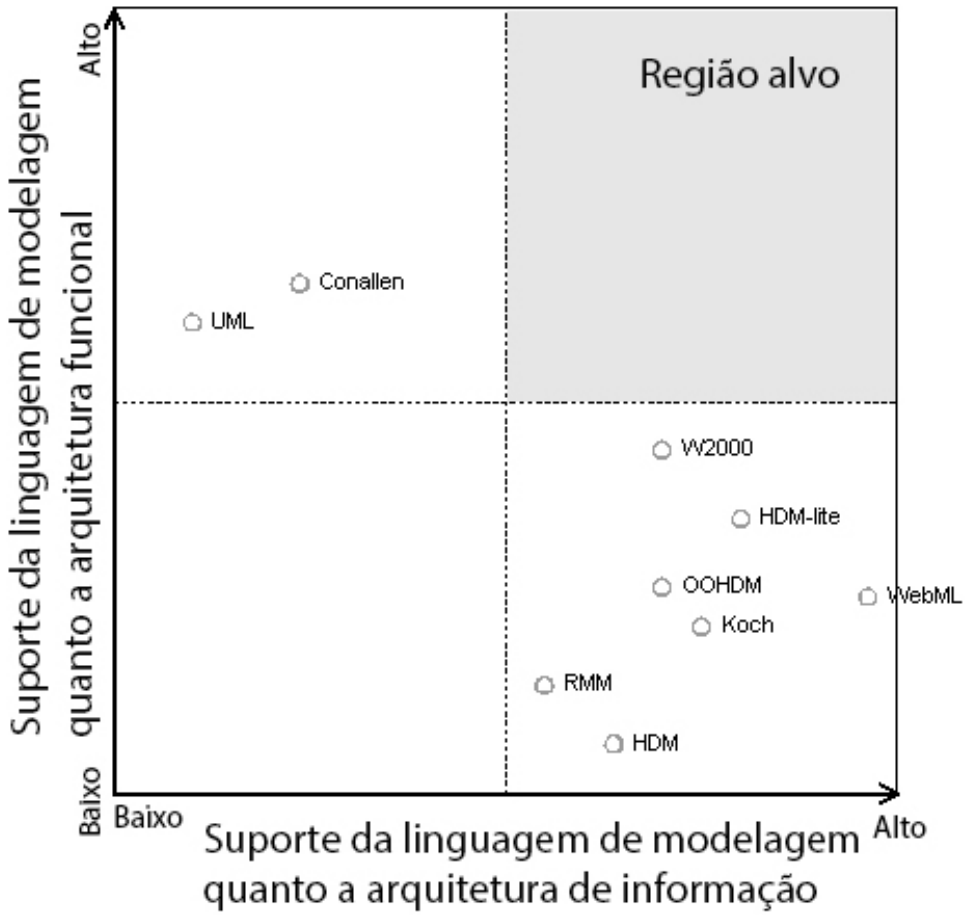
Fonte: Adaptado de (GU, HENDERSON-SELLERS, LOWE, 2002).

Analisando os resultados da tabela acima, é possível traduzi-los para uma representação visual em termos de eficácia de cada WML em relação a arquitetura funcional e arquitetura de informação.

Para a criação da Figura 3.1, a Tabela 3.1 foi traduzida de forma que critérios positivos valem 1 ponto, critérios parciais valem 0,5 pontos, os negativos valem 0 pontos e todos resultados foram somados.

Para fins de comparação, três outras linguagens foram incluídas no gráfico, UML, que é uma linguagem de modelagem padrão de mercado, e também HDM (*Hypermedia Design Model*) e RMM (*Relationship Management Model*), que foram as primeiras linguagens de modelagem para hipertexto.

Figura 3.1 - Análise de abordagens de modelagem existentes



Fonte: Adaptado de (GU, HENDERSON-SELLERS, LOWE, 2002).

A Figura 3.1 demonstra que as abordagens de modelagem existentes não fornecem suporte simultâneo para ambas arquiteturas. Além disso pode-se observar que WAE (Conallen) é a linguagem que fornece maior suporte para a arquitetura funcional, enquanto WebML demonstra uma grande vantagem na modelagem da arquitetura de informação (GU, HENDERSON-SELLERS, LOWE, 2002).

4 A LINGUAGEM: WEBML E IFML

A Linguagem de modelagem Web (WebML), foi concebida em 1998 com o objetivo de apoiar na elaboração e implementação de aplicativos web que acessam e mantém grandes quantidades de dados estruturados, normalmente armazenados como registros em sistema de banco de dados. As aplicações podem variar entre comércio eletrônico, sites institucionais de organizações públicas e privadas, bibliotecas digitais, portais corporativos e sites de comunidades (BRAMBILLA et al., 2007).

Neste contexto, WebML reutilizou um conceito existente de modelo de dados e propôs uma notação original para expressar a navegação e composição de recursos de interface de hipertexto (BRAMBILLA et al., 2007).

Recentemente sugiram iniciativas para que a linguagem WebML passasse a se tornar uma linguagem padrão reconhecida pela OMG. Para que isto fosse possível surgiu uma nova linguagem chamada IFML (*Interaction Flow Modeling Language*), a qual foi resultado de três anos de trabalho envolvendo a OMG e a WebRatio (WEBRATIO, 2013a). Segundo a documentação oficial da IFML (OMG, 2013), a linguagem IFML foi o resultado de 15 anos de experiência de desenvolvimento baseado em modelos para interfaces web adquiridos pela WebRatio e Politecnico di Milano, sendo assim, a WebML foi uma fonte de inspiração para a concepção da IFML.

Com isso, em março de 2013 a linguagem IFML foi finalmente adotada como padrão pela OMG, porém algumas adaptações tiveram que ser feitas em relação a WebML, como o nome dos elementos e a representação visual das notações que compõem um modelo (WEBRATIO, 2013a).

Segundo o site oficial da WebML (2013), a linguagem fornece gráficos e especificações que são incorporados em um processo de modelagem completo, o qual pode ser auxiliado por uma ferramenta de modelagem. Os principais objetivos do processo de modelagem são:

1. Expressar a estrutura de uma aplicação web em um alto nível de descrição, o qual pode ser utilizado para consulta, evolução e manutenção da aplicação.
2. Fornecer múltiplas visões de um mesmo conteúdo.
3. Separar o conteúdo de informação da composição das páginas, navegação e apresentação, desta forma ambos podem evoluir de forma independente.

4. Armazenar meta-informações coletadas durante o processo de desenvolvimento dentro de um repositório, que pode ser usado durante o tempo de vida da aplicação para, dinamicamente, gerar páginas web.
5. Modelar usuários e grupos explicitamente no repositório, de forma que permita a especificação e personalização de políticas e permissionamento.
6. Permitir a especificação de operações de manipulação de dados para atualizar o conteúdo do site ou interagir com serviços externos.

A fim de alcançar os objetivos citados acima, a linguagem contempla quatro dimensões de abstração, cada uma delas associadas a um dos seguintes modelos distintos (CERI et al., 2001):

O **Modelo de Estrutura** descreve a organização dos dados do site de forma conceitual, em termos de entidades e relacionamentos.

O **Modelo de Apresentação** permite expressar a apresentação gráfica da página por meio de definição de regras padrões de renderização independente da linguagem final escolhida.

O **Modelo de Hipertexto** permite definir um ou mais hipertextos que serão publicados no site e seus relacionados aos demais hipertextos que constituem a aplicação. A visão do site é subdividida em dois submodelos: modelo de composição, que abrange o conteúdo das páginas, e modelo de navegação, que expressa como as páginas e conteúdos estão ligados.

O **Modelo de Derivação** permite estender o modelo estrutural e adaptá-lo a diferentes requisitos.

Cada um dos modelos citados descrevem um nível conceitual da aplicação, desta forma obtém-se uma independência em relação as linguagens de implementação de cada componente. As seções a seguir apresentam a maioria dos recursos que caracterizam cada modelo (CERI et al., 2001).

4.1 MODELO DE ESTRUTURA

O modelo de estruturas descreve a organização dos dados que serão utilizados por um *site*. Esta organização é representada a partir uma adaptação de modelos conceituais já conhecidos, sendo, desta forma, compatível com o modelo de dados Entidade e Relacionamento e com o diagrama de classes UML (CERI et al., 2001).

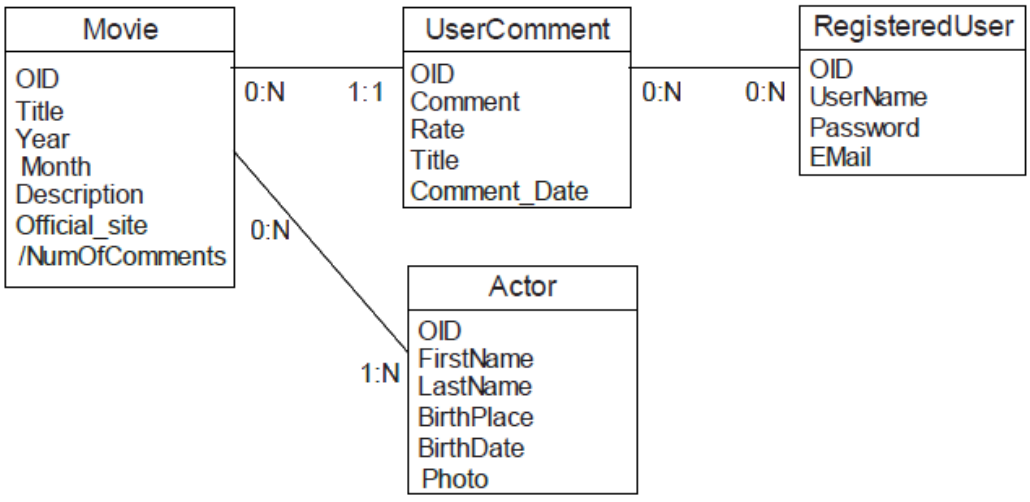
Existem basicamente dois elementos fundamentais utilizados no modelo de estrutura, que são (CERI et al., 2001):

Entidades: Definido como containers de elementos de dados, cada entidade possui propriedades que são nomeadas atributos, onde cada atributo possui um tipo (inteiro, string, booleano, etc) associado. Além disso, toda entidade deve possuir um ou mais atributos identificadores, que irão garantir a unicidade de cada registro na aplicação.

Relacionamentos: Definido como conexões semânticas entre as entidades, são representados por linhas que ligam duas entidades. Sendo que estas conexões podem ser restringidas por meio de restrições de cardinalidade.

A Figura 4.1 ilustra um modelo de estrutura de uma aplicação simples de filmes. Observa-se que neste modelo existem quatro entidades (Movie, UserComment, RegisteredUser e Actor) e seus relacionamentos. Além disso é possível observar que um "Movie" pode ter vários ou nenhum "Actor" associados, representado pelo indicador de cardinalidade 0:N, já um "Actor" deve necessariamente estar relacionado com um "Movie", ou mais. Além disso, observa-se que cada entidade possui um atributo chamado OID (*object identifier*), que está representando a chave de unicidade.

Figura 4.1 - Um esquema de dados para uma aplicação de filmes



Fonte: (BRAMBILLA et al., 2007)

4.2 MODELO DE HIPERTEXTO

Um modelo de hipertexto especifica a composição e navegação de um *site*, para isto este modelo permite definir a interface de *front-end* que será apresentada para o usuário final. A estrutura de *front-end* de uma aplicação é definida em *site views, areas, pages e view components* (CERI et al., 2001).

Site views são definidos no topo do esquema, geralmente criados para atender um conjunto específico de requisitos. Muitas vezes o *site* é organizado de forma que uma *site view* represente a visão de um grupo de usuários ou a visão da organização do conteúdo quando acessado de diferentes dispositivos (por exemplo *smart phones*) (CERI et al., 2001).

Areas são as seções principais de um *site view*, elas são utilizadas para organizar uma *site view* e podem compreender páginas ou outras *areas* recursivamente (CERI et al., 2001).

Pages são containeres de informações que serão entregues ao usuário (CERI et al., 2001).

View components são componentes utilizados para incluir conteúdo nas páginas. Existem também componentes para especificar operações o procedimentos, estes são chamados de *operation units* (CERI et al., 2001).

Para ilustrar esta organização apresentada acima, observa-se que a Figura 4.2 é composta por uma *site view* chamada **MOVIE DB**, sendo que dentro desta *site view* encontra-se uma *page* chamada **HomePage** e duas *areas*, **ShoppingCart Area** e **Moviews Area**. Além disso a **ShoppingCart Area** contém uma *page* chamada **ShoppingCart Data**, enquanto a **Moviews Area** contém três *pages* que são **RecentMoviewsList**, **SearchMovies** e **InsertComment**.

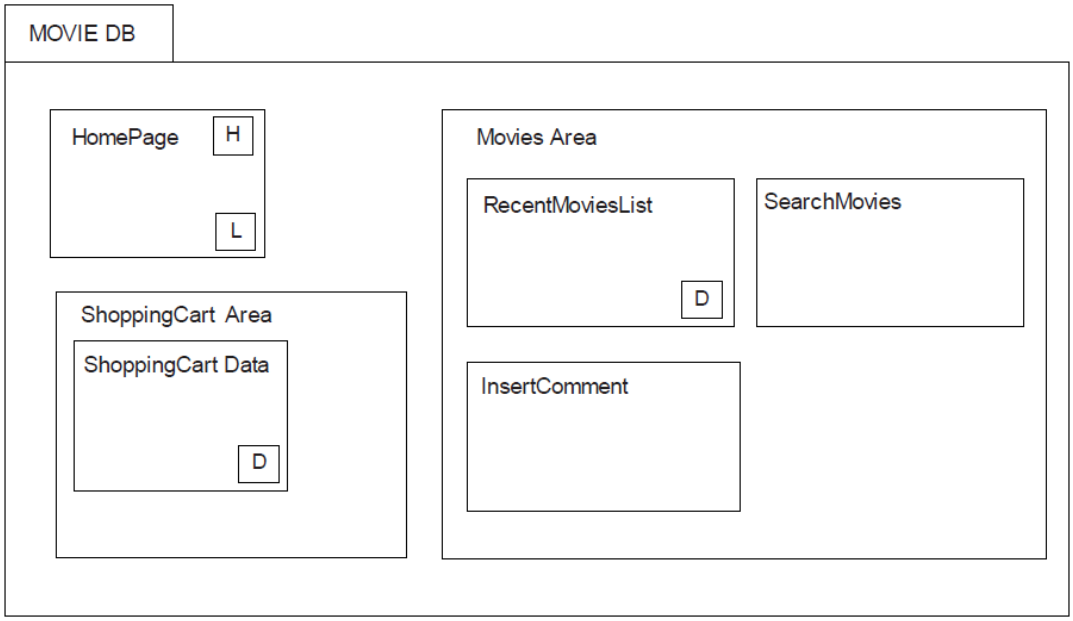
Pages e *areas* são caracterizadas por algumas propriedades relevantes, no exemplo da Figura 4.2 existem três tipos em especial (BRAMBILLA et al., 2007):

A ***home page*** (demarcada por um ícone com a letra "H") é a primeira página a ser apresentada quando um usuário acessa a aplicação, sendo que só existe uma na aplicação inteira.

A ***default page*** (demarcada por um ícone com a letra "D") é a primeira página a ser apresentada quando uma determinada *area* é acessada. Só existe uma para cada *area*. No exemplo da Figura 4.2, **Shopping Cart Data** e **Recent Movies List** são as *default pages* de suas *areas*.

A ***landmark page*** (demarcada por um ícone com a letra "L"), é acessível a partir de todas as outras *pages* ou *areas* dentro de seu módulo. Por exemplo, na Figura 4.2 da *home page* também é uma *landmark page*, o que significa que um *link* para ela estará disponível a partir de qualquer outra *page* do ponto de vista local.

Figura 4.2 - Exemplos de *site view* baseado em *areas* e *pages*



Fonte: (BRAMBILLA et al., 2007)

Neste contexto apresentado, o modelo de hipertexto é subdividido em dois outros modelos distintos, que são o modelo de composição e o modelo de navegação, abaixo serão abordados quais elementos compõem cada um destes submodelos.

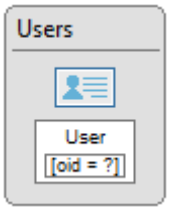
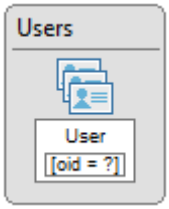
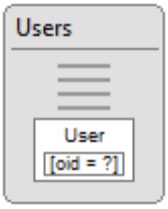
4.2.1 Modelo de composição

O modelo de composição especifica quais *pages* compõe o hipertexto, e quais *view components* compõe uma *page*. Além disso, existem os componentes utilizados para especificar operações, utilizados para gerenciar conteúdo ou procedimentos de autenticação de usuários, são chamados de *operation units*. Nas próximas seções serão apresentados os principais componentes disponíveis (CERI et al., 2001).

4.2.2 View Components

Conforme visto anteriormente, *view components* são componentes de conteúdo atômicos utilizados para publicar informações no modelo, cinco tipos básicos podem ser utilizados para compor uma *page* e podem ser visualizados na Tabela 4.1.

Tabela 4.1 - Componentes de conteúdo básicos.

Details	Multiple Details	Simple List	Scroller	Form
				

Fonte: (WEBML, 2003a) e (WEBRATIO, 2013b)

O elemento ***Details*** é utilizado para publicar um objeto único oriundo de uma entidade, ou seja, irá obter informações sobre um registro da base de dados e exibir suas informações (WEBML, 2003a).

O elemento ***Multiple Details*** é utilizado para apresentar múltiplos objetos oriundos de uma entidades juntos, repetindo o processo de um *Details* por várias vezes, ou seja, diferentemente do *Details* que apresenta informações a respeito de um registro, irá apresentar informações de vários registros (WEBML, 2003a).

O elemento ***Simple List*** é utilizado para exibir uma lista com múltiplos registros de uma entidade, geralmente em forma de tabela. Ainda existem outros dois elementos que são extensões deste, que são o c***heckable list***, que exibe uma tabela com a possibilidade de selecionar uma ou mais linhas, e o ***hierarchy***, que exibe os registros organizados em forma de árvore com multinível (WEBML, 2003a).

O elemento ***Scroller*** fornece comandos para rolar através de um conjunto de elementos.

O elemento ***Form*** é utilizado para a criação de formulários, sendo que cada campo do formulário deve ser criado dinamicamente, podendo referenciar um elemento de uma entidade (WEBML, 2003a).

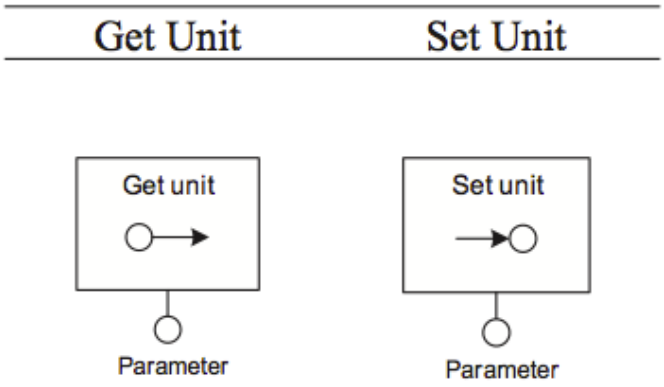
4.2.3 Operation Units

As unidades de operações, chamado de *operation units*, são utilizadas para especificar parâmetros, manipulação de dados e procedimentos.

Um dos tipos de unidades disponíveis são as unidades de parâmetros globais. Sendo que parâmetros globais são utilizados para armazenar e recuperar informações globais que são definidos uma vez durante a execução e podem ser recuperadas a qualquer momento, sem a necessidade de transferir parâmetros durante a navegação (BRAMBILLA et al., 2007). Para a

manipulação destes parâmetros, dois elementos demonstrados na Figura 4.3 podem ser utilizados.

Figura 4.3 - Unidades e parâmetros globais



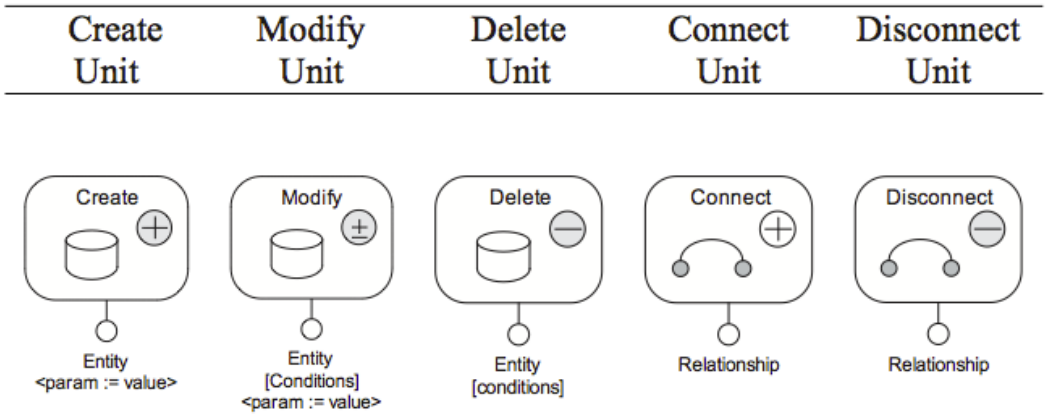
Fonte: (WEBML, 2003a)

O elemento ***Get*** é utilizado para definir um valor para um atributo de variável global (WEBML, 2003a).

O elemento ***Set*** é utilizado para obter o valor de um atributo de variável global (WEBML, 2003a).

Também existem as unidades de operações, chamadas de *operation units*, que fornecem a capacidade de criar, atualizar e remover registros da base de dados, além de criar relacionamentos entre registros. Graças a estas operações é possível criar sites dinâmicos. A Figura 4.4 demonstra as operações possíveis.

Figura 4.4 - Unidades de operação



Fonte: (WEBML, 2003a)

O elemento ***Create*** permite a criação de um registro em alguma entidade determinada, para isto é necessário informar os dados do novo registro como parâmetro (WEBML, 2003a).

O elemento ***Modify*** permite alterar um ou mais registros em alguma entidade determinada, para isto é necessário informar os dados a serem alterados, além de o critério para definir quais registros devem ser alterados (WEBML, 2003a).

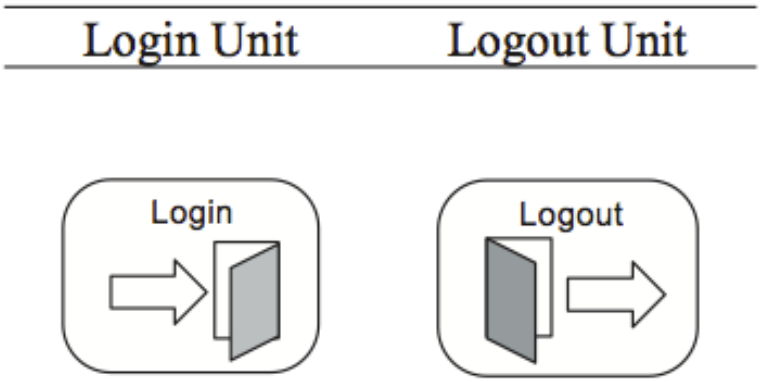
O elemento ***Delete*** permite remover um ou mais registros de uma entidade determinada, para isto é necessário informar critérios para definir quais registros devem ser removidos (WEBML, 2003a).

O elemento ***Connect*** permite criar uma relação entre dois registros da base de dados, por exemplo, é possível associar um usuário a outro como uma relação de amigos em uma rede social (WEBML, 2003a).

O elemento ***Disconnect*** permite remover a relação entre dois registros da base de dados (WEBML, 2003a).

Ainda, as unidades de operação de *login* e *logout* são utilizadas para manipular o controle e verificação da identidade do usuário que esta acessando a aplicação web. Sendo capazes de criar, visualizar e fechar sessões de autenticação de usuário. As duas operações estão representadas na Figura 4.5.

Figura 4.5 - Operações de *login* e *logout*



Fonte: (WEBML, 2003a)

O elemento ***Login*** verifica os dados de usuário e senha de um usuário na base de dados e permite autenticar o mesmo no site, através do registro de alguns atributos de variáveis globais (WEBML, 2003a).

O elemento ***Logout*** limpa todos os atributos de variáveis globais referentes a autenticação de usuário, desta forma o usuário perde sua autenticação no site (WEBML, 2003a).

4.2.4 Modelo de navegação

O propósito do modelo de navegação é especificar *links* entre *view components* e *pages*, sendo que as ligações podem ser entre unidades dentro de uma mesma página, unidades de páginas diferentes ou entre páginas. Entre os objetivos de um *link* estão mover o foco da apresentação de uma página para outra, passar informações de uma unidade para outra ou disparar a execução de alguma ação (CERI et al., 2001).

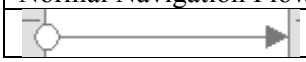
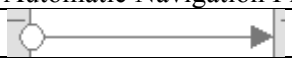
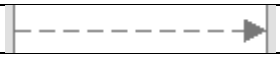
A informação transportada ao longo de uma ligação realizada por um *link* é chamada de contexto de navegação, sendo que existem dois tipos de ligações (CERI et al., 2001):

Ligação contextual: São *links* que transportam informações de contexto, por exemplo, ao acessar um item de uma lista, o *link* para visualizar mais detalhes contém a informação do ID único do registro que será aberto na página seguinte.

Ligação não contextual: São *links* que não tem associado informações de contexto, por exemplo, um *link* para uma área estática da página.

A Tabela 4.2 demonstra as opções de *links* existentes.

Tabela 4.2 - Elementos que representam *links*.

Normal Navigation Flow	Automatic Navigation Flow	Data Flow
		

Fonte: (WEBML, 2003a) e (WEBRATIO, 2013b)

Um *link* do tipo ***normal navegation flow***, é acionado ao clique do usuário e conforme explicado anteriormente, pode ser contextual ou não contextual.

Um *link* do tipo ***automatic navegation flow*** é acionado de forma automática, sem a necessidade de interação do usuário (WEBML, 2003a).

Um *link* do tipo ***data flow*** é utilizado para passar parâmetros de uma unidade para outra, porém não é percebido na tela (WEBML, 2003a).

Os modelos de composição e navegação estão totalmente integrados, uma vez que as características de um modelo pode afetar o outro, por exemplo, para exibir o conteúdo de uma página, os *links* apropriados devem permitir o acesso. Sendo assim, a soma dos dois modelos forma o modelo de hipertexto, o qual define os conteúdos e ligações de um *site* como um todo.

4.3 MODELO DE DERIVAÇÃO

A derivação é uma fase da modelagem que torna possível acrescentar atributos a entidades do esquema estrutural, a partir da importação ou cálculos feitos através de objetos

relacionados, por exemplo, é possível criar um atributo em uma tabela de estados chamado “TotalDeCidades”, que automaticamente calcula quantas cidades relacionadas a esse estado existem na base de dados (WEBML, 2003b).

A derivação é possível através da criação de expressões que são chamadas de *derivation queries* , as quais podem ser transformadas em *SQL views*, nativas do banco de dados escolhido. Pode-se derivar quatro tipos de atributos (WEBML, 2003b):

Constant attribute: Este tipo de atributo é sempre constante, por exemplo, um texto "Estudo da WebML".

Imported attribute: Este tipo de atributo é a importação de um atributo de outra tabela que tenha relação com a de origem, por exemplo, na tabela livro é possível importar o atributo “Nome” da tabela “Autor”, utilizando-se a referência do Id do autor que existe na tabela livro.

Aggregate attribute: Este tipo de atributo utiliza funções de agregação, um exemplo é criar um atributo "Total de livros" na tabela autor, o qual irá somar todos os livros que estão relacionados com o autor.

Calculated attribute: Este tipo de atributo permite a realização de operações com atributos da entidade, um exemplo é a multiplicação do valor monetário de um livro com o valor de desconto, para obter um novo atributo que indica o valor que deve ser cobrado.

A derivação é formalmente expressada por consultas em uma linguagem conceitual chamada WebML-OQL, o qual se aplica aos elementos do esquema estrutural e constrói conceitos adicionais. A utilização de entidades derivadas, permite criar índices significativos de acesso ao conteúdo do site (WEBML, 2003b).

4.4 MODELO DE APRESENTAÇÃO

O modelo de apresentação é responsável por definir a experiência de uso dos usuários através da criação de estilos para o *layout* definido no modelo de composição. Para isso, a linguagem não fornece um modelo específico para expressar a apresentação, ao invés disso, utiliza-se de abordagens padrões utilizadas no contexto de sistemas web, que acabam sendo familiares para os especialistas gráficos e da comunicação (CERI et al, 2001).

A especificação da apresentação é feita através de folhas de estilo construídas em XSL (*Extensible Stylesheet Language*) que são anexadas ao site, desta forma a criação das folhadas de estilo independem da linguagem utilizada para o desenvolvimento da aplicação.

Sendo que uma implementação pode incluir várias folhas de estilo, tanto para o layout da aplicação inteira, como para conteúdos específicos (CERI et al., 2001).

4.5 A FERRAMENTA WEBRATIO

Um dos princípios do projeto original da WebML, é o objetivo de tornar as modelagens executáveis, através da utilização dos conceitos de MDD. Para atingir este objetivo, em 2001 a Universidade Politecnico de Milano criou uma empresa chamada Models Web, com a missão de implementar e comercializar métodos e ferramentas que permitissem a automatização do processo de desenvolvimento através de modelos WebML (BRAMBILLA et al., 2007).

Dentre os resultados alcançados, o que demonstrou melhores resultados foi a ferramenta WebRatio, que fornece um ambiente capaz de modelar aplicações completas utilizando-se de WebML e ainda implementa a geração de código a partir dos modelos utilizando MDD (BRAMBILLA et al., 2007).

Neste contexto, a WebRatio consolidou-se como uma realidade industrial, passando a ser utilizado em muitas universidades e instituições no mundo todo. O que ajudou a aprimorar ainda mais a ferramenta, uma vez que a empresa passou a receber *feedbacks* de seus utilizadores (BRAMBILLA et al., 2007).

Entre as características encontradas no site oficial do WebRatio (2013a), destacam-se:

- O WebRatio é integrado com a IDE (*Integrated Development Environment*) Eclipse, que é um aplicativo conhecido de mercado, sendo que aplicação possuí a mesma aparência, ferramentas e extensões já conhecidas.
- Aplicações construídas com WebRatio, utilizando o padrão Java/JSP 2.0, podem ser implantadas em qualquer aplicativo de servidor, incluindo Apache Tomcat, jBoss, Caucho Resin, Oracle WebLogic Application Server e IBM WebSphere.
- Aplicações construídas com WebRatio, utilizam a biblioteca do Java de manipulação de banco de dados Hibernate, sendo assim, pode ser conectada a qualquer banco de dados que utilize o padrão JDBC (*Java Database Connectivity*), incluindo PostgreSQL, MySQL, Oracle 8i/9i/10g/11g, IBM DB2, Microsoft SQL Server 2000/2005/2008 e Apache Derby.

- Aplicações construídas com WebRatio, utilizam as bibliotecas mais populares do Java, incluindo Hibernate, Struts, JSTL, JSP, e Java Servlet. Além disso, o código gerado é completamente aberto e não utiliza componentes proprietários, de forma que se for necessário a manutenção pode ser feita sem a utilização do WebRatio, mas sim na programação Java convencional.
- A geração de códigos segue regras que proporcionam um código altamente otimizado, sempre preferindo padrões do domínio. Sendo que, novas regras podem ser incorporadas pelos usuários.

A partir da versão 7.2 do Webratio, a linguagem de modelagem WebML foi substituída pelas notações do IFML, que é a nova linguagem de modelagem de aplicações *front-end* padrão da OMG.

5 EXPERIMENTO PRÁTICO

Visando avaliar a utilização da linguagem de modelagem IFML, em termos de produtividade e resultados obtidos a partir da utilização de MDD, este capitulo demonstra o processo e desenvolvimento de um sistema.

Ambos, processo e sistema, são insumos para a avaliação pretendida. Além disso, o capítulo apresenta as etapas práticas do MDD realizadas através da IFML e WebRatio.

5.1 ESTRATÉGIA UTILIZADA

Para a realização do experimento prático, o processo de desenvolvimento utilizado foi baseado no framework apresentado no subcapítulo 3.2 deste trabalho, o qual propõe uma estrutura genérica para auxiliar nas atividades de comunicação, planejamento, modelagem e construção. O framework proposto ainda auxilia na etapa de implementação, porém esta etapa foi suprimida do trabalho, uma vez que o objetivo é avaliar apenas o processo de desenvolvimento do projeto.

A Tabela 5.1 demonstra uma relação entre as etapas executadas no experimento com as fases previstas no framework.

Tabela 5.1 - Relação entre etapas do framework e etapas executadas.

Framework	Etapas do Experimento
Comunicação	Comunicação
Planejamento	Planejamento e Modelagem
Modelagem	
Construção	Construção e Utilização
Implementação	-

Fonte: Autor (2013).

5.2 COMUNICAÇÃO

A etapa de comunicação é a atividade que estabelece um “destino” para o projeto, para isto, algumas tarefas são necessárias para garantir que o projeto siga o caminho correto, este trabalho seguiu as etapas contidas em (PRESSMAN, LOWE, 2009).

Na primeira etapa, **identificar as partes interessadas no negócio**, foi essencial a colaboração da aluna Marina Lopes, estudante de um curso da área de Comunicação na Universidade Feevale. Ela compartilhou as informações de uma aplicação que foi idealizada para a disciplina de Web para Comunicação, além de debater e sanar as duvidas durante o levantamento dos requisitos, sendo assim, ela foi eleita como a mantenedora da aplicação,

uma vez que a ideia proposta pela aluna foi adotada para ser desenvolvida através do processo estabelecido.

Após identificar as partes interessadas, seguiu-se a etapa de **Definir os principais objetivos da aplicação**, sendo assim, pode-se afirmar que a aplicação, chamada Lumni, consiste basicamente em uma rede social que permite aos usuários postarem vídeos que serão compartilhados com outros usuários da rede com o objetivo principal de troca de conhecimentos. Levando em conta a etapa de **identificar os perfis de usuários**, constatou-se apenas o perfil de usuário utilizador do sistema, o qual, para ter acesso, deve primeiramente criar uma conta através do fornecimento de alguns dados. Após, abordou-se as etapas de **reunir requisitos** e **desenvolver cenários de uso**, onde identificou-se as principais ações do usuário como postar vídeos, procurar vídeos, seguir outros usuários, curtir e comentar vídeos, além de outras opções que estarão descritas nos casos de uso. Levando em conta a etapa de **formular o contexto do negócio**, sabe-se que a aplicação estará disponível na internet, através do acesso ao *site*, e além disso, com base nas categorias de aplicações web estudadas no subcapitulo 3.1, o sistema Lumni apresenta características de uma **WebApp de interação**, o que significa que os casos de uso e interface devem ser planejados para privilegiar o envolvimento dos usuários do sistema, através de avaliações, comentários e envio de conteúdo.

5.3 PLANEJAMENTO E MODELAGEM

A etapa de comunicação auxiliou na identificação de um destino para o projeto, contudo, é necessário planejar os meios para se alcançar o objetivo identificado, além de criar representações conceituais para alguns aspectos da aplicação que será construída. Seguindo as etapas propostas por Pressman e Lowe (2009), o processo de planejamento e modelagem chegou nos resultados a seguir.

Primeiramente, cada um dos requisitos identificados foi **detalhado de forma mais precisa** em relação as funcionalidades descritas pela aluna, estes encontram-se delineados nas próximas seções deste capitulo. Além disso, esta etapa preocupou-se em **calcular o esforço necessário** para o desenvolvimento de cada um deles.

Por último, foi realizado o processo de modelagem, iniciando pela **identificação dos requisitos de banco de dados**, e então a modelagem da aplicação **representando os conteúdos da aplicação**, **identificando a relação entre os conteúdos**.

Neste sentido, as próximas seções deste subcapitulo demonstra os resultados dos itens citados anteriormente.

5.3.1 Casos de uso

Segundo COCKBURN (2005), casos de usos descrevem comportamentos do sistema sob diversas condições, conforme o sistema responde a requisições de atores da aplicação.

Um caso de uso serve como comunicação entre uma pessoa e outra, geralmente utilizando-se de textos simples que devem ser compreendido em pouco tempo por quem está interpretando. Não existem um padrão na escrita de caso de uso, porém três conceitos devem ser considerados ao escreve-los (COCKBURN, 2005):

Escopo: Descrição da funcionalidade que está sob discussão;

Ator primário: Usuário que irá interagir com a funcionalidade;

Nível: Um caso de uso pode ser de alto nível, quando possuí uma breve descrição sobre o processo, mas também pode ser um caso de uso estendido, que descreve passo a passo os fluxos de eventos do processo.

Com base na coleta de informações para a aplicação Lumni, os casos de uso estendidos apresentados no **Apêndice A,** baseados no modelo sugerido por Bezerra (2002)**,** tem como objetivo documentar a análise de forma a facilitar o processo de desenvolvimento posterior. As funcionalidades a seguir foram identificadas e transformadas em casos de uso, sempre utilizando como ator o usuário utilizador do sistema.

5.3.1.1 Login no sistema

O login no sistema é o caso de uso, conforme apresentado no **Apêndice A 01- Autenticação no sistema**, responsável por consultar na base de dados às informações sobre um usuário cadastrado previamente no sistema, e validar a existência do login informado e comparar se a senha informada coincide com a senha cadastrada no sistema para esse login. Este método consiste em uma validação de usuário e senha, sendo o seu preenchimento efetuado pelo usuário utilizador do sistema. Além disso, para que a autenticação possa ser realizada, a entidade *Usuario* precisa ser consultada. Após a validação as informações do usuário devem ser registradas na sessão e o sistema redirecionado para a página inicial.

5.3.1.2 Cadastro de usuário

O cadastro de usuário é o caso de uso, conforme apresentado no **Apêndice A 02-Cadastro de usuário**, responsável por inserir na base de dados os usuários que serão utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em um cadastro de informações referentes ao usuário, sendo o seu preenchimento efetuado pelo usuário utilizador do sistema. Além disso, após o cadastro ser realizado o caso de uso **Login no sistema** é automaticamente acionado.

5.3.1.3 Consulta de vídeos recentes

A consulta de vídeos recentes é o caso de uso, conforme apresentado no **Apêndice A 03-Consulta de vídeos recentes**, responsável por consultar na base de dados as informações sobre os últimos 10 vídeos já cadastrados por qualquer usuário que foi previamente sinalizado com a opção de seguir pelo usuário utilizador do sistema, e disponibilizar estes em forma de relatórios para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo sistema. Além do mais, para que os relatórios possam ser gerados, as entidades *Video*, *Usuario* e *Usuario_segue* do sistema precisam ser consultadas.

5.3.1.4 Consulta de usuários seguindo

A consulta de usuários seguindo é o caso de uso, conforme apresentado no **Apêndice A 04-Consulta de usuários seguindo**, responsável por consultar na base de dados as informações sobre todos os usuários do sistema sinalizados na opção de seguir pelo usuário utilizador, e disponibilizar estes em forma de relatórios para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo sistema. Além do mais, para que os relatórios possam ser gerados, as entidades *Usuario* e *Usuario_segue* do sistema precisam ser consultadas.

5.3.1.5 Consulta de usuários seguidores

A consulta de usuários seguindo é o caso de uso, conforme apresentado no **Apêndice A 05-Consulta de usuários seguidores**, responsável por consultar na base de dados as informações sobre todos os usuários do sistema pelos quais o usuário utilizador foi sinalizado com a opção de seguir, e disponibilizar estes em forma de relatórios para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo sistema. Além do mais, para que os relatórios possam ser gerados, as entidades *Usuario* e *Usuario_segue* do sistema precisam ser consultadas.

5.3.1.6 Alteração de dados do usuário

A alteração de dados do usuário é o caso de uso, conforme apresentado no **Apêndice A 11-Alteração de dados do usuário**, responsável por alterar na base de dados as informações de um usuário, que será utilizado nas demais funcionalidades do sistema. Basicamente, esse método consiste em um cadastro de informações com as opções dos campos já pré-determinadas e carregadas pela aplicação, cujo preenchimento é efetuado pelo usuário utilizador do sistema. Além do mais, para que os parâmetros a serem inseridos pelo usuário já sejam automaticamente preenchidos pelo sistema, a entidade *Usuario* precisa ser consultada.

5.3.1.7 Consulta de informações do vídeo

A consulta de informações do vídeo é o caso de uso, conforme apresentado no **Apêndice A 12-Consulta de informações do vídeo**, responsável por consultar na base de dados às informações sobre um vídeo cadastrado por qualquer usuário do sistema, e disponibilizá-lo em forma de relatório para visualização. Este método consiste em uma consulta, que é parametrizada pelo usuário utilizador do sistema. Além disso, para que esse relatório possa ser gerado, as entidades *Video*, *Usuario*, *Categorias*, *Curtir* e *Comentario* precisam ser consultadas.

5.3.1.8 Cadastro de comentário ao vídeo

O cadastro de comentário ao vídeo é o caso de uso, conforme apresentado no **Apêndice A 13-Cadastro de comentário ao vídeo**, responsável por inserir na base de dados os comentários de vídeos que serão utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em um cadastro de informações referentes ao comentário, cujo preenchimento é efetuado pelo usuário utilizador do sistema.

5.3.1.9 Opção de curtir um vídeo

A opção de curtir um vídeo é o caso de uso, conforme apresentado no **Apêndice A 14-Opção de curtir um vídeo**, responsável por inserir na base de dados os registros de curtir vídeos que serão utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em um novo registro de curtir ao vídeo, cujo solicitação é efetuada pelo usuário utilizador do sistema.

5.3.1.10 Consulta de informações do usuário

A consulta de informações do usuário é o caso de uso, conforme apresentado no **Apêndice A 15-Consulta de informações do usuário**, responsável por consultar na base de dados às informações sobre um usuário cadastrado previamente no sistema, e disponibilizá-lo em forma de relatório para visualização. Este método consiste em uma consulta, que é parametrizada pelo usuário utilizador do sistema. Além disso, para que esse relatório possa ser gerado, as entidades *Usuario*, *Usuario_segue* e *Video* precisam ser consultadas.

5.3.1.11 Opção de seguir um usuário

A opção de seguir um usuário é o caso de uso, conforme apresentado no **Apêndice A 16-Opção de seguir um usuário**, responsável por inserir na base de dados os registros de seguir usuários que serão utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em um novo registro de seguir um usuário, cujo solicitação é efetuada pelo usuário utilizador do sistema.

5.3.1.12 Opção de deixar de seguir um usuário

A opção de deixar de seguir um usuário é o caso de uso, conforme apresentado no **Apêndice A 17-Opção de deixar de seguir um usuário**, responsável por remover na base de dados os registros de seguir usuários que deixarão de ser utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em remover o registro de seguir um usuário, cujo solicitação é efetuada pelo usuário utilizador do sistema.

5.3.1.13 Pesquisa de vídeos

A pesquisa de vídeos é o caso de uso, conforme apresentado no **Apêndice A 06-Pesquisa de vídeos**, responsável por consultar na base de dados às informações sobre vídeos já inseridos no sistema por qualquer usuário e disponibilizar esses em forma de um relatório para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo usuário utilizador do sistema, informando o nome do vídeo ou parte do nome e podendo, opcionalmente, escolher uma ou mais categorias que irão ser utilizadas como critério ao buscar apenas vídeos pertencentes a alguma delas. Ainda, para que esse relatório possa ser gerado, as entidades *Video* e *Categoria* precisam ser consultadas.

5.3.1.14 Pesquisa de usuários

A pesquisa de usuários é o caso de uso, conforme apresentado no **Apêndice A 07-Pesquisa de usuários**, responsável por consultar na base de dados às informações sobre

usuários já inseridos no sistema e disponibilizar esses em forma de um relatório para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo usuário utilizador do sistema, informando opcionalmente o nome do usuário, login ou e-mail. Ainda, para que esse relatório possa ser gerado, a entidade *Usuario* precisa ser consultada.

5.3.1.15 Consulta de vídeos de autoria do usuário

A consulta de vídeos de autoria do usuário é o caso de uso, conforme apresentado no **Apêndice A 08-Consulta de vídeos de autoria do usuário**, responsável por consultar na base de dados às informações sobre vídeos já inseridos no sistema pelo usuário utilizador e disponibilizar esses em forma de um relatório para visualização. Basicamente, esse método consiste em uma consulta, que é parametrizada pelo sistema. Ainda, para que esse relatório possa ser gerado, as entidades *Video* e *Categoria* precisam ser consultada.

5.3.1.16 Cadastro de vídeo

O cadastro de vídeo é o caso de uso, conforme apresentado no **Apêndice A 09-Cadastro de vídeo**, responsável por inserir na base de dados os vídeos que serão utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em um cadastro de informações referentes ao vídeo, cujo preenchimento é efetuado pelo usuário utilizador do sistema. Ainda, para que esse formulário de cadastro possa ser gerado, a entidade *Categoria* precisa ser consultada.

5.3.1.17 Exclusão de vídeo

A exclusão de vídeo é o caso de uso, conforme apresentado no **Apêndice A 10-Exclusão de vídeo**, responsável por excluir da base de dados vídeos que deixarão de ser utilizados nas demais funcionalidades do sistema. Basicamente, esse método consiste em uma exclusão, cuja ação é solicitada pelo usuário utilizador do sistema responsável pelo cadastro do vídeo.

5.3.2 Estimativa de esforço

Uma vez que as funções do sistema foram definidas, a etapa de estimativa é importante para prever o tamanho do sistema resultante, sendo que o tamanho pode ser um indicador da complexidade do projeto (PRESSMAN, 2011).

De acordo com PRESSMAN (2011), a métrica ponto de função pode ser usada efetivamente como um meio para medir a funcionalidade fornecida por um sistema. Sendo

que, baseada em dados históricos, pode-se ser utilizada para estimar o custo ou trabalho necessário para projetar, codificar e testar o software; prever o número de erros que serão encontrados durante o teste; e presumir o número de componentes e/ou o número de linhas projetadas de código fonte no sistema a ser implementado. O cálculo dos pontos de função são derivados a partir de uma relação entre medidas calculáveis do domínio de informações do software e avaliações qualitativas da complexidade de software. Os valores do domínio de informações são definidos da seguinte maneira:

Número de entradas externas (EE): Cada entrada externa é originada de um usuário ou a partir outra aplicação. Estas entradas são muitas vezes usadas para atualizar arquivos lógicos internos.

Número de saídas externas (SE): Cada saída externa é formada por dados oriundos da aplicação e fornecem informações para o usuário. Sendo assim, são consideradas saídas externas relatórios, telas, mensagens, etc.

Número de consultas externas (CE): Uma consulta externa é definida como a apresentação de informações ao usuário, geralmente obtida através de arquivos lógicos internos. A principal diferença entre uma consulta externa em relação a uma saída externa, é que a consulta apenas apresenta os dados, enquanto a saída externa pode utilizar cálculos, fórmulas, entre outros.

Número de arquivos lógicos internos (ALI): Cada arquivo lógico interno é armazenado dentro da aplicação e mantido através de entradas externas.

Número de arquivos de interface externos (AIE): Cada arquivo de interface externo é um agrupamento lógico de dados armazenado fora da aplicação, porém fornece informações que podem ser utilizados pela aplicação.

Para definir a complexidade de cada função utiliza-se dois indicadores, os **tipo de dados** (TD) referenciam o número de campos/colunas envolvidos e os arquivos referenciados ou tipo de registros (AR/TR) referenciam o número de entidades que serão envolvidas na função.

De acordo com a analise apresentada no **Apêndice B**, a Tabela 5.2 resume a contagem de pontos de função do sistema Lumni. Sendo que a estimativa total foi de 119 pontos.

Tabela 5.2 - Contagem de pontos de função da aplicação Lumni.

Função	Tipo	TD	AR/TR	Complex.	PF
Tabelas principais					
Vídeo	ALI	5	1	Baixa	7
Usuário	ALI	5	2	Baixa	7

Função	**Tipo**	**TD**	**AR/TR**	**Complex.**	**PF**
Categoria	ALI	2	1	Baixa	7
Comentário	ALI	3	1	Baixa	7
Curtir	ALI	1	1	Baixa	7
Total					35
Página de login					
Logar-se utilizando usuário e senha	EE	2	1	Baixa	3
Criar um cadastro	EE	5	1	Baixa	3
Total					6
Home					
Consultar vídeos recentes de quem estou seguindo	CE	2	2	Baixa	3
Consultar usuários que estou seguindo	CE	1	2	Baixa	3
Consultar usuários que estão me seguindo	CE	1	2	Baixa	3
Seguir um usuário	EE	1	1	Baixa	3
Deixar de seguir um usuário	EE	1	1	Baixa	3
Total					15
Pesquisar vídeos					
Listar categorias que podem ser utilizadas na pesquisa	CE	1	1	Baixa	3
Pesquisar vídeo informando o título e categorias	EE	2	1	Baixa	3
Consultar vídeos que contém o termo informado	SE	4	2	Baixa	4
Total					10
Pesquisar usuários					
Pesquisar usuário informando nome ou login ou e-mail	EE	1	1	Baixa	3
Consultar usuários que contém o termo informado no nome ou login ou e-mail	SE	3	1	Baixa	4
Total					7
Meus vídeos					
Consultar vídeos que sou autor	CE	1	1	Baixa	3
Consultar categorias que poderão ser escolhidas para o novo vídeo	CE	1	1	Baixa	3
Cadastrar novo vídeo	EE	4	1	Baixa	3
Excluir vídeo	EE	1	1	Baixa	3
Total					12
Meus dados					
Consultar meus dados	CE	2	1	Baixa	3
Alterar meus dados	EE	2	1	Baixa	3
Total					6
Detalhes do vídeo					
Consultar dados do vídeo	SE	7	3	Média	7
Consultar comentários do vídeo	CE	3	2	Baixa	3

Função	Tipo	TD	AR/TR	Complex.	PF
Criar novo comentário	EE	1	1	Baixa	3
Curtir o vídeo	EE	1	1	Baixa	3
Total					16
Detalhes do usuário					
Consultar dados do usuário	CE	3	1	Baixa	3
Consultar vídeos do usuário	CE	3	1	Baixa	3
Seguir usuário	EE	1	1	Baixa	3
Deixar de seguir usuário	EE	1	1	Baixa	3
Total					12

Fonte: Autor (2013).

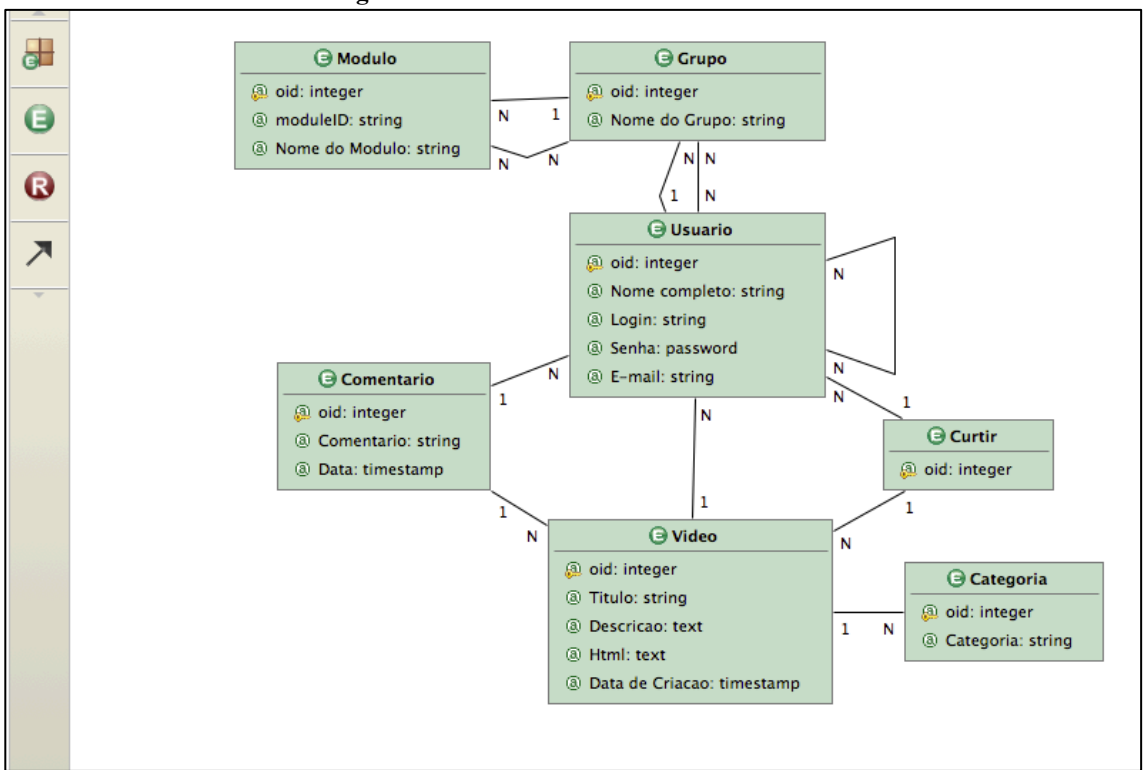
Por último, seguiu-se algumas etapas do processo de modelagem, iniciando pela **identificação dos requisitos de banco de dados**, e então a modelagem da aplicação **representando os conteúdos da aplicação**, **identificando a relação entre os conteúdos**.

5.3.3 Modelagem do banco de dados

Conforme estudado anteriormente, a linguagem IFML utiliza modelos de dados previamente existentes e conhecidos como modelo Entidade-Relacionamento (ER).

Sendo assim, com base nos conceitos do Modelo ER e a partir da analise dos requisitos demonstrados a partir dos casos de uso, a Figura 5.1 demonstra a versão final do modelo de dados conceitual utilizado para a aplicação.

Figura 5.1 - Modelo ER do sistema Lumni



Fonte: Autor (2013).

5.3.4 Modelagem da aplicação

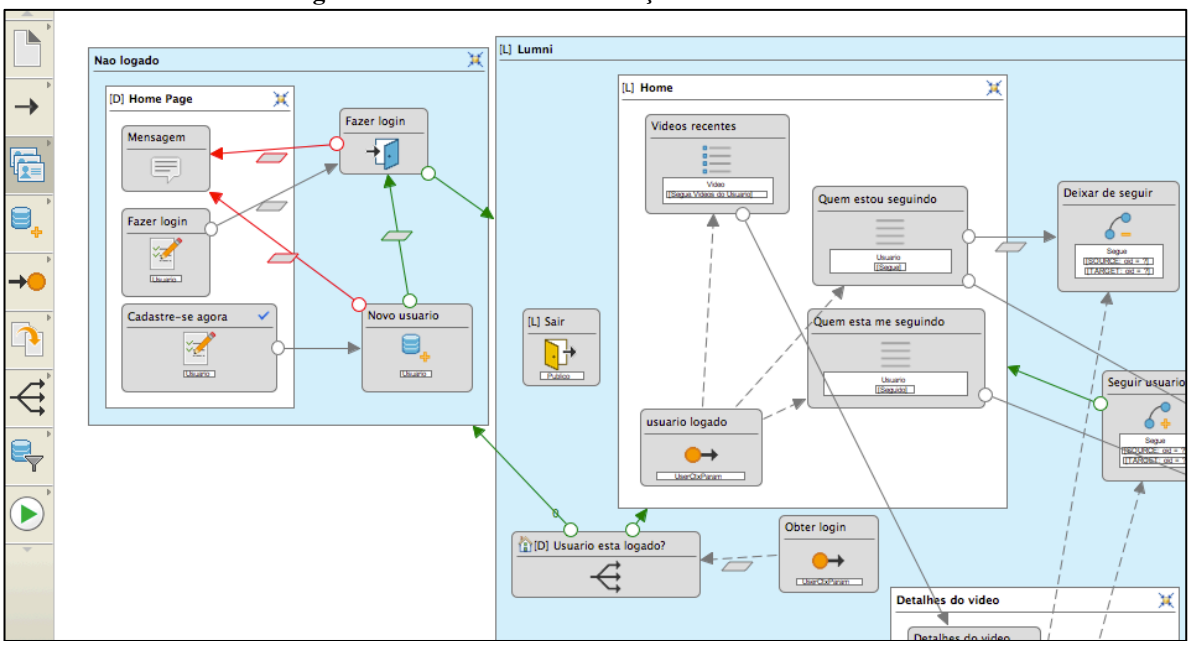
Com base nas natações e conceitos de IFML, apresentados no capitulo 5 deste trabalho, e a partir da analise dos requisitos demonstrados a partir dos casos de uso, a seguir será apresentada a modelagem desenvolvida e utilizada para obter a aplicação posteriormente.

Algumas das funções inicias da aplicação podem ser observadas na Figura 5.2. O objeto “Usuario está logado?” está sinalizado com a letra D, o que significa que esta é a *default page* e será a primeira ação a ser executada quando a aplicação rodar, e é o objeto responsável por identificar se o usuário está autenticado no sistema e então redirecionar para a Home em caso positivo ou página de login em caso negativo.

Além disso, observa-se que o objeto “Sair” está sinalizado com a letra L, o que significa que em qualquer tela dentro do *site view* Lumni esta função estará disponível.

Outro aspecto importante de ser observado, é que a aplicação foi organizada em duas *site views* distintas, sendo que uma chamada de “Lumni” abrange as *pages* para usuários autenticados no sistema, enquanto a outra chamada de “Nao logado” disponibiliza as opções de login e cadastro no sistema.

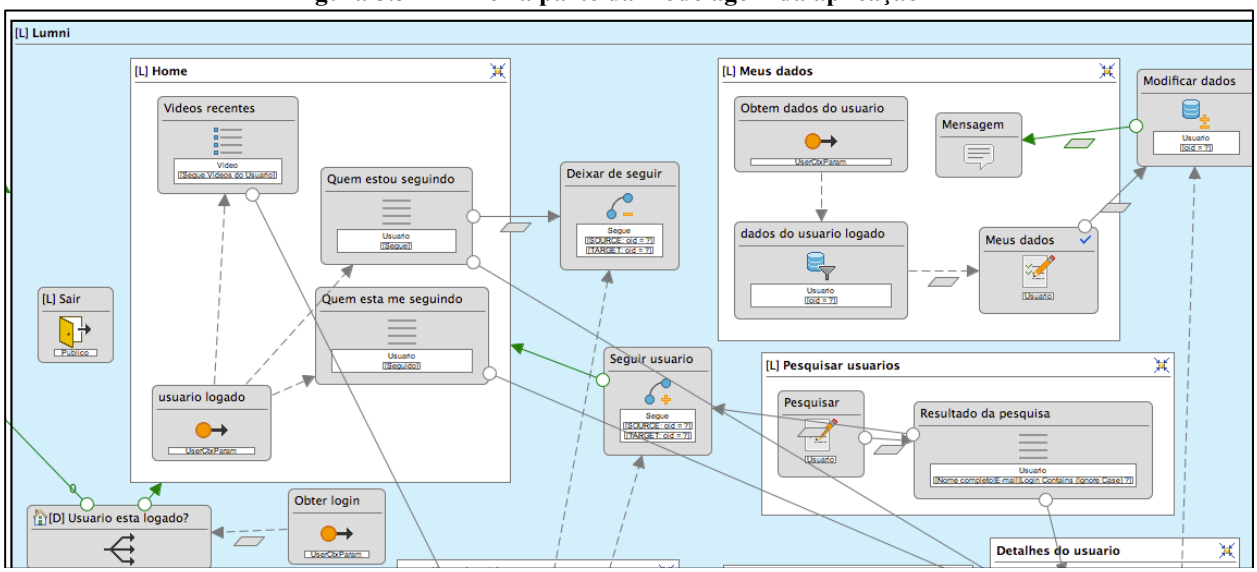
Figura 5.2 - Modelo com as funções iniciais do Lumni



Fonte: Autor (2013).

A Figura 5.3 abrange as *pages* “Home”, “Meus dados” e “Pesquisar usuários”. Além de demonstrar o conteúdo de cada página e algumas funções relacionadas.

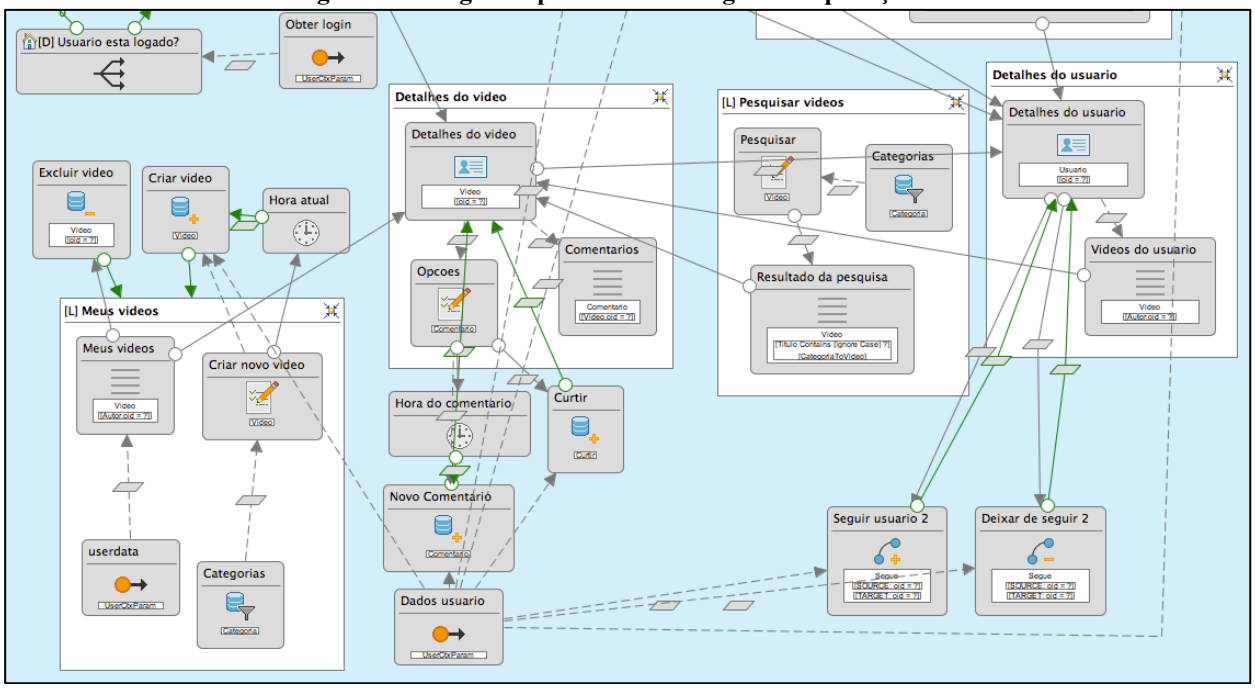
Figura 5.3 - Primeira parte da modelagem da aplicação



Fonte: Autor (2013).

A Figura 5.4 abrange as *pages* “Meus vídeos”, “Detalhes do vídeo”, “Pesquisar vídeos” e “Detalhes do usuário”. Além de demonstrar o conteúdo de cada página e algumas funções relacionadas.

Figura 5.4 - Segunda parte da modelagem da aplicação



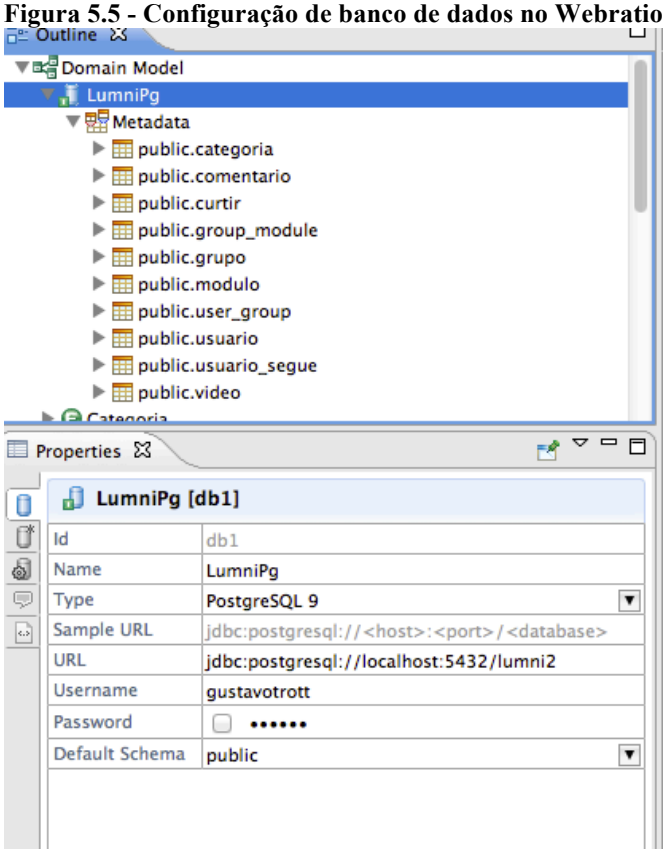
Fonte: Autor (2013).

5.4 CONSTRUÇÃO E UTILIZAÇÃO

A etapa de construção da aplicação seguiu a funcionalidade de geração de código a partir de modelos, fornecido pela ferramenta WebRatio. Neste sentido, as etapa de **implementar cada layout de página, função, forma e capacidade de navegação** foi implementada de forma automatizada de acordo com os modelos construídos na etapa de modelagem.

Referente ao banco de dados, uma versão do PostgreSQL 9.2.0, disponibilizado através do site oficial do PostgreSQL (2013), foi instalada para armazenar os dados referentes a aplicação Lumni. Após a instalação e criação de login e senha para acesso a base de dados, os mesmos foram configurados na ferramenta Webratio, conforme a Figura 5.5.

Figura 5.5 - Configuração de banco de dados no Webratio



Fonte: Autor (2013)

A criação de tabelas e relações é feita de maneira automatizada, uma vez que o modelo conceitual foi criado previamente. A ferramenta se responsabiliza pela geração do script SQL, sendo que a estrutura de tabelas geradas pode ser observado no **Apêndice D**.

As etapas de **testar o conteúdo, funções e navegação** foram realizadas, afim de encontrar erros na modelagem dos requisitos, para isto utilizou-se os fluxos de exceções, referentes a cada um dos casos de uso, contidos no **Apêndice A**.

Do ponto de vista da utilização do sistema, tratado neste capítulo, a tabela 5.3 demonstra uma relação entre os casos de uso desenvolvidos e as telas obtidas na versão final da aplicação.

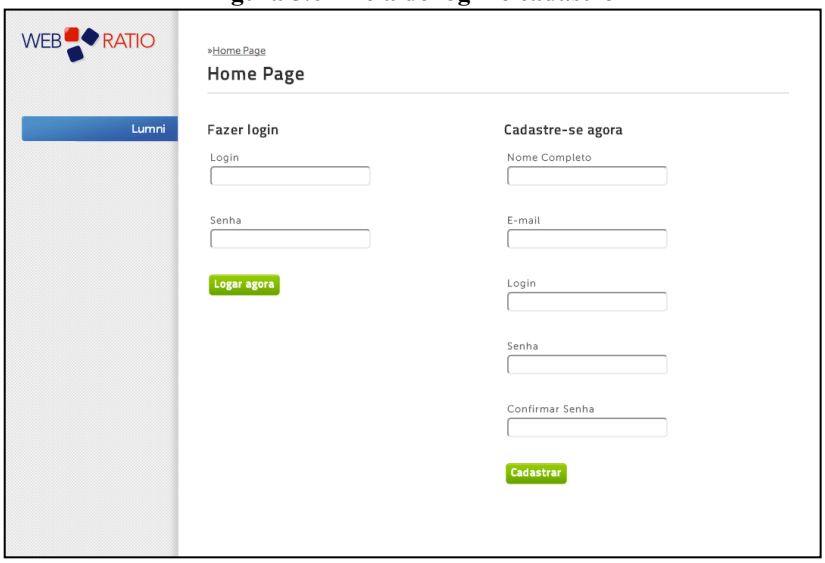
Tabela 5.3 - Relação entre telas do sistema e casos de uso.

Tela	**Casos de uso envolvidos**
Figura 5.6 – Tela de login e cadastro	Login no sistema Cadastro de usuário
Figura 5.7 – Tela inicial do usuário	Consulta de vídeos recentes Consulta de usuários seguindo Consulta de usuários seguidores Opção de deixar de seguir um usuário
Figura 5.8 – Tela de pesquisa de vídeos	Pesquisa de vídeos
Figura 5.9 – Tela de pesquisa de usuários	Pesquisa de usuários Opção de seguir um usuário

Tela	Casos de uso envolvidos
Figura 5.10 – Tela de vídeos do usuário	Consulta de vídeos de autoria do usuário Cadastro de vídeo Exclusão de vídeo
Figura 5.11 – Tela de alteração de dados do usuário	Alteração de dados do usuário
Figura 5.12 – Tela de detalhes de um vídeo	Consulta de informações do vídeo Cadastro de comentário ao vídeo Opção de curtir um vídeo
Figura 5.13 – Tela de detalhes de um usuário	Opção de seguir um usuário Opção de deixar de seguir um usuário

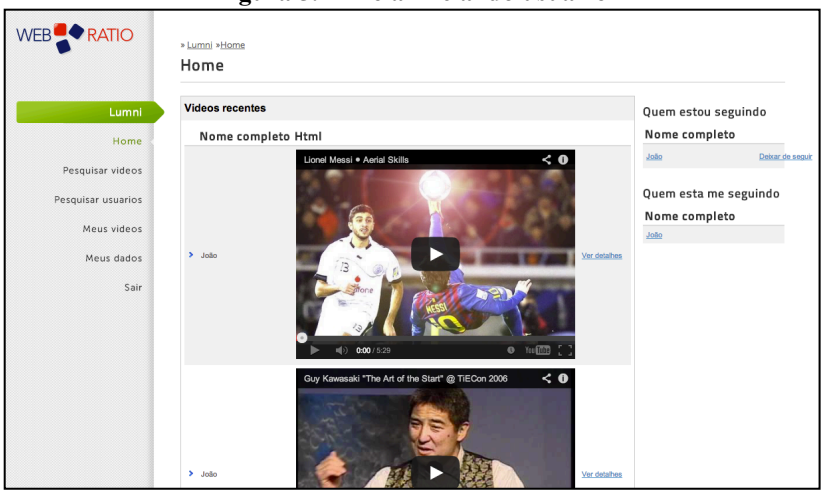
Fonte: Autor (2013)

Figura 5.6 - Tela de login e cadastro



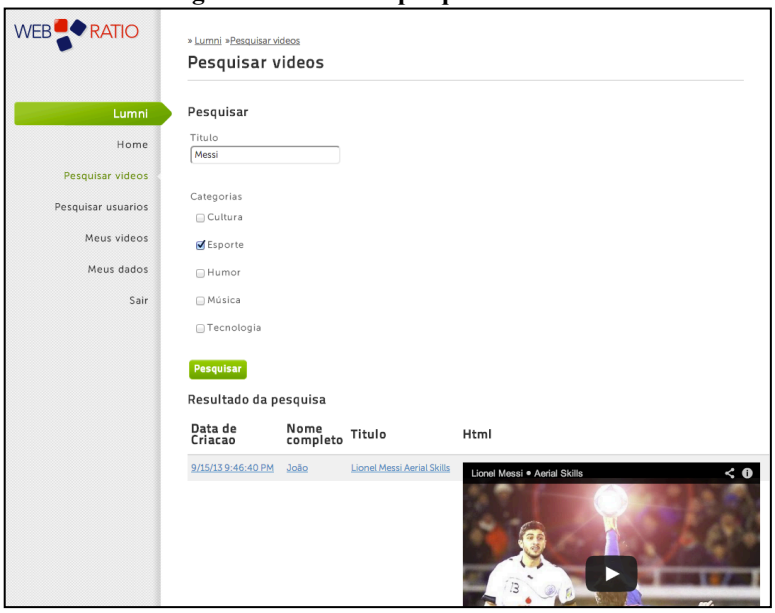
Fonte: Autor (2013)

Figura 5.7 - Tela inicial do usuário



Fonte: Autor (2013)

Figura 5.8 - Tela de pesquisa de vídeos

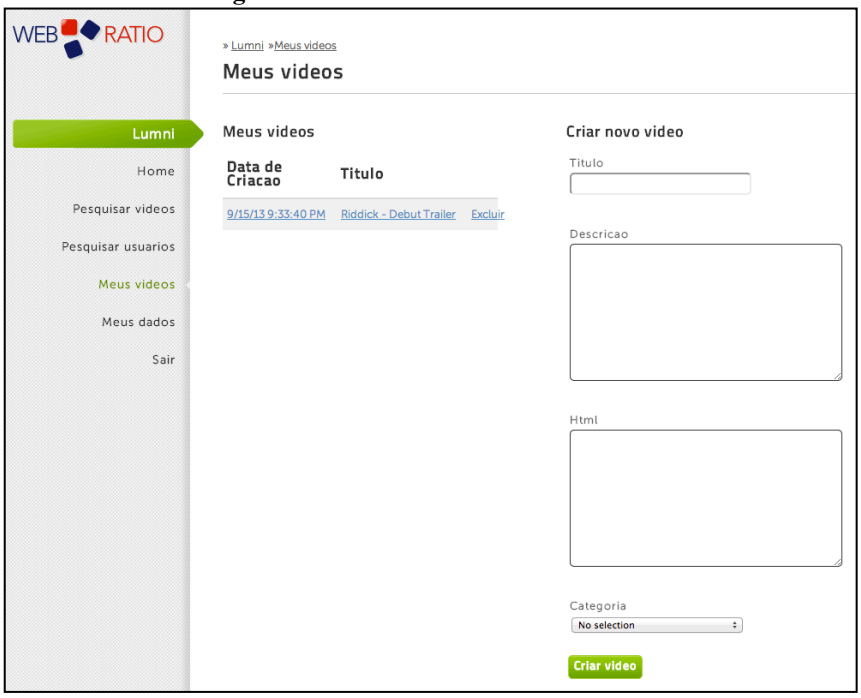


Fonte: Autor (2013)

Figura 5.9 - Tela de pesquisa de usuários

Fonte: Autor (2013)

Figura 5.10 - Tela de vídeos do usuário

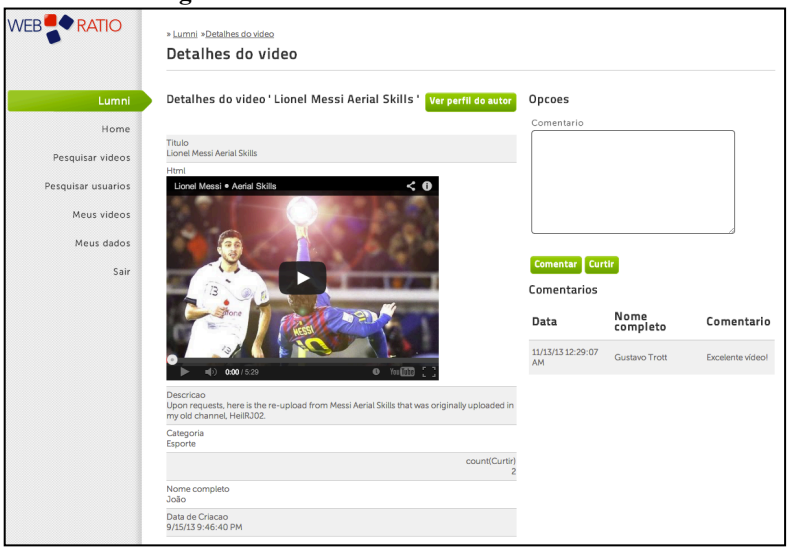


Fonte: Autor (2013)

Figura 5.11 - Tela de alteração de dados do usuário

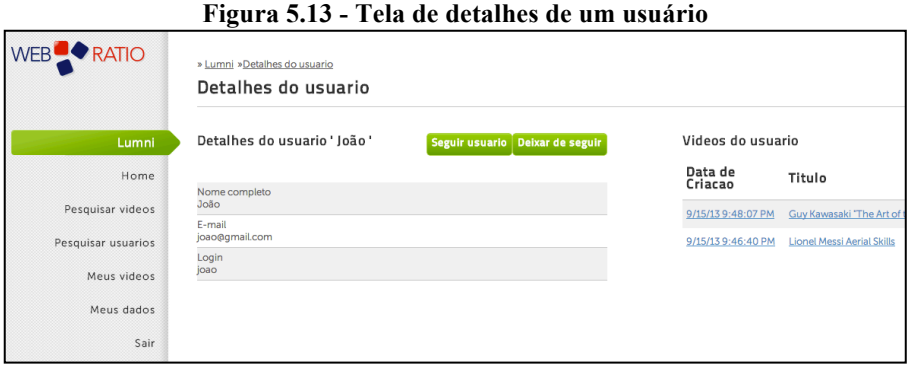
Fonte: Autor (2013)

Figura 5.12 - Tela de detalhes de um vídeo



Fonte: Autor (2013)

Figura 5.13 - Tela de detalhes de um usuário



Fonte: Autor (2013)

6 AVALIAÇÃO DA ESTRATÉGIA UTILIZADA

Para realizar o processo de desenvolvimento, algumas abordagens estudadas ao longo do trabalho foram seguidas. Sendo que seguiu-se um modelo de processo de desenvolvimento proposto por Pressman e Lowe (2009), enquanto a linguagem modelagem utilizada foi a IFML, que consolidou-se como sendo a linguagem de modelagem de iteração de fluxo padrão da OMG. Já para a geração do código fonte, utilizou-se o conceito de MDD que, conforme estudado, propõe um ganho de produtividade em relação ao desenvolvimento convencional (escrita de código manualmente).

Nas seções a seguir, será abordada uma avaliação quanto ao modelo de desenvolvimento escolhido para este trabalho, além de um comparativo de esforço, a fim de observar os ganhos de produtividade ao utilizar-se MDD para desenvolvimento.

6.1 AVALIAÇÃO DO MODELO DE DESENVOLVIMENTO

O processo de desenvolvimento proposto por Pressman e Lowe (2009) propõe etapas que devem ser seguidas durante as atividades de comunicação, planejamento, modelagem, construção e implementação. Neste trabalho, apenas algumas das etapas foram realizadas, uma vez que o foco do trabalho é avaliar a modelagem e desenvolvimento, porém o processo mostrou-se adequado e completo para o desenvolvimento de uma aplicação web real, uma vez que ele é baseado em experiências de projetos realizados e abrange etapas que irão buscar uma maior qualidade para a aplicação.

A avaliação aqui contida resultante da experimentação do processo em conjunto com aplicação da IFML e do Webratio. Assim, a utilização da linguagem IFML para modelar o sistema afirmou-se como uma ótima opção, sendo que a linguagem possuí notações que preveem a maioria das funcionalidades encontradas em uma aplicação web. Todavia, a linguagem é complexa para a criação de componentes não previstos e o tempo para a implementação de um módulo pode comprometer todo o ganho obtido com os módulos fornecidos de antemão.

A facilidade de relacionar formulários e tabelas com as entidades do modelo estrutural torna criação e consulta de registros um procedimento rápido e livre de erros. Além de possuir uma vasta variedade de validações que podem ser utilizar nos formulários de forma simples.

A ferramenta Webratio 7.2.0 torna mais fácil a utilização da linguagem IFML, uma vez que a interface simples e intuitiva acelera a compreensão dos conceitos da linguagem, mesmo para usuários iniciantes.

Os principais programas de banco de dados do mercado são suportados pela ferramenta, sendo que a escolhida o experimento foi o PostgreSQL 9.2.0, por ser uma aplicação grátis e de utilização habitual do autor. A criação de tabelas foi um processo fácil de ser realizado, porém a ferramenta demonstrou problemas quando o modelo conceitual sofreu alterações e foi necessário sincronizar com o aplicativo de banco de dados, sendo que por algumas vezes foi necessário excluir a base de dados existente manualmente e criar todas as tabelas novamente pelo Webratio.

A ferramenta Webratio fornece uma opção para encontrar erros na modelagem, o que ajuda a solucionar qualquer problema que possa ter ocorrido durante a execução da aplicação. Sendo que por muitas vezes a alteração do modelo de estruturas pode afetar na modelagem de hipertextos e esta função de encontrar erros pode ajudar a evitar problemas neste sentido.

De um modo geral, a modelagem em IFML mostrou-se eficaz para sistemas web, principalmente se comparado com uma modelagem utilizando-se de linguagens de modelagem generalistas. A partir de uma observação no modelo final, é possível identificar as funcionalidades do sistema e a relação entre as páginas e conteúdos.

O processo MDD fornecido pela ferramenta Webratio, gerou uma aplicação com código fonte em JSP e foi testado rodando em um servidor executando a aplicação Apache Tomcat 6.0, o qual foi fornecido e instalado automaticamente pela ferramenta. O gerador de código possui um algoritmo maduro, uma vez que a aplicação final gerada segue rigorosamente a modelagem estabelecida.

6.2 COMPARATIVO DE ESFORÇO

Seguindo o processo de desenvolvimento descrito no subcapitulo 3.2, levou-se 12 horas para o desenvolvimento do modelo e a geração automática do código da aplicação. Levando-se em conta a tabela de pontos de função obtida conforme o **Apêndice B**, que totaliza 119 pontos, a produtividade para o desenvolvimento da aplicação Lumni chegou a aproximadamente 10 pontos de função por hora. Cabe ressaltar que nas 12 horas utilizadas para o desenvolvimento da aplicação, não estão contabilizadas horas de estudo do Webratio e da IFML.

Para fim de comparação, convidou-se três empresas da grande porto alegre para realizar uma estimativa de horas necessárias para o desenvolvimento da mesma aplicação utilizada no experimento deste trabalho. O modelo do e-mail enviado está descrito no **Apêndice D**, enquanto a documentação da aplicação anexada no e-mail está contida no **Apêndice B**.

Das três empresas solicitadas, duas responderam a solicitação enviando estimativas em horas para os processo de codificação, ignorando outras etapas do projeto, como analise ou testes. A Tabela 6.1 demonstra os resultados obtidos.

Tabela 6.1 - Comparação de produtividade de empresas da região.

Empresa	**Total de horas**	**Pontos de função**	**Produtividade**
Empresa 1	128	119	0,92 pontos por hora
Empresa 2	264	119	0,45 pontos por hora
Autor	12	119	9,91 pontos por hora

Fonte: Autor (2013)

Observa-se que a produtividade utilizando-se o processo estabelecido neste trabalho é quase dez vezes superior ao processo de empresas da região. Isto está diretamente relacionado ao fato das empresas pesquisadas não utilizarem MDD em seus processos, mas sim uma codificação manual. Por outro lado, os sistemas fornecidos pelas empresas serão altamente customizados, enquanto a aplicação gerada neste trabalho segue uma linha padrão da ferramenta, por muitas vezes limitada.

CONCLUSÃO

Cada vez mais as empresas de desenvolvimento de software estão preocupadas em aumentar a produtividade da equipe durante o processo de concepção de um software, desenvolvendo mais aplicações em menos tempo, mas sem abrir mão da qualidade do produto resultante.

Neste contexto, essa monografia propôs o estudo de duas abordagens que são conhecidas por obter melhores resultados neste sentido, que são as linguagens de modelagem para domínio específico, chamadas de DSM e que procuram aumentar o nível de abstração dos modelos e possibilitam a automação de partes do desenvolvimento, e o desenvolvimento baseado em modelos, conhecido por MDD e que possibilita que os modelos deixem de ser apenas referencias e passem a gerar código para aplicação final. Então, a partir de estudos bibliográficos cumpriu-se os objetivos: **Apresentou conceitos relacionados ao MDD** e **Estudou a linguagem de modelagem visual WebML/IFML**.

Sabendo-se do aumento exponencial de demandas para sistemas web, este foi o domínio escolhido para estudo, sendo que dentre as linguagens citadas no trabalho, a escolha da IFML para o estudo mostrou-se acertada, uma vez que a linguagem é hoje reconhecida como padrão para o domínio de aplicações *front-end* pela OMG, além de apresentar uma proposta de notação original para expressar a navegação e composição de recursos de interface de hipertexto, a qual mostrou-se madura e capaz de modelar um sistema completo de forma simples e eficiente.

Além disso, a utilização da IFML em conjunto com a ferramenta Webratio proporciona um ambiente completo para modelagem e geração de código em Java/JSP 2.0 através dos conceitos de MDD. Assim, em consonância esse trabalho, conforme objetivos: **Definiu um processo de desenvolvimento aderente ao MDD e a WebML/IFML**; **Designou uma ferramenta de modelagem que ofereça as notações da WebML/IFML**.

Ainda, **modelou um sistema web real utilizando a linguagem WebML/IFML** e, assim promoveu uma **avaliação de todo o processo e cálculo de esforço** comprovaram a eficácia de um processo utilizando MDD, sendo que o tempo total do processo de desenvolvimento foi quase dez vezes menor que um processo utilizando-se de meios manuais para geração de código. Vale ainda ressaltar que a linguagem da aplicação final obtida através da geração automática de código não é de domínio do autor, o que comprova uma das premissas do MDD de que o autor necessita apenas do domínio do negócio e das notações da

linguagem de origem, sem necessariamente conhecer a linguagem de destino que será obtida após a transformação. Isto permitiu a conclusão de todos os objetivos propostos neste trabalho.

Como trabalhos futuros, pode-se sugerir um estudo aprofundado em modelos de derivação da IFML, uma vez que esta funcionalidade não foi mais explorada por não ter sido um requisito do experimento. Além disso, a criação de componentes e novos estilos são oportunidades para complementar esta pesquisa explorando aspectos que não puderam ser construídos neste momento, bem como, aumentar o número de empresas participantes do teste de esforço realizado.

Ainda, a geração de outros sistemas mais complexos ou de outro domínio de negócio podem trazer resultados que permitam avaliações complementares a este estudo.

REFERÊNCIAS BIBLIOGRÁFICAS

ATKINSON, Colin. KUHNE, Thomas. **Model-Driven Development: A Metamodeling Foundation.** IEEE Computer Society, 2003.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML.** Rio de Janeiro, RJ: Campus, 2002.

BIANCHINI, Sandro Lopes. **Avaliação de métodos de desenvolvimento de aplicações web.** USP, São Carlos, 2008.

BOOCH, Grady. RUMBAUGH, James. JACOBSON, Ivar. **Uml, Guia do Usuário.** Ed. Campus, 2005.

BRAMBILLA, Marco et al. **Designing Web Applications with WebML and WebRatio.** In book: G. Rossi, O. Pastor, D. Schwabe, L. Olsina (Eds.). Web Engineering: Modelling and Implementing Web Applications (Human-Computer Interaction Series). Springer, October 2007, ISBN: 978-1846289224. Disponível em: http://www.webml.org/webml/upload/ent5/1/Chapter%209%20-%20WebML.pdf, Acesso em: Acesso em 28 de Março de 2013.

CERI, Stefano et al. **Designing multi-role, collaborative Web sites with WebML: a Conference Management System case study**. IWWOST, 2001 Disponível em: <http://www.webml.org/webml/upload/ent5/2/IWWOST01.pdf > Acessado em 24/10/2013.

COCKBURN, Alistair. **Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software.** Ed. Bookman, 2005.

FOWLER, Martin. **UML essencial: um breve guia para a linguagem padrão de modelagem de objetos.** Trad. João Tortello. Porto Alegre: Bookman, 2005.

FRATERNALI, Piero. PAOLINI, Paolo. "**Model-Driven Development of Web Applications: the Autoweb System**", 2000 Disponível em: < http://www.webml.org/webml/upload/ent5/1/TOIS.pdf>. Acesso em Outubro de 2013.

GU, Alice. HENDERSON-SELLERS, Brian. LOWE, David. "**Web Modelling Languages: The Gap Between Requirements And Current Exemplars**", 2002 Disponível em: < http://ausweb.scu.edu.au/aw02/papers/refereed/lowe/paper.html>. Acesso em Outubro de 2013.

IFML: The standard Interaction Flow Modeling Language. Disponível em < http://www.ifml.org>. Acesso em Outubro de 2013.

KELLY, Steven. TOLVANEN, Juha-Pekka. **Domain-Specific Modeling, Enabling Full Code Generation.** IEEE Computer Society, 2008.

LOBO, Edson Junio Rodrigues. **Guia Prático de Engenharia de Software.** São Paulo: Digerati Books, 2009.

MELLOR, Stephen J. CLARK, Anthony N. TOYO, Takao Futagami. **Model-driven Development.** IEEE Computer Society, 2003.

OMG: Object Management Group. Disponível em <http://www.omg.org>. Acesso em Outubro de 2013.

ORACLE. **The Java EE 5 Tutorial**. 2010. Disponível em: <http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html> Acessado em 29/10/2013.

PostgreSQL. Disponível em <http://www.postgresql.org/>. Acesso em Julho de 2013.

PRESSMAN, Roger S. **Engenharia de Software: Uma abordagem Profissional, Sétima Edição.** The McGraw-Hill Companies, 2011.

PRESSMAN, Roger S. LOWE, David. **Web Engineering: A practitioner's Approach**. McGraw-Hill, 2009.

SCHIMIDT, Douglas C. **Model-Driven Engineering.** IEE Computer, 2006 Disponível em: < http://www.cs.wustl.edu/~schmidt/GEI.pdf> Acessado em 24/03/2013.

SELIC, Bran. **The Pragmatics of Model-Driven Development.** IEEE Computer Society, 2003.

SILVEIRA, Sandro Madruga. **Criação de um framework modular e expansível para geração de código fonte.** Universidade Feevale, Novo Hamburgo, 2012.

SPARX SYSTEMS. **MDA Overview**, 2007 Disponível em: <http://www.sparxsystems.com>. Acesso em 10 de Março de 2013.

SUH, Woojong. **Web Engineering, Principles and Techniques.** IGI Global, 2004.

W3C: **XML: Extensible Markup Language.** 2013a. Disponível em: <http://www.w3.org/XML/>. Acessado em 10/11/2013.

W3C: **CSS: Cascading Style Sheets.** 2013b. Disponível em: <http://www.w3.org/Style/CSS/>. Acessado em 10/11/2013.

W3C: **JAVASCRIPT WEB APIS.** 2013c. Disponível em: <http://www.w3.org/standards/webdesign/script>. Acessado em 10/11/2013.

WATSON, Andrew. **Visual Modelling: past, present and future.** 2008. Disponível em: < http://www.uml.org/Visual_Modeling.pdf > Acessado em 24/03/2013.

WEBML. Disponível em: <http://www.webml.org>. Acesso em 10 de Março de 2013.

WEBML. **Summary of WebML hypertext elements**. Politecnico di Milano, 2003a. Disponível em: <http://webml.org/webml/upload/ent17/1/webml_elements.pdf> Acessado em 24/10/2013.

WEBML. **Redundant data modeling: Derivation**. Politecnico di Milano, 2003b. Disponível em: <http://www.webml.org/webml/page102.do?dau43.oid=13> Acessado em 15/10/2013.

WEBRATIO. Disponível em: <http://www.webratio.org>. 2013a. Acesso em 10 de Março de 2013.

WEBRATIO. **The differences between the WebML and IFML notation**. 2013b, Tutorial fornecido pela ferramenta WebRatio 7.2. Disponível em http://www.webratio.com/. Acesso em 22/08/2013.

APÊNDICE A – DOCUMENTAÇÃO LUMNI

Introdução

Este documento contempla a documentação de um sistema fictício que será utilizado como base para um caso de estudo de um trabalho de conclusão de curso de sistemas de informação pela universidade Feevale. Além de ser utilizado como base para obtenção de orçamentos em horas visando medir a produtividade da tecnologia estudada no trabalho em relação a empresas de desenvolvimento de software da região.

É importante ressaltar que o nome das empresas que fornecerem orçamentos não serão divulgados no trabalho. Além disso, são solicitadas apenas as horas destinadas a **desenvolvimento (programação e banco de dados)**, suprimindo outras etapas do processo, tal como: fase destinada a análise, ou testes, sendo assim não é necessário o valor financeiro do projeto e, sim a quantidade de horas de esforço destinadas ao desenvolvimento.

No que refere-se a tecnologia empregada, pede-se orçamento, para um sistema Web, utilizando-se de Java e banco de dados PostgreSQL.

O trabalho

O objetivo do trabalho é estudar os conceitos oriundos das técnicas de MDD associando a linguagem de modelagem visual WebML. Ao final, estabelecer um processo e desenvolver uma aplicação web utilizando como base códigos gerados a partir do processo estabelecido.

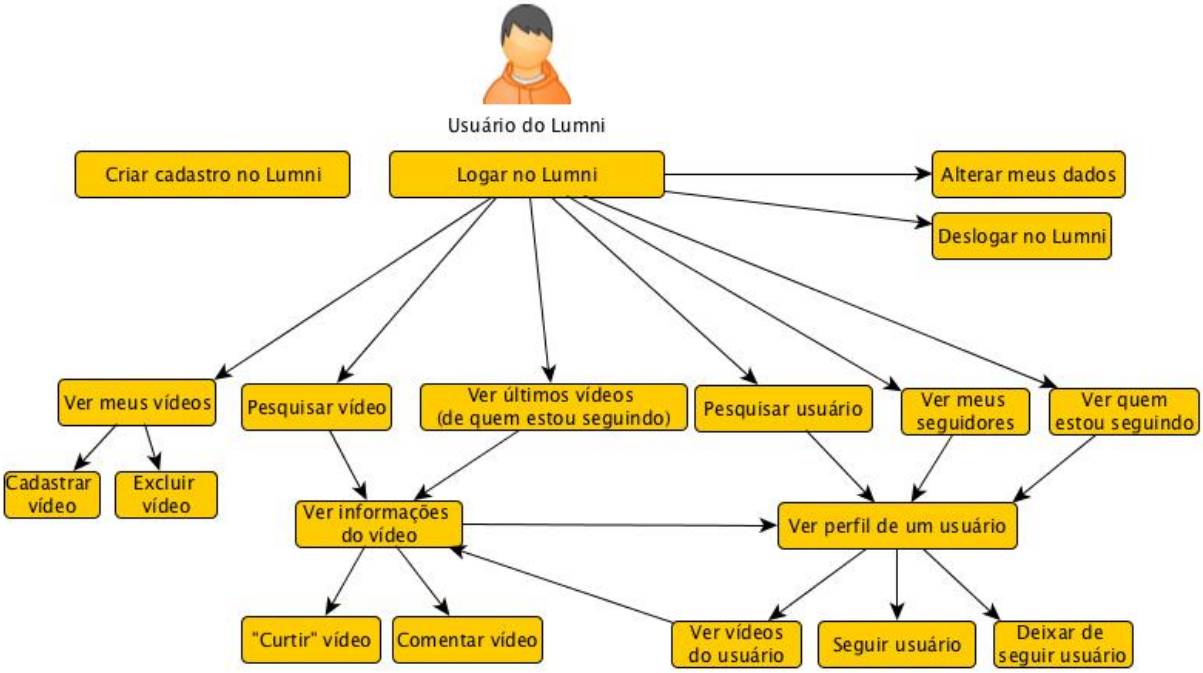
Lumni

O Lumni consiste basicamente em uma aplicação web que permita aos usuários postarem vídeos que serão compartilhados com outros usuários da rede com o objetivo principal de troca de conhecimentos.

Para acessar o sistema, o usuário deve primeiramente criar uma conta através do fornecimento de alguns dados e após esse procedimento o usuário estará apto a postar vídeos, procurar vídeos, seguir outros usuários, curtir e comentar vídeos, além de outras opções que estarão descritas detalhadamente neste documento.

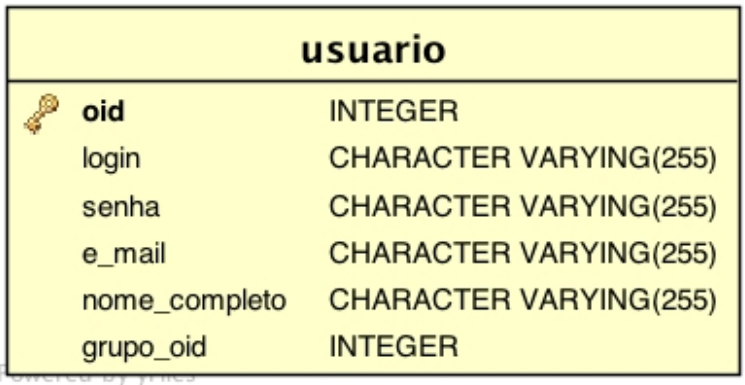
Os vídeos postados pelos usuários serão apenas "links" / incorporados de outros sites como youtube e vimeo, sendo assim, não serão gerenciados e nem hospedados pela aplicação.

Na figura abaixo, pode-se observar as principais funções do sistema mapeadas de forma encadeada para uma melhor visualização do seu funcionamento.



Nome do caso de uso: 01- Autenticação no sistema
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá autenticar-se no sistema e ter acesso a parte restrita a usuários autenticados.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário não pode estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Usuário ou senha inválidos.

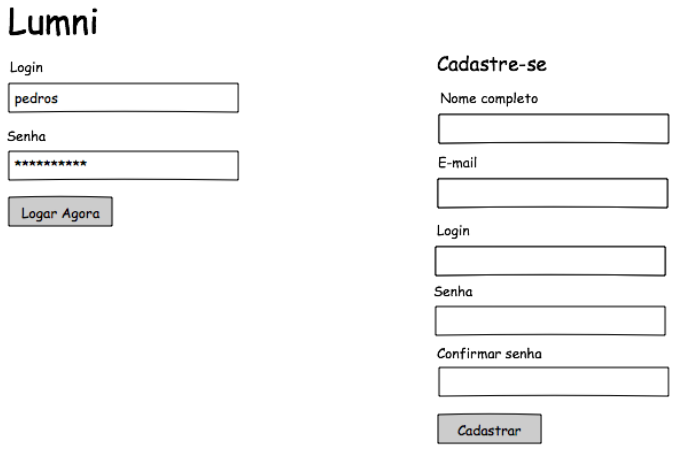
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Login e senha são de preenchimento obrigatório.
RN02	Caso o login informado não exista na base de dados, retornar a MSG01.
RN03	Caso a senha informada não coincida com a senha cadastrada no sistema para o login, retornar MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com os campos vazios.
RI02	O sistema apresenta a tela com os campos preenchidos.

Interface:



Fluxos principais:

Logar no sistema:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche os campos de Login e Senha.	
Pressiona o botão: Logar agora.	Valida ação conforme RN01, RN02 e RN03.
	Registra o usuário como autenticado no sistema.

Fluxos exceção:

1-Login e senha são de preenchimento obrigatório:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche o campo de Login.	
Pressiona o botão: Logar agora.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02.

2-Login deve existir na base de dados:

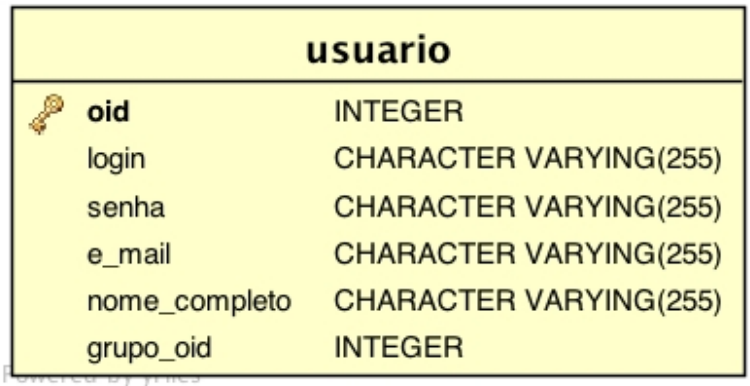
Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche o campo de Login e senha, porém com um login inválido.	
Pressiona o botão: Logar agora.	Valida ação conforme RN02.
	Apresenta tela conforme: RI02.

3-Senha deve coincidir com a senha cadastrada no sistema anteriormente:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche o campo de Login e senha, porém com uma senha inválida.	
Pressiona o botão: Logar agora.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02.

Nome do caso de uso: 02-Cadastro de usuário
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá criar o seu cadastro para utilizar a parte interna do sistema.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário não pode estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao cadastrar novo usuário.

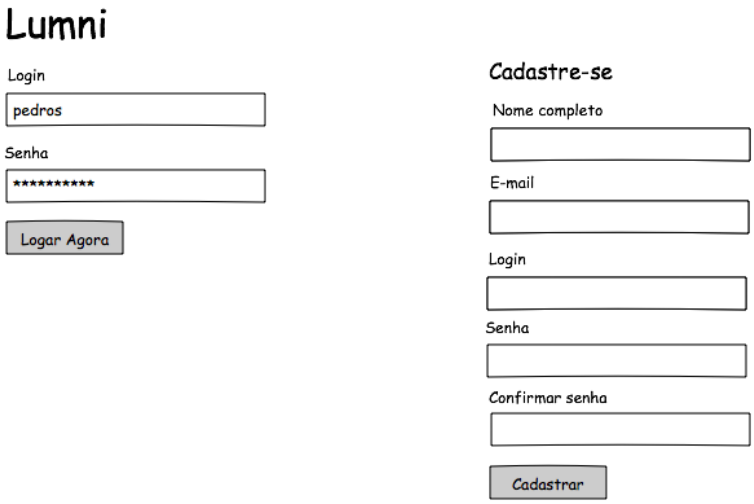
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Não permitir a gravação caso algum campo não esteja preenchido, sinalizando erro nos campos em branco.
RN02	Não permitir a gravação caso já exista algum login idêntico cadastrado no sistema, sinalizando erro no campo.
RN03	Não permitir a gravação caso confirmação da senha não coincida com a senha, sinalizando erro no campo.
RN04	Não permitir a gravação caso formato do e-mail for inválido, sinalizando erro no campo.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com os campos vazios, prontos para a inserção de novos registros.
RI02	O sistema apresenta a tela com os campos preenchidos.

Interface:



Fluxos principais:

Cadastrar usuário no sistema:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche os campos do formulário os com os dados do usuário.	
Pressiona o botão: Cadastrar.	Valida ação conforme RN01, RN02, RN03, RN04, RN05 e RN06.
	Criar novo usuário na base de dados e registra o usuário como autenticado no sistema.

Fluxos exceção:

1-Todos os campos são de preenchimento obrigatório:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche os campos do formulário os com os dados do usuário, porém com algum campo em branco.	
Pressiona o botão: Cadastrar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02.

2-Login deve ser único:

Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche os campos do formulário os com os dados do usuário, porém o login informado já existe no sistema.	
Pressiona o botão: Cadastrar.	Valida ação conforme RN02.
	Apresenta tela conforme: RI02.

3-Todos os campos são de preenchimento obrigatório:

Ator	Sistema

Acessa o sistema.	Apresenta tela conforme: RI01.
Preenche os campos do formulário os com os dados do usuário, porém a confirmação de senha informada é diferente da senha informada.	
Pressiona o botão: Cadastrar.	Valida ação conforme RN03.
	Apresenta tela conforme: RI02.

4-Formato do e-mail deve ser válido:

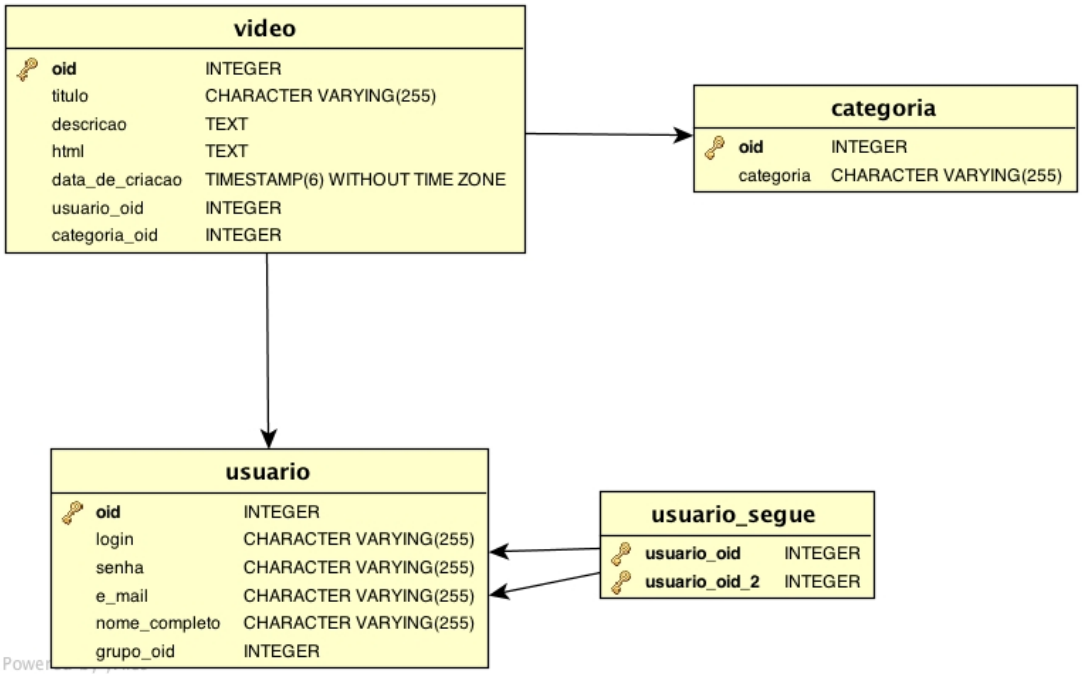
Ator	Sistema
Acessa o sistema.	Apresenta tela conforme: RI01
Preenche os campos do formulário os com os dados do usuário, porém o e-mail parece ser inválido.	
Pressiona o botão: Cadastrar.	Valida ação conforme RN04.
	Apresenta tela conforme: RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome Completo	Informe o nome completo.	Input text	Varchar	150	Sim
E-Mail	Informe o e-mail.	Input text	Varchar	60	Sim
Login	Informe o login.	Input text	Varchar	16	Sim
Senha	Informe a senha.	Input password	Varchar	32	Sim
Confirmar senha	Repita a senha.	Input password	Varchar	32	Sim

Nome do caso de uso: 03- Consulta de vídeos recentes
Prioridade: Média.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar os últimos vídeos cadastrados por usuários sinalizados por ele na opção de seguir.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Nenhum vídeo para ser exibido.

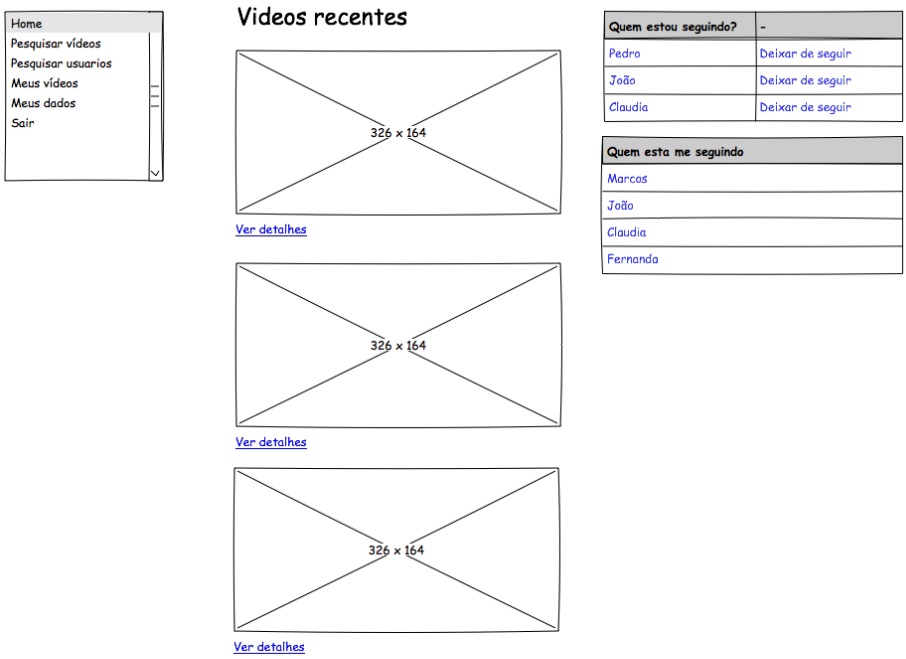
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	O sistema irá apresentar os 10 vídeos com maior data de cadastro.
RN02	Só serão exibidos os vídeos que o usuário criador está sinalizado na opção de seguir do usuário utilizador.
RN03	Se não houver vídeos para exibir, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com os campos Nome do autor e Html do vídeo.
RI02	O sistema deve apresentar o html do vídeo de forma que o usuário utilizador possa visualizar o vídeo a partir desse relatório.
RI03	O sistema deve apresentar um link que remete ao caso de uso “Consulta de informações do vídeo”.

Interface:



Fluxos principais:

Visualizar relatório de vídeos recentes:

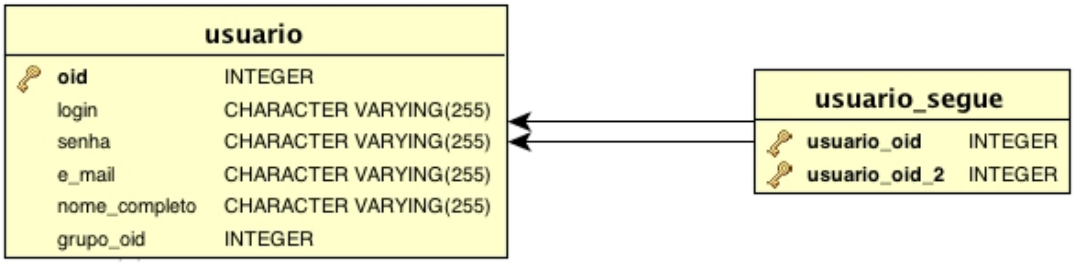
Ator	Sistema
Acessa o menu Home do sistema.	Apresenta tela conforme: RI01, RI02 e RI03.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome do autor	Informa o nome completo do usuário autor.	Label	Varchar	150	Sim
Html	Informa link incorporado do vídeo.	Label	Text / Html		Sim
Ver detalhes	Informa um link com acesso a detalhes do vídeo.	Link			Sim

Nome do caso de uso: 04- Consulta de usuários seguindo
Prioridade: Baixa.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar todos os usuários do sistema sinalizados por ele na opção de seguir.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Você não está seguindo nenhum usuário.

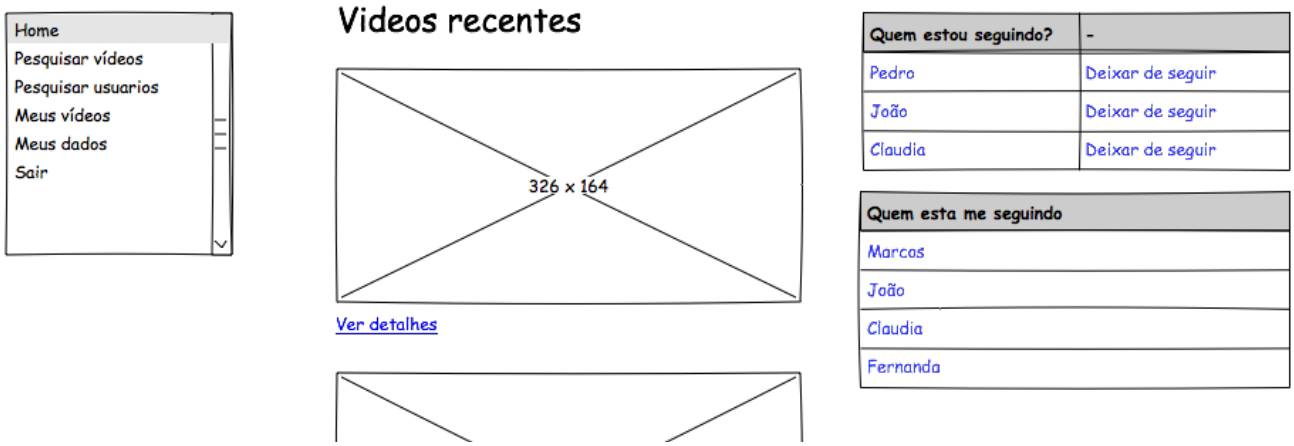
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	A lista deve ser ordenada pelo nome do usuário em ordem crescente.
RN02	Se não houver usuários para exibir, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com o campo Nome do usuário.
RI02	Ao clicar sobre o campo Nome do Usuário, o sistema será remetido ao caso de uso “Consulta de informações do usuário”.
RI03	O sistema deve apresentar um link que remete ao caso de uso “Opção de deixar de seguir um usuário”.

Interface:



Fluxos principais:

Visualizar relatório de usuários seguindo:

Ator	Sistema
Acessa o menu Home do sistema.	Apresenta tela conforme: RI01, RI02 e RI03.

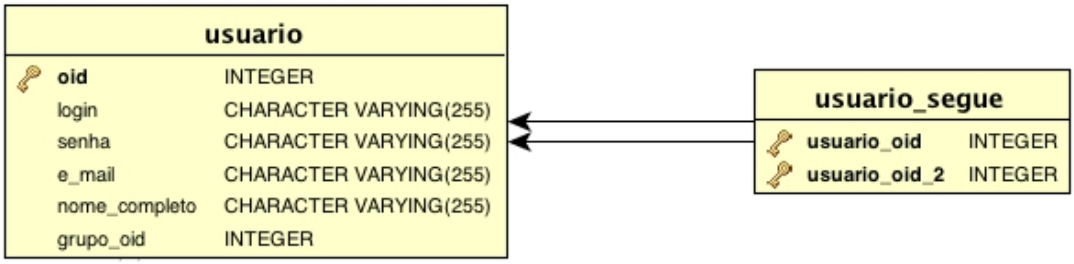
Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome do	Informa o nome	Label / Link	Varchar	150	Sim

usuário	completo do usuário.				
Deixar de seguir	Informa um link com a opção para deixar de seguir o usuário.	Link			Sim

Nome do caso de uso: 05- Consulta de usuários seguidores
Prioridade: Baixa.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar todos os usuários do sistema pelos quais ele foi sinalizado com a opção de seguir.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Nenhum usuário está seguindo você.

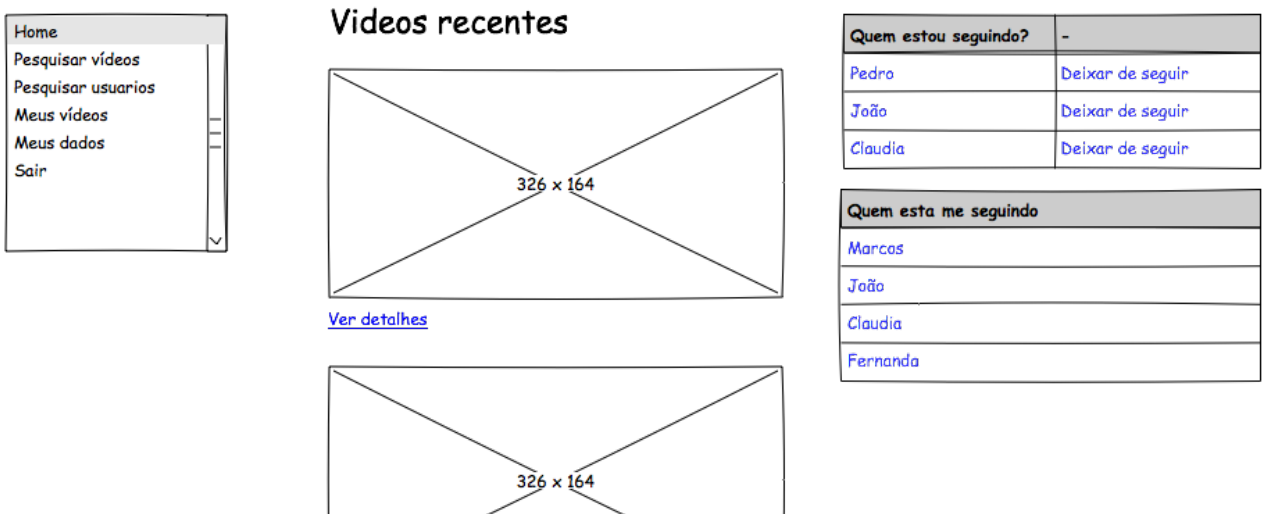
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	A lista deve ser ordenada pelo nome do usuário em ordem crescente.
RN02	Se não houver usuários para exibir, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com o campo Nome do usuário.
RI02	Ao clicar sobre o campo Nome do Usuário, o sistema será remetido ao caso de uso “Consulta de informações do usuário”.

Interface:



Fluxos principais:

Visualizar relatório de usuários seguidores:

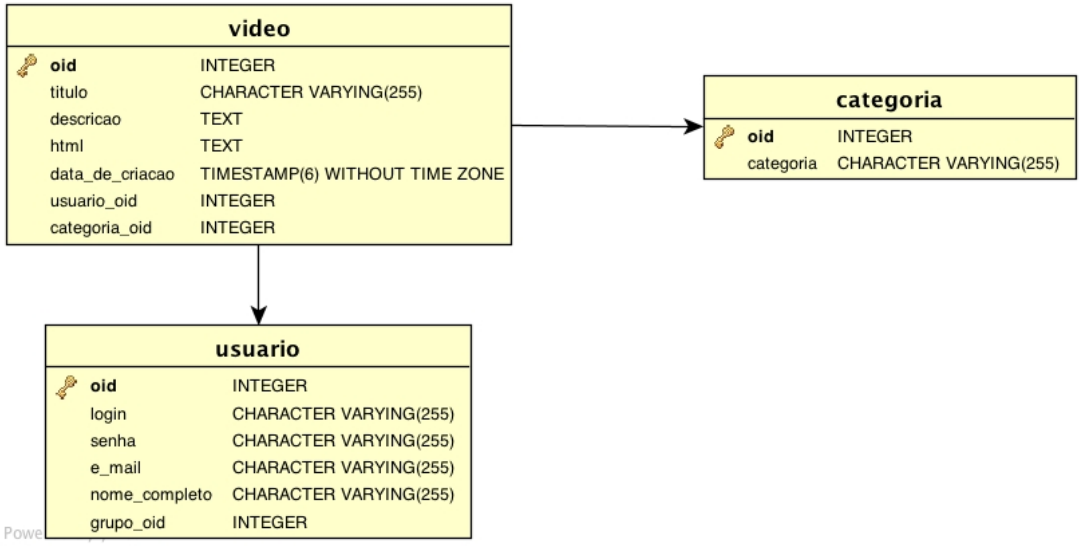
Ator	Sistema
Acessa o menu Home do sistema.	Apresenta tela conforme: RI01 e RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome do usuário	Informa o nome completo do usuário.	Label / Link	Varchar	150	Sim

Nome do caso de uso: 06- Pesquisa de vídeos
Prioridade: Baixa.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar vídeos cadastrados no sistema a partir de uma busca pelo nome ou parte dele e podendo opcionalmente escolher uma ou mais categorias que irão ser utilizadas como critério ao buscar apenas vídeos pertencentes a alguma delas.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Nenhum resultado para sua pesquisa.

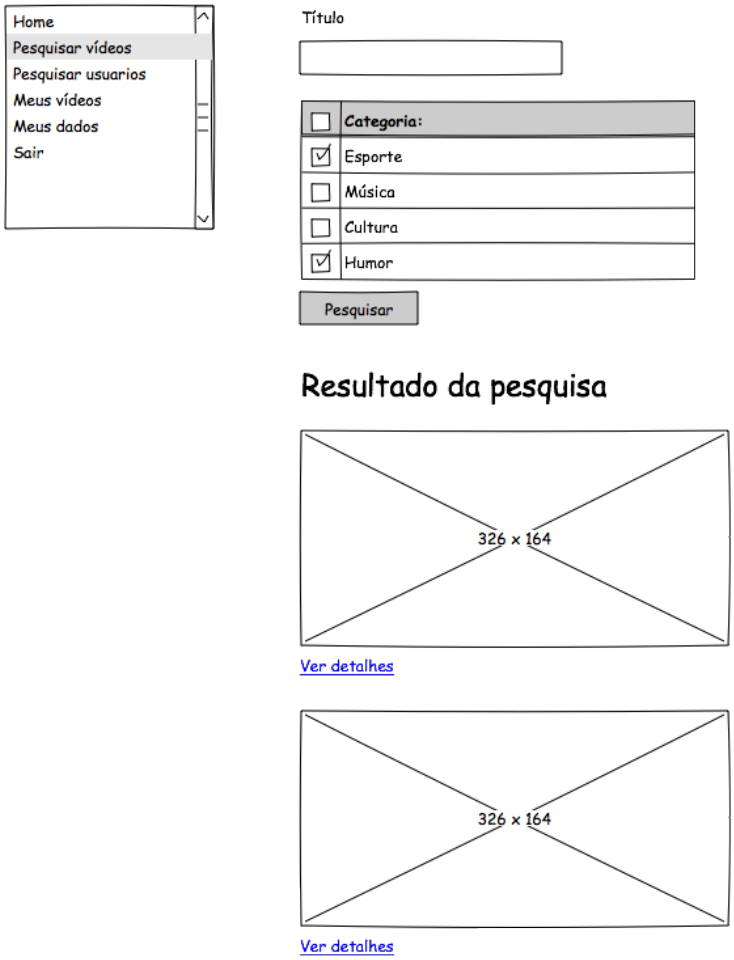
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	O usuário deverá obrigatoriamente informar um termo no campo de pesquisa “Titulo”.
RN02	O sistema apresentará vídeos que o título contenha o termo informado.
RN03	O campo Categorias é de preenchimento opcional, podendo o usuário selecionar uma ou mais.
RN03	Se não houver vídeos para os filtros selecionados, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com os campos Título e Categorias vazios.
RI02	O sistema apresenta a tela com os campos Título e Categorias preenchidos.
RI03	O sistema apresenta o relatório com os campos Nome do autor e Html do vídeo.
RI04	O sistema deve apresentar o html do vídeo de forma que o usuário utilizador possa visualizar o vídeo a partir desse relatório.
RI05	O sistema deve apresentar um link que remete ao caso de uso “Consulta de informações do vídeo”.

Interface:



Fluxos principais:

Pesquisar vídeo:

Ator	Sistema
Acessa o menu Pesquisar vídeos do sistema.	Apresenta tela conforme: RI01.
Preenche os campos de Título e Categoria.	
Pressiona o botão: Pesquisar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02, RI03, RI04 e RI05.

Fluxos exceção:

1-Campo título é de preenchimento obrigatório:

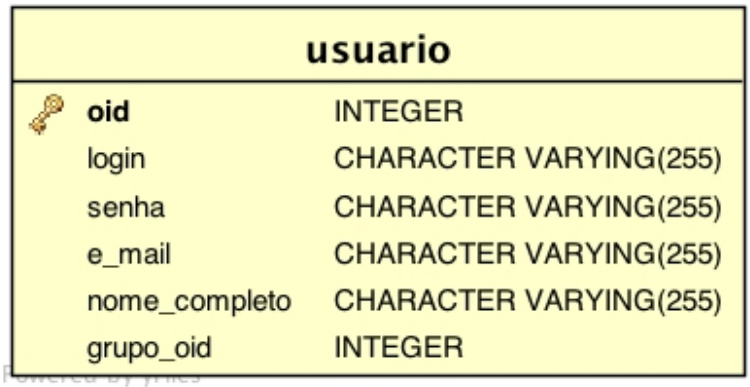
Ator	Sistema
Acessa o menu Pesquisar vídeos do sistema.	Apresenta tela conforme: RI01.
Preenche o campo Categoria.	
Pressiona o botão: Pesquisar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Titulo	Informe o Titulo a ser pesquisado.	Label	Varchar	150	Sim
Categoria	Informe a(s) categoria(s) que desejas pesquisar.	Select	Varchar / Integer	150	Não
Data de Criacao	Informa a data de cadastro do vídeo.	Label	Date		Sim
Nome do autor	Informa o nome completo do usuário autor.	Label	Varchar	150	Sim
Titulo	Informa o título do vídeo.	Label	Varchar	150	Sim
Html	Informa link incorporado do vídeo.	Label	Text / Html		Sim
Ver detalhes	Informa um link com acesso a detalhes do vídeo.	Link	Varchar		Sim

Nome do caso de uso: 07- Pesquisa de usuários
Prioridade: Baixa.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar outros usuários cadastrados no sistema a partir de uma busca pelo nome, login ou e-mail.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Nenhum resultado para sua pesquisa.

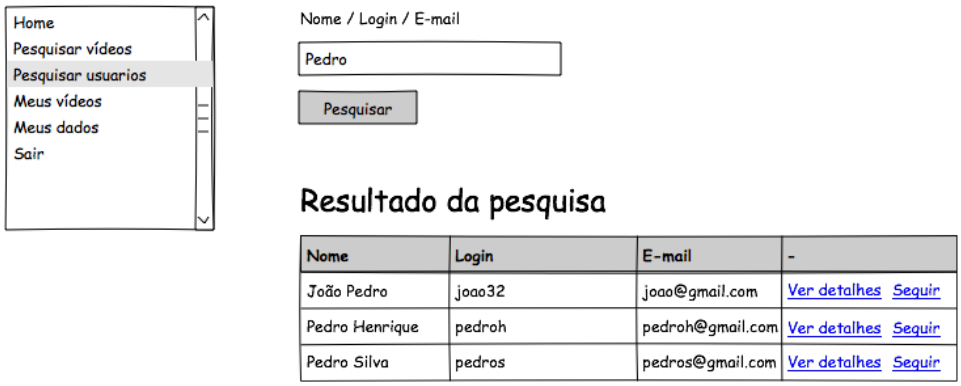
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	O usuário deverá obrigatoriamente informar um termo no campo de pesquisa "Nome/Login/E-mail".
RN02	O sistema apresentará usuários que o título nome contenha o termo informado ou o login contenha o termo informado ou o e-mail contenha o termo informado.
RN03	Se não houver usuários para o filtro selecionado, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com o campo Nome / Login / E-mail vazio.
RI02	O sistema apresenta a tela com o campo Nome / Login / E-mail preenchido.
RI03	O sistema apresenta o relatório com os campos Nome do autor e Html do vídeo.
RI04	O sistema deve apresentar o html do vídeo de forma que o usuário utilizador possa visualizar o vídeo a partir desse relatório.
RI05	O sistema deve apresentar um link que remete ao caso de uso “Consulta de informações do vídeo”.
RI06	O sistema deve apresentar um link que remete ao caso de uso “Opção de deixar de seguir um usuário”.

Interface:



Fluxos principais:

Pesquisar usuário:

Ator	Sistema
Acessa o menu Pesquisar usuários.	Apresenta tela conforme: RI01
Preenche o campo Nome / Login / E-mail.	
Pressiona o botão: Pesquisar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02, RI03, RI04,RI05 e RI06.

Fluxos exceção:

1-Campo Nome/Login/E-mail é de preenchimento obrigatório:

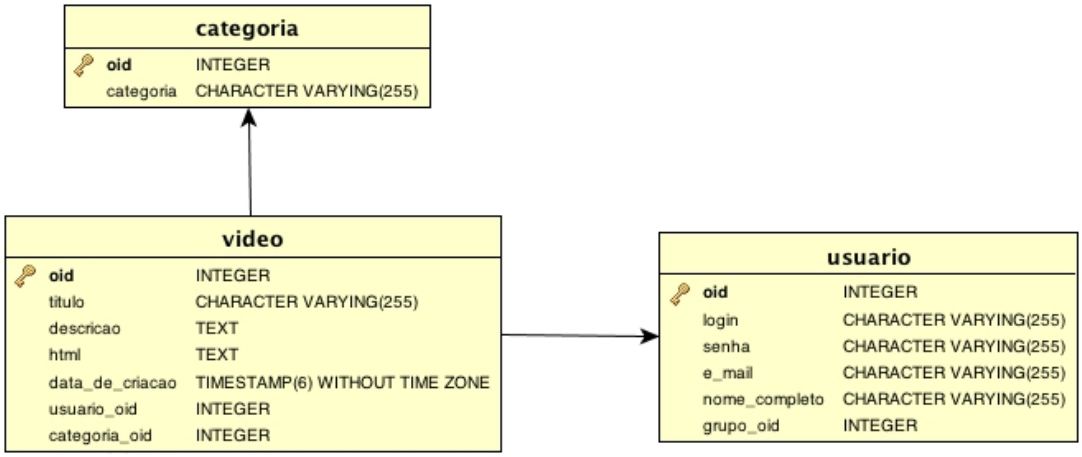
Ator	Sistema
Acessa o menu Pesquisar usuários.	Apresenta tela conforme: RI01.
Deixa o campo Nome/Login/E-mail em branco.	
Pressiona o botão: Pesquisar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI01.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome / Login / E-mail	Informe o Nome ou Login ou E-mail a ser pesquisado.	Label	Varchar	150	Sim
Nome completo	Informa o nome completo do usuário autor.	Label	Varchar	150	Sim
E-mail	Informa o e-mail do usuário.	Label	Varchar	150	Sim
Login	Informa o login do usuário.	Label	Varchar	150	Sim
Ver detalhes	Informa um link com a opção para ver detalhes do usuário.	Link			Sim
Seguir	Informa um link com a opção para seguir o usuário.	Link			Sim

Nome do caso de uso: 08- Consulta de vídeos de autoria do usuário
Prioridade: Baixa.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar os vídeos cadastrados por ele até o momento.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Nenhum vídeo cadastrado.

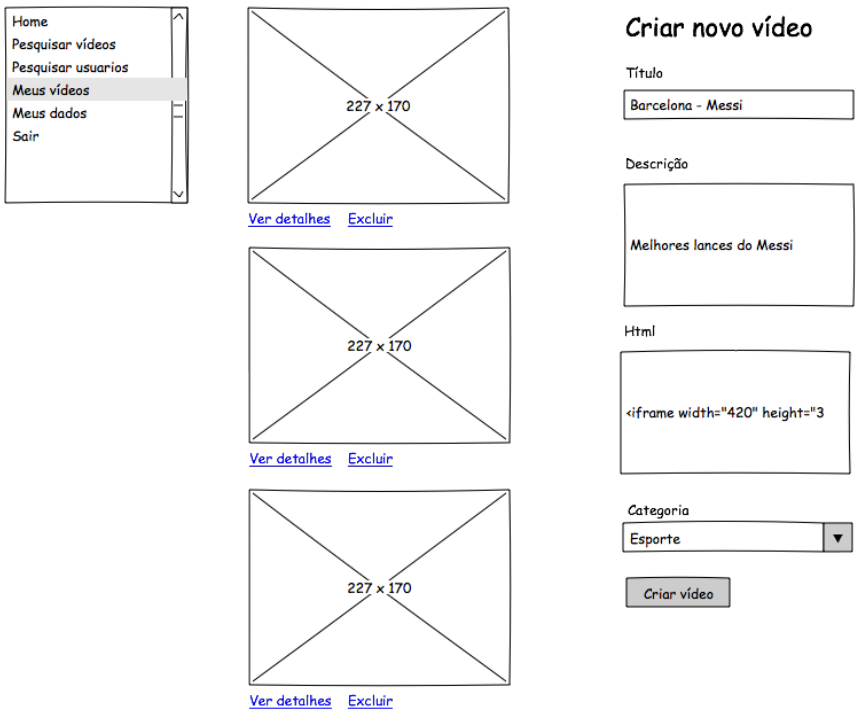
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Só serão exibidos vídeos cadastrados pelo utilizador do sistema.
RN02	Se não houver vídeos cadastrados pelo usuário, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com os campos Data de criação e Titulo.
RI02	O sistema deve apresentar um link que remete ao caso de uso “Consulta de informações do vídeo”.
RI03	O sistema deve apresentar um link que remete ao caso de uso “Exclusão de vídeo”.

Interface:



Fluxos principais:

Visualiza relatórios com vídeos de autoria do utilizador:

Ator	Sistema
Acessa o menu Meus vídeos.	Apresenta tela conforme: RI01, RI02 e RI03.

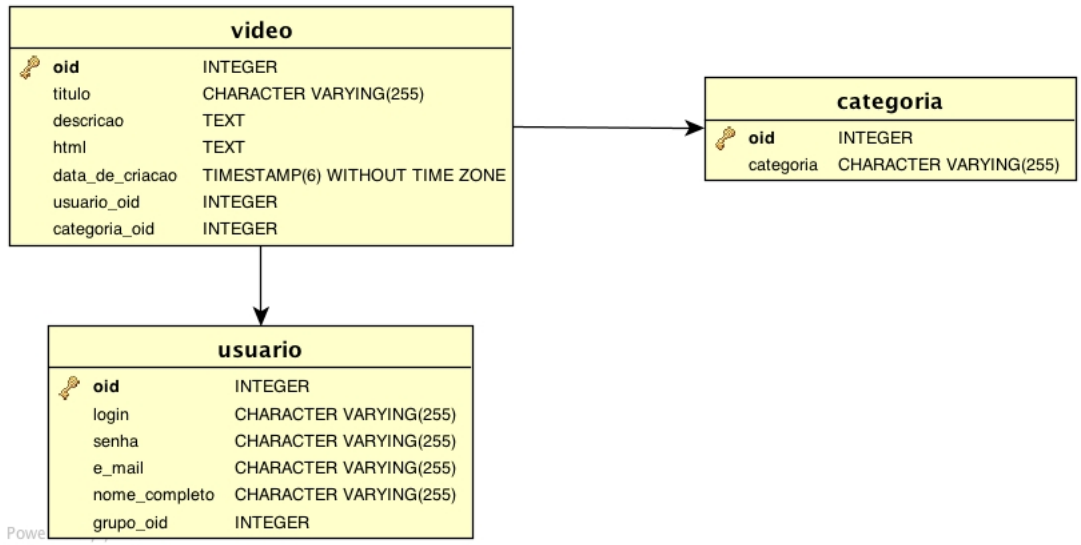
Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Data de Criação	Informe a data de cadastro do vídeo.	Label	Date		Sim
Titulo	Informa o título do vídeo.	Label	Varchar	150	Sim
Ver detalhes	Informa um link com a opção para ver detalhes do vídeo.	Link			Sim

Excluir	Informa um link com a opção para excluir o vídeo.	Link			Sim

Nome do caso de uso: 09- Cadastro de vídeo
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá cadastrar um vídeo, informando os dados e o "link" incorporados obtido através do youtube ou vimeo.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao cadastrar vídeo.

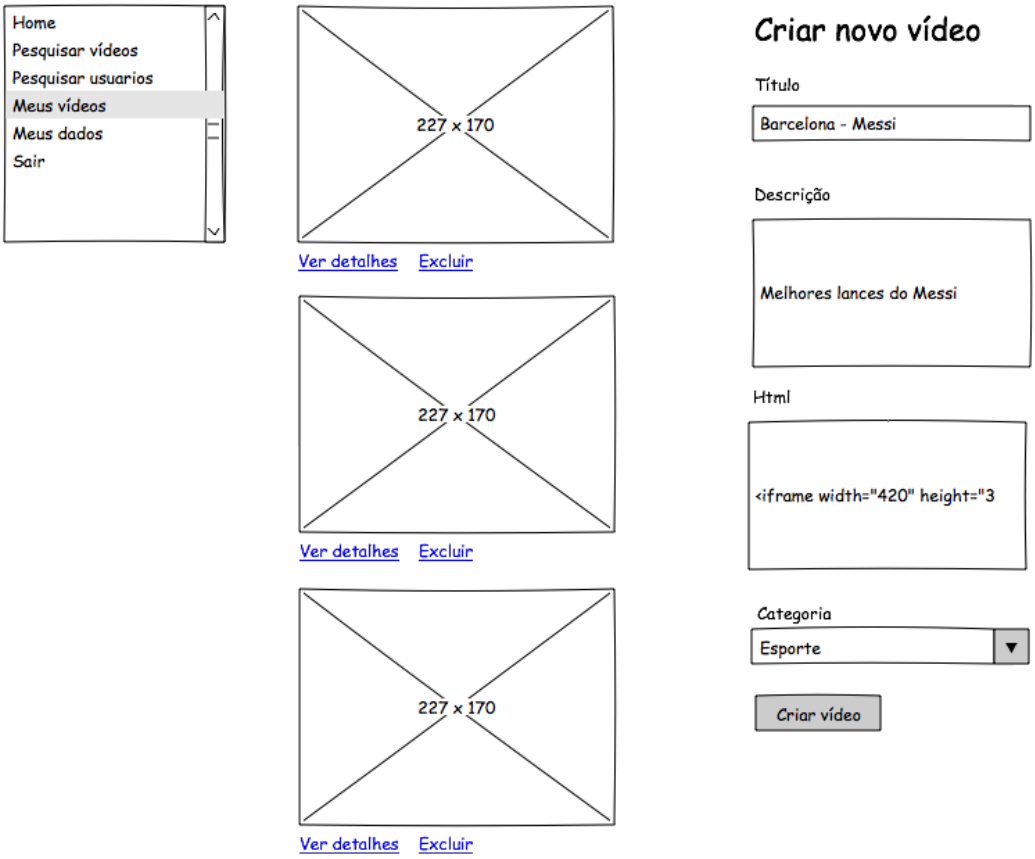
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Não permitir a gravação caso algum campo não esteja preenchido, sinalizando erro nos campos em branco e retornando MSG01.
RN02	O vídeo não ficará hospedado no sistema, será cadastrado apenas o link incorporado fornecido pelo youtube ou vimeo.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com os campos vazios, prontos para a inserção de novos registros.
RI02	O sistema apresenta a tela com os campos preenchidos.

Interface:



Fluxos principais:

Cadastrar vídeo:

Ator	Sistema
Acessa o menu Meus vídeos.	Apresenta tela conforme: RI01.
Preenche o formulário com as informações do vídeo.	
Pressiona o botão: Criar vídeo.	Valida ação conforme RN01.
	Cria novo registro de vídeo na base de dados.
	Apresenta tela conforme: RI01.

Fluxos exceção:

1-Todos os campos são de preenchimento obrigatório:

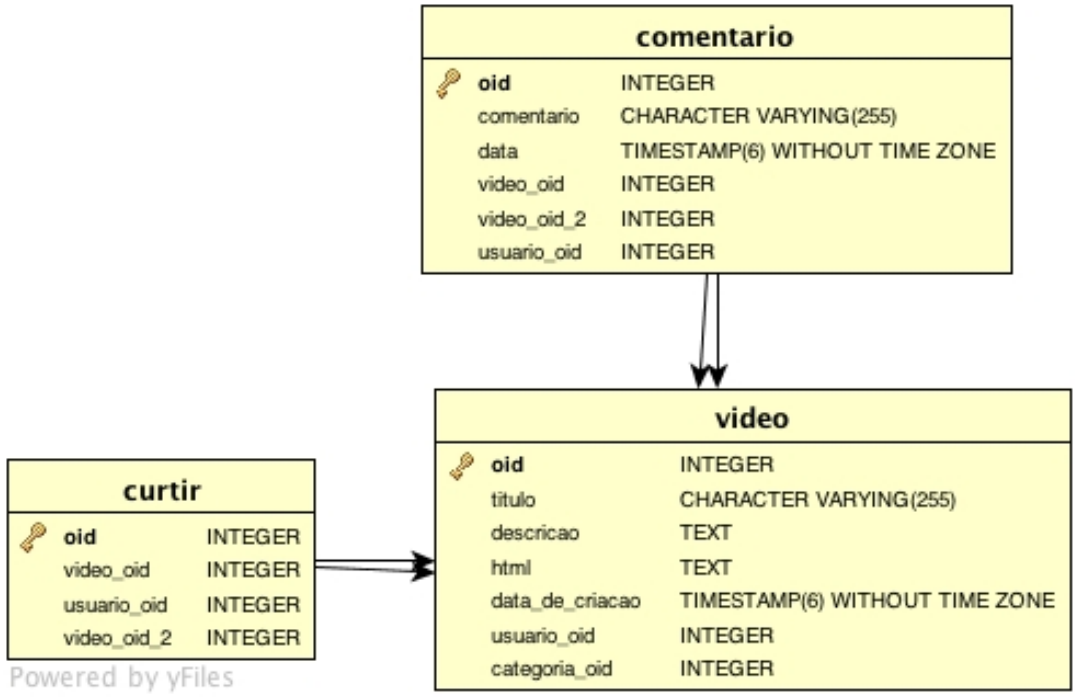
Ator	Sistema
Acessa o menu Meus vídeos.	Apresenta tela conforme: RI01.
Preenche o formulário com as informações do vídeo, porém deixa algum campo em branco.	
Pressiona o botão: Criar vídeo.	Valida ação conforme RN01.
	Apresenta a tela conforme: RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Titulo	Informe o título.	Input text	Varchar	150	Sim
Descricao	Informe a descrição.	Textarea	Text		Sim
Html	Informe o link incorporado do youtube ou vimeo.	Textarea	Text / Html		Sim
Categoria	Informe a categoria.	Select	Varchar / Integer		Sim

Nome do caso de uso: 10- Exclusão de vídeo
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá excluir um vídeo cadastrado previamente por ele mesmo.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema, usuário deve ser o responsável pelo cadastro do vídeo.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao excluir vídeo.

Regras de negócio:

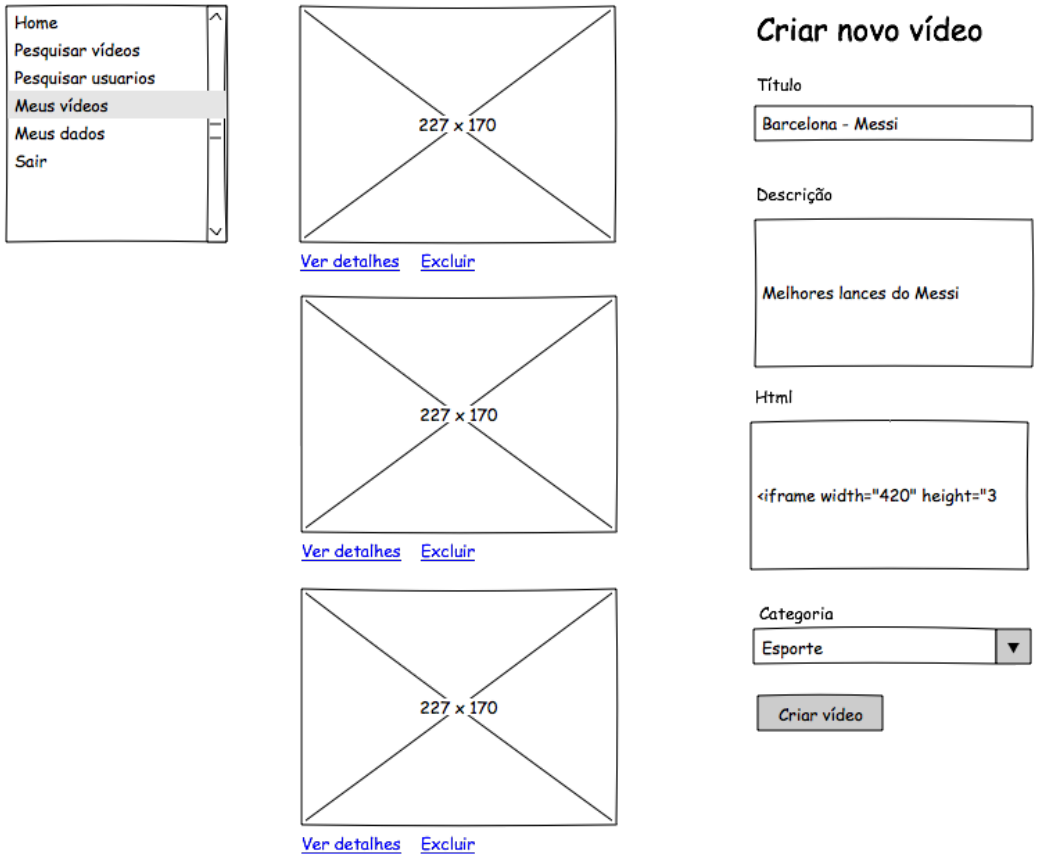
Cód. Regra	Descrição da Regra
RN01	Não permitir a exclusão caso o usuário não for o responsável pelo cadastro do vídeo.

RN02	Qualquer eventual erro o sistema deverá gerar uma exceção com a mensagem MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	Apresenta relatório com os vídeos do usuário.

Interface:



Fluxos principais:

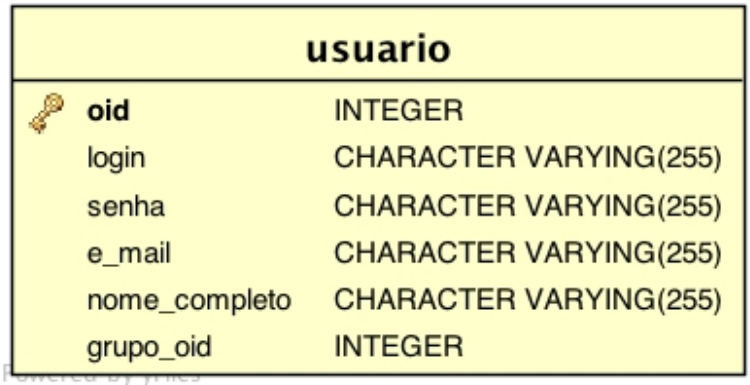
Excluir vídeo:

Ator	Sistema
Acessa o menu Meus vídeos.	Apresenta tela conforme: RI01.
Clica no link Excluir de algum vídeo do relatório.	Valida ação conforme RN01.
	Remove vídeo informado da base de dados.
	Apresenta tela conforme: RI01.

Nome do caso de uso: 11- Alteração de dados do usuário
Prioridade: Média.
Estado: Definido.

Resumo: Nessa rotina, o usuário utilizador do sistema poderá alterar os seus dados de cadastro.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao alterar o cadastro.

Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Não permitir a gravação caso algum campo não esteja preenchido, sinalizando erro nos campos em branco.
RN02	É permitido alterar apenas Nome Completo e E-mail.
RN03	Somente o usuário utilizador tem permissão para alterar seus dados.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com os campos preenchidos.

Interface:

Fluxos principais:

Alterar dados:

Ator	Sistema
Acessa o menu Meus dados.	Apresenta tela conforme: RI01.
Altera os campos conforme desejado.	

Pressiona o botão: Salvar.	Valida ação conforme RN01 e RN03.
	Altera registro do usuário na base de dados.
	Apresenta tela conforme: RI01.

Fluxos exceção:

1-Nenhum campo pode estar em branco:

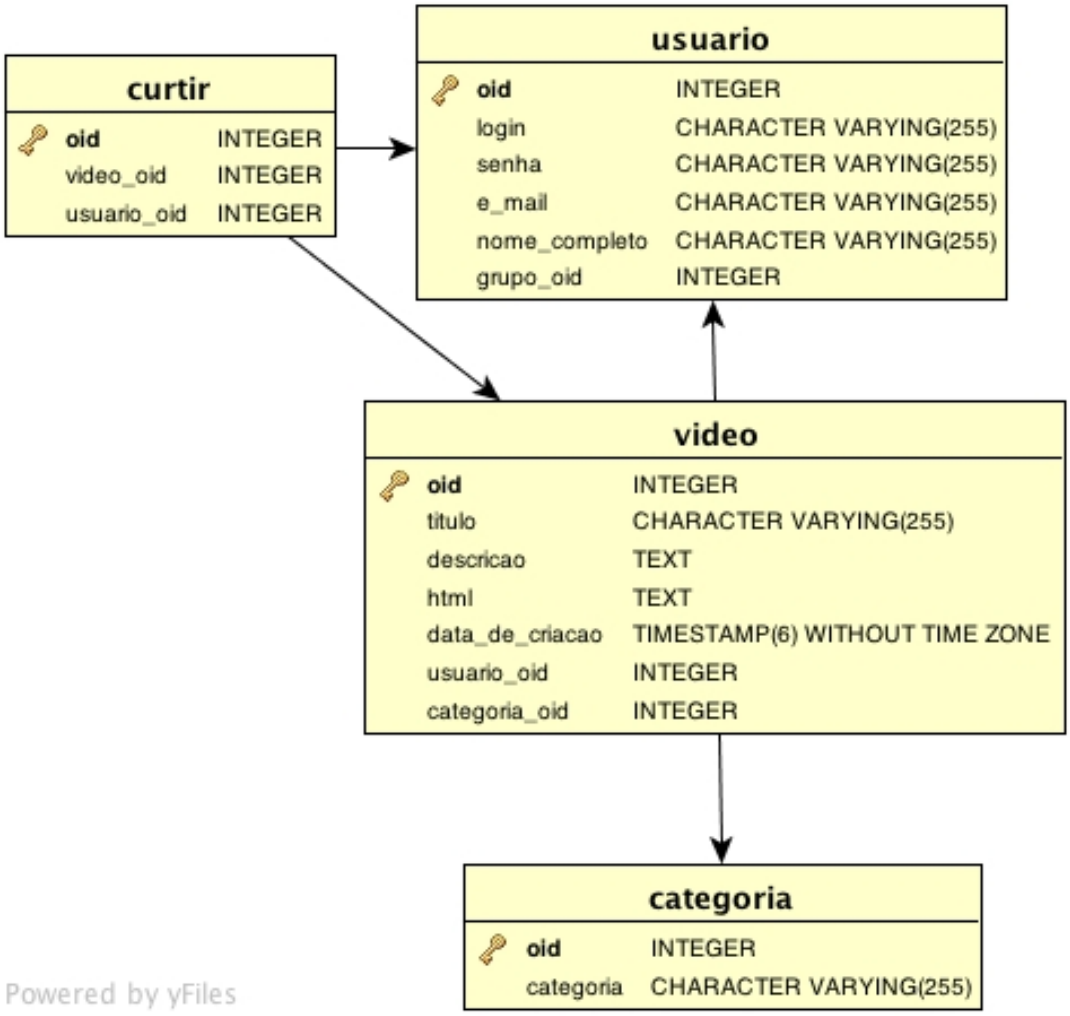
Ator	Sistema
Acessa o menu Meus dados.	Apresenta tela conforme: RI01
Apaga o conteúdo de algum dos campos.	
Pressiona o botão: Salvar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome completo	Informe o nome completo.	Input text	Varchar	150	Sim
E-mail	Informe a e-mail.	Input text	Varchar	70	Sim

Nome do caso de uso: 12- Consulta de informações do vídeo
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar informações detalhadas de um vídeo.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Vídeo não encontrado.

Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Se o vídeo solicitado não existir na base de dados, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com os campos descrição, categoria, autor, data de criação e número total de curtidas.
RI02	O sistema deve apresentar um link que remete ao caso de uso “Consulta informações do usuário” para o autor do vídeo.

Interface:

Fluxos principais:

Visualizar informações do vídeo:

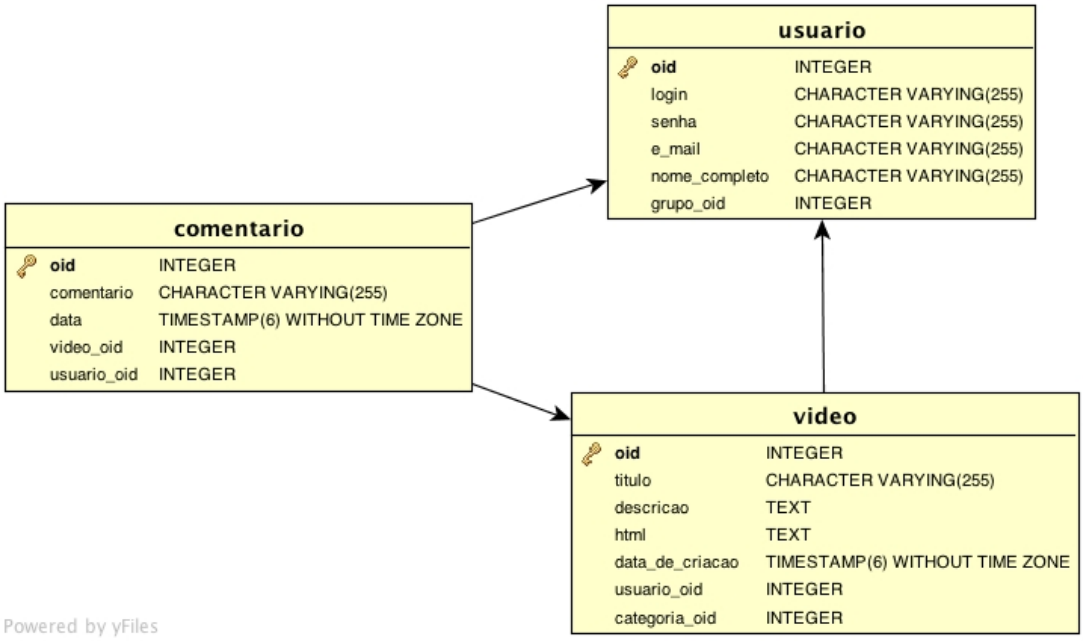
Ator	Sistema
Acessa detalhes de algum vídeo.	Apresenta tela conforme: RI01 e RI02.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Html	Informa link incorporado do vídeo.	Label	Text / Html		Sim
Descrição	Informa a descrição do vídeo.	Label	Text		Sim
Categoria	Informa a categoria do vídeo.	Label	Varchar	150	Sim
Autor	Informa o nome completo do autor.	Label	Varchar	150	Sim
Data de Criação	Informa a data de cadastro do vídeo.	Label	Date		
Curtidas	Informa o total de usuários que sinalizaram o vídeo na opção de curtir.	Label	Integer		Sim

Nome do caso de uso: 13- Cadastro de comentário ao vídeo
Prioridade: Média.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá cadastrar um comentário para um vídeo.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao comentar o vídeo.
MSG02	Nenhum comentário cadastrado.

Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Não permitir a gravação caso o campo comentário não esteja preenchido, sinalizando erro no campo e retornando MSG01.
RN02	Será exibido na tela todos os comentários cadastrados anteriormente.
RN03	Caso não exista nenhum comentário cadastrado, retornar a MSG02.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta a tela com o campo comentário vazio, pronto para a inserção de um novo registro.
RI02	O sistema apresenta o relatório de comentários com os campos preenchidos.

Interface:

Fluxos principais:

Cadastrar comentário:

Ator	Sistema
Acessa os detalhes de algum vídeo.	Apresenta tela conforme: RI01 e RI02.
Preenche o campo Comentário.	
Pressiona o botão: Comentar.	Valida ação conforme RN01.
	Insere novo registro de comentário na base de dados
	Apresenta tela conforme: RI01 e RI02.

Fluxos exceção:

1-Campo comentário é de preenchimento obrigatório:

Ator	Sistema
Acessa os detalhes de algum vídeo.	Apresenta tela conforme: RI01 e RI02.
Não preenche o campo comentário.	
Pressiona o botão: Comentar.	Valida ação conforme RN01.
	Apresenta tela conforme: RI01 e RI02.

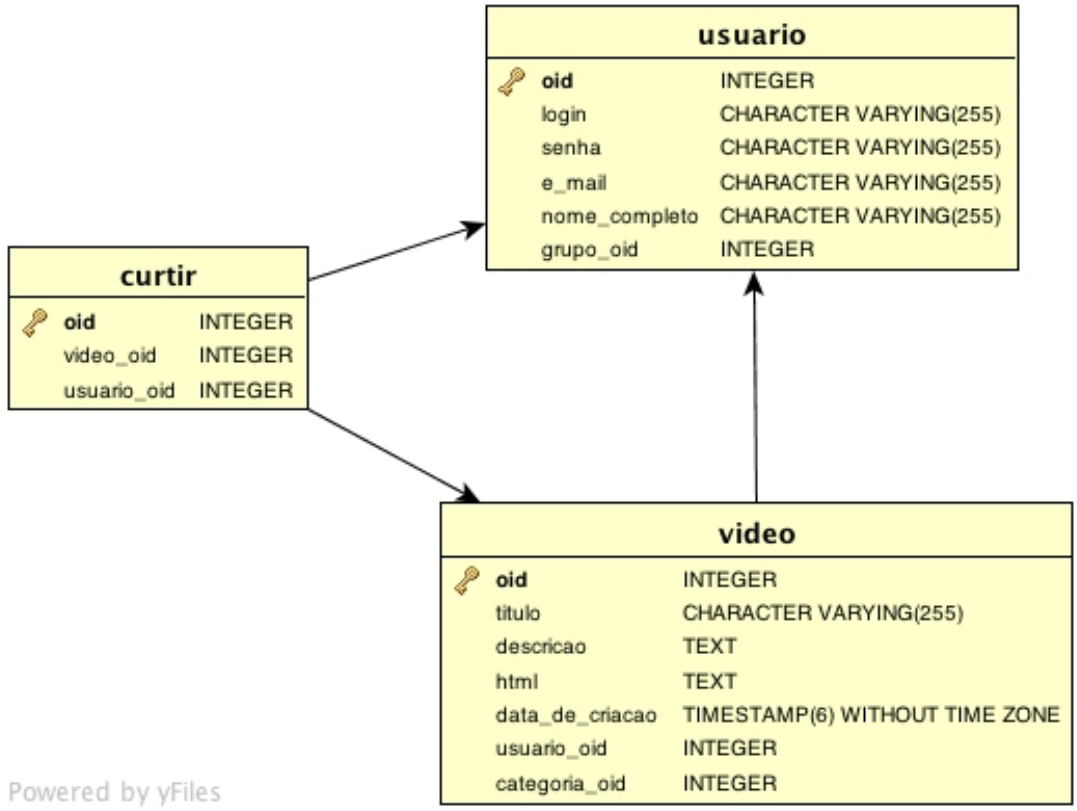
Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Comentário	Informe o comentário.	Textarea	Varchar	150	Sim
Data	Informa a data de criação do comentário.	Label	Date		Sim

Autor	Informa o nome completo do usuário responsável pelo comentário.	Label	Varchar	150	Sim
Comentário	Informa o comentário.	Label	Text		Sim

Nome do caso de uso: 14- Opção de curtir um vídeo
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá curtir um vídeo cadastrado na base de dados.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao curtir vídeo.

Regras de negócio:

Cód. Regra	Descrição da Regra

RN01	Cada usuário pode curtir um vídeo uma única vez, se tentar curtir novamente não será contabilizado.
RN02	Qualquer eventual erro o sistema deverá gerar uma exceção com a mensagem MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta na tela um botão Curtir.

Interface:

Fluxos principais:

Curtir um vídeo:

Ator	Sistema
Acessa os detalhes de algum vídeo.	Apresenta tela conforme: RI01.
Clica no link Curtir.	Valida ação conforme RN01.
	Insere novo registro de curtir na base de dados.
	Apresenta tela conforme: RI01.

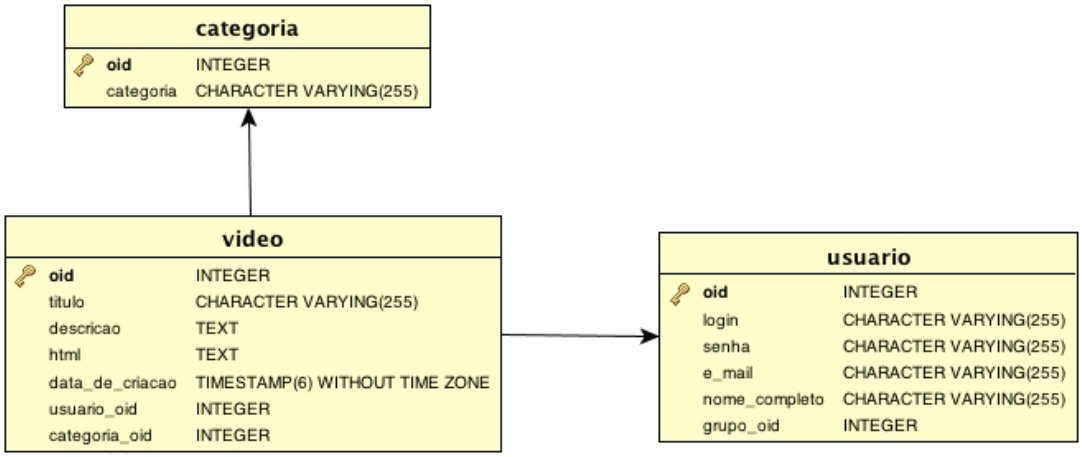
Fluxos exceção:

1-Casa usuário pode curtir um vídeo uma única vez.

Ator	Sistema
Acessa os detalhes de algum vídeo.	Apresenta tela conforme: RI01.
Clica no link Curtir.	Valida ação conforme RN01.
	Não executa nenhuma ação.
	Apresenta tela conforme: RI01.

Nome do caso de uso: 15- Consulta de informações do usuário
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá consultar informações detalhadas de um usuário.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Usuário não encontrado.

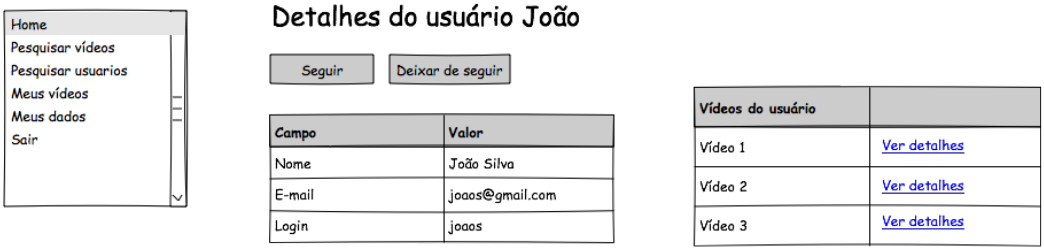
Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Se o usuário solicitado não existir na base de dados, o sistema retornará a MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	O sistema apresenta o relatório com os campos nome, e-mail e login.
RI02	O sistema deve apresentar um link que remete ao caso de uso “Opção de seguir um usuário”.
RI03	O sistema deve apresentar um link que remete ao caso de uso “Opção de deixar de seguir um usuário”.
RI04	O sistema deve apresentar um relatório com os vídeos cadastrados pelo usuário, informando o campo título e um link para o caso de uso “Consulta de informações do vídeo”.

Interface:



Fluxos principais:

Visualizar detalhes de um usuário:

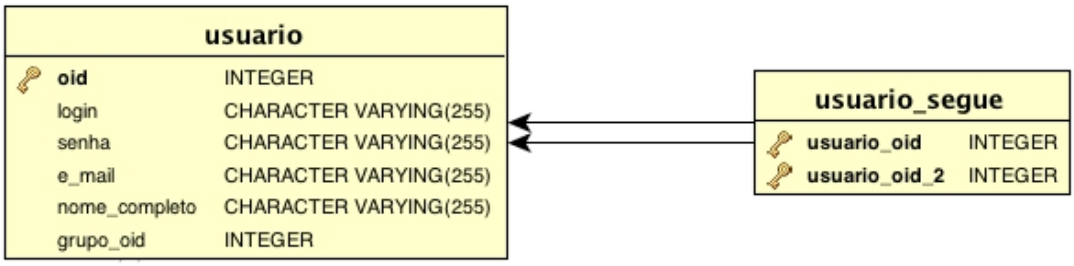
Ator	Sistema
Acessa detalhes de algum usuário.	Apresenta tela conforme: RI01, RI02, RI03 e RI04.

Campos:

Campo	Descrição	Controle	TP Dado	TAM	OBR
Nome	Informa nome completo do usuário.	Label	Varchar	150	Sim
E-mail	Informa o e-mail do usuário.	Label	Varchar	60	Sim
Login	Informa o login do usuário.	Label	Varchar	16	Sim

Nome do caso de uso: 16- Opção de seguir um usuário
Prioridade: Alta.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá sinalizar um usuário do sistema com a opção de seguir.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao seguir usuário.

Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	Cada usuário pode seguir outro usuário.
RN02	Qualquer eventual erro o sistema deverá gerar uma exceção com a mensagem MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	Após a execução do caso de uso, o sistema deve retornar para a tela responsável pela chamada de sua execução.

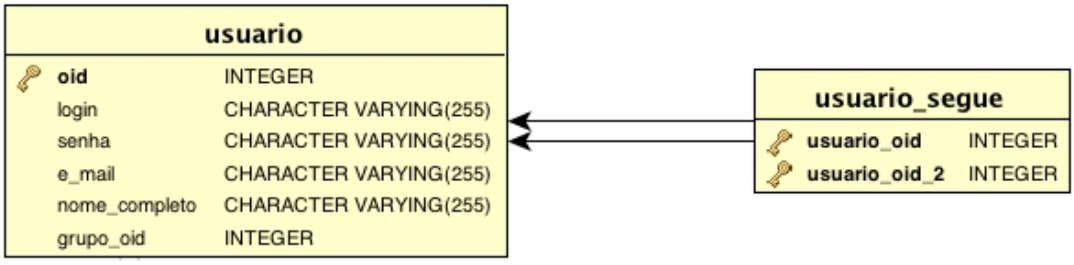
Fluxos principais:

Seguir um usuário:

Ator	Sistema
Exibe um relatório de usuários do sistema.	
Clica no link para seguir o usuário.	Registra novo item de usuario_sgue na base de dados do sistema.
	Apresenta tela conforme RI01.

Nome do caso de uso: 17- Opção de deixar de seguir um usuário
Prioridade: Média.
Estado: Definido.
Resumo: Nessa rotina, o usuário utilizador do sistema poderá retirar a sinalização de seguir de um usuário do sistema.
Atores: Usuário utilizador do sistema.
Pré-condições: Usuário deve estar autenticado no sistema.

Relacionamento das entidades:



Mensagens do sistema:

Cód. Mensagem	Descrição da Mensagem
MSG01	Erro ao deixar de seguir usuário.

Regras de negócio:

Cód. Regra	Descrição da Regra
RN01	O usuário deve estar seguindo o usuário para poder então deixar de segui-lo.
RN02	Qualquer eventual erro o sistema deverá gerar uma exceção com a mensagem MSG01.

Regras de interface:

Cód. Interface	Descrição da Regra
RI01	Após a execução do caso de uso, o sistema deve retornar para a tela responsável pela chamada de sua execução.

Fluxos principais:

Deixar de seguir usuário:

Ator	Sistema
Exibe um relatório de usuários do sistema.	
Clica no link para deixar de seguir o usuário.	Valida ação conforme RN01.
	Remove registro de usuario_sgue na base de dados do sistema.
	Apresenta tela conforme R101.

Fluxos exceção:

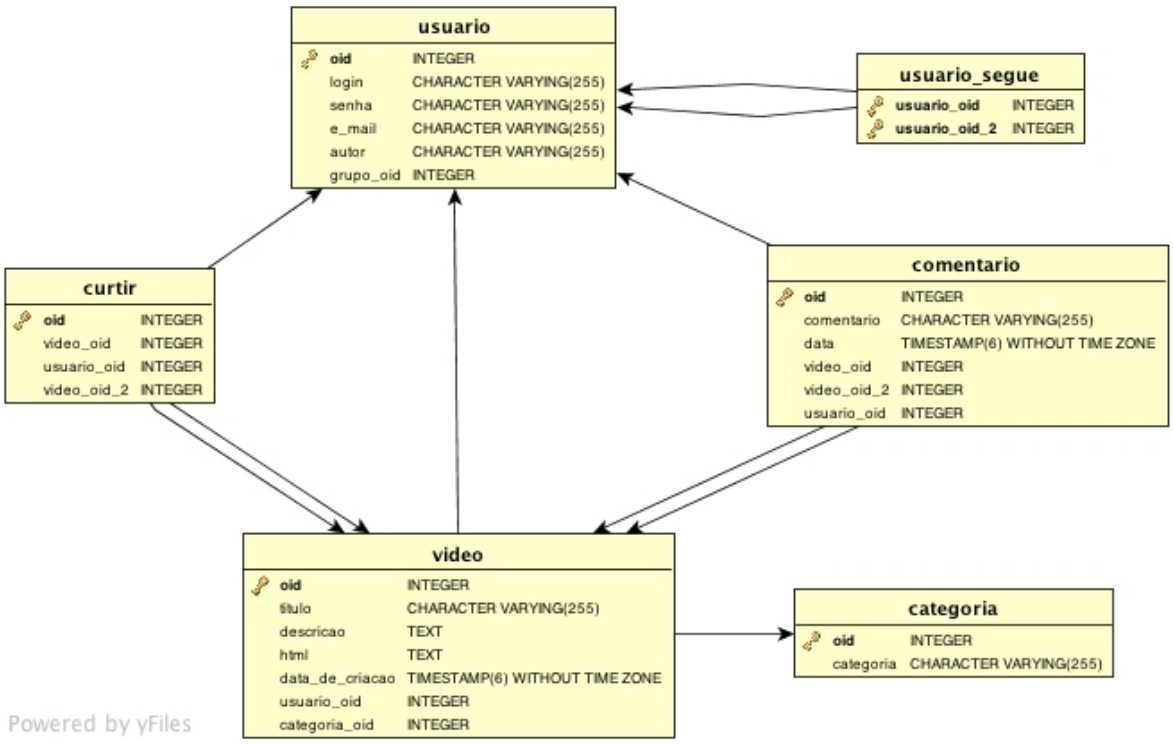
1-Usuário deve estar sendo seguido pelo utilizador:

Ator	Sistema
Exibe um relatório de usuários do sistema.	
Clica no link para deixar de seguir o usuário.	Valida ação conforme RN01.
	Não executa nenhuma ação.
	Apresenta tela conforme R101.

APÊNDICE B – CONTAGEM DETALHADA DOS PONTOS DE FUNÇÃO

Contagem de funções de dados:

A tabela usuario_segue não foi considerada na contagem das funções de dados do sistema , pois não é mantida pelo usuário através das funcionalidades disponibilizadas. Apenas foi considera para determinar a complexidade das transações que a referencia (como ALR – Arquivos Lógicos Referenciados).



Detalhamento da contagem:

Página de login

Tela padrão do sistema para usuários que não estão logados. Nesta ela é possível efetuar o login ou cadastrar-se no sistema.

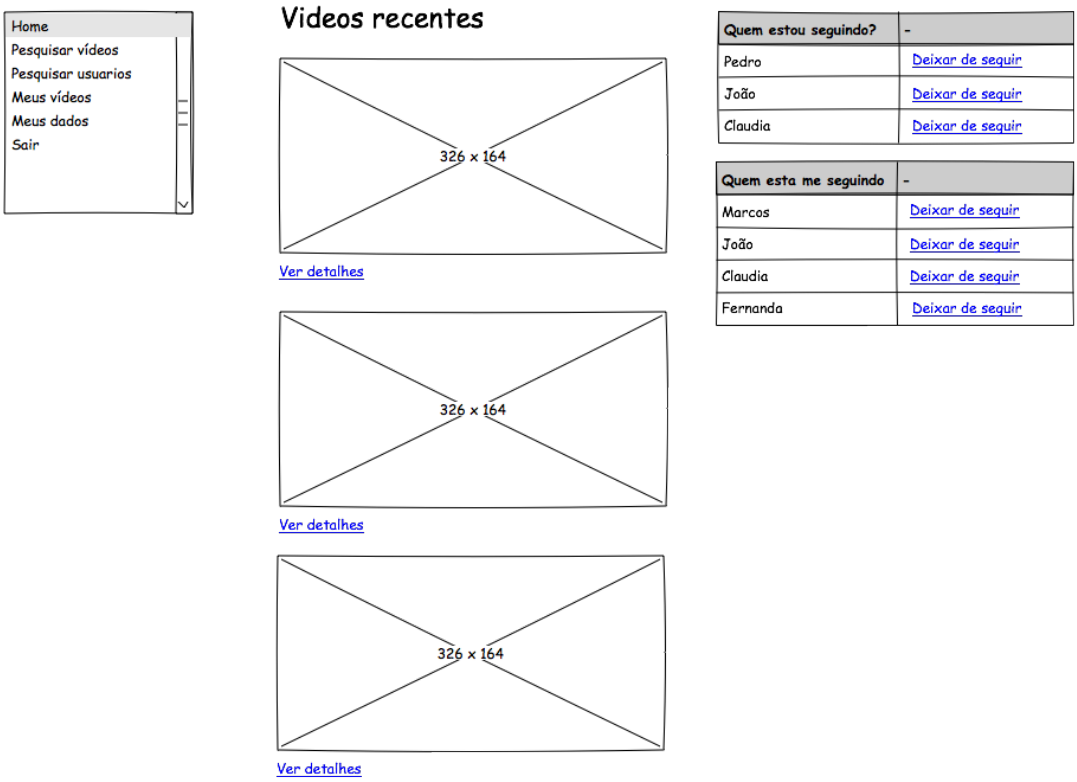
Página de login	Tipo	TD	AR/TR	Complex.	PF
Logar-se utilizando usuário e senha	EE	2	1	Baixa	3
Criar um cadastro	EE	5	1	Baixa	3
					6

Logar-se utilizando usuário e senha:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Login, Senha

Criar um cadastro
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome Completo, E-mail, Login, Senha, Confirmar senha

Home

Tela padrão para usuários logados no sistema. Nesta tela é possível visualizar vídeos recentes de usuários que o usuário está seguindo. Além de listar usuários que o está seguindo e os usuários que estão seguindo-o.



Home	Tipo	TD	AR/TR	Complex.	PF
Consultar vídeos recentes de quem estou seguindo	CE	2	2	Baixa	3
Consultar usuários que estou seguindo	CE	1	2	Baixa	3
Consultar usuários que estão me seguindo	CE	1	2	Baixa	3
Seguir um usuário	EE	1	1	Baixa	3
Deixar de seguir um usuário	EE	1	1	Baixa	3
					15

Consultar vídeos recentes de quem estou seguindo:
Arquivos referenciados (AR/TR): Vídeo, Usuario_segue
Tipos de dados (TD): Título, Html

Consultar usuários que estou seguindo:
Arquivos referenciados (AR/TR): Usuário, Usuario_segue
Tipos de dados (TD): Nome completo

Consultar usuários que estão me seguindo:
Arquivos referenciados (AR/TR): Usuário, Usuario_segue
Tipos de dados (TD): Nome completo

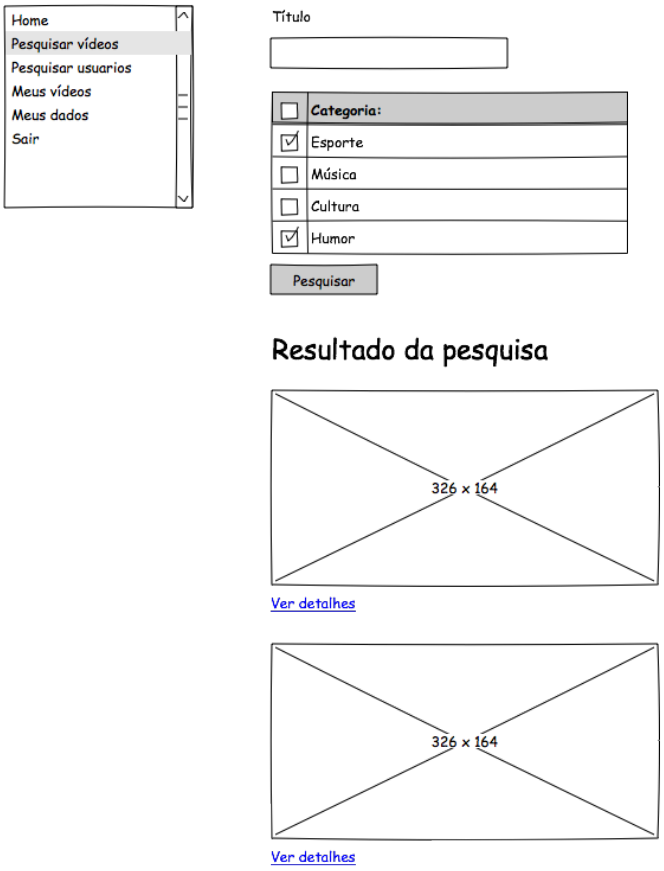
Seguir um usuário:
Arquivos referenciados (AR/TR): Usuario_segue

Tipos de dados (TD): Id do usuário

Deixar de seguir um usuário:
Arquivos referenciados (AR/TR): Usuario_segue
Tipos de dados (TD): Id do usuário

Pesquisar vídeos

Tela onde é possível pesquisar por vídeos. É obrigatório a sugestão de um termo para a pesquisa e opcional restringir categorias para a pesquisa.



Pesquisar vídeos	Tipo	TD	AR/TR	Complex.	PF
Listar categorias que podem ser utilizadas na pesquisa	CE	1	1	Baixa	3
Pesquisar vídeo informando o título e categorias	EE	2	1	Baixa	3
Consultar vídeos que contém o termo informado	SE	4	2	Baixa	4
					10

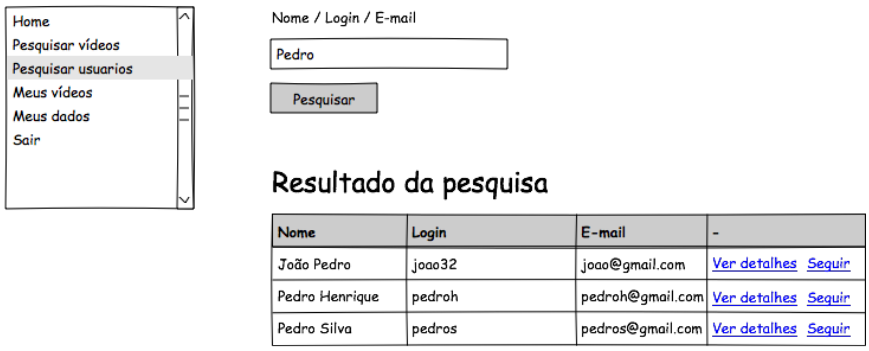
Listar categorias que podem ser utilizadas na pesquisa:
Arquivos referenciados (AR/TR): Categoria
Tipos de dados (TD): Descrição

Pesquisar vídeo informando o título e categorias:
Arquivos referenciados (AR/TR): Video
Tipos de dados (TD): Título, Categoria

Consultar vídeos que contém o termo informado:
Arquivos referenciados (AR/TR): Vídeo, Usuário
Tipos de dados (TD): Data de Criação, Título, Html, Nome do Autor

Pesquisar usuários

Tela para pesquisar por usuários do sistema. É obrigatório sugerir um termo para a pesquisa. É possível ver detalhes e seguir os usuários encontrados na pesquisa.



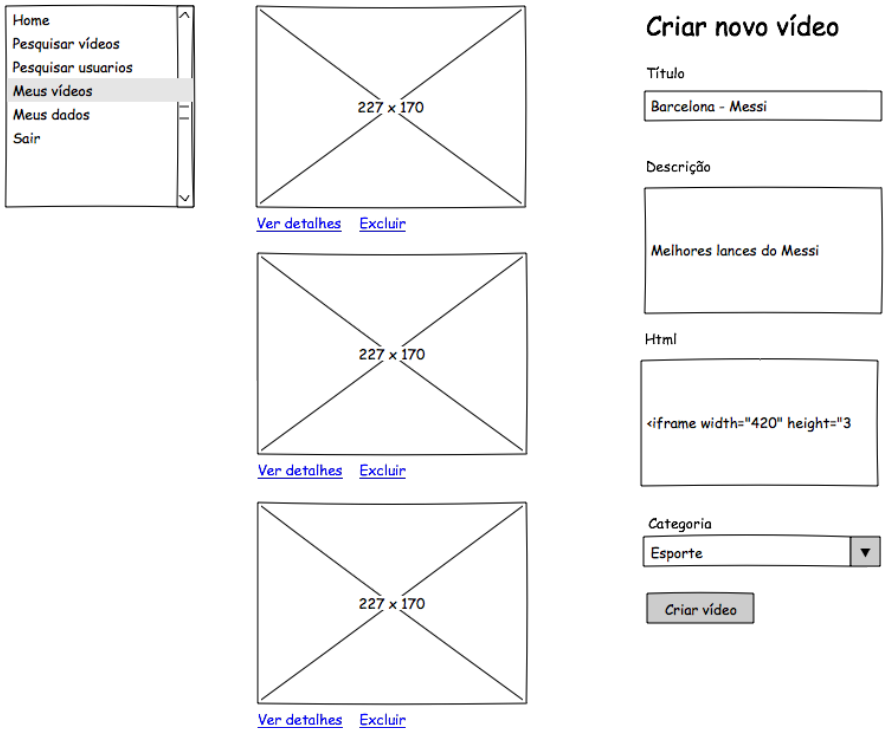
Pesquisar usuários	Tipo	TD	AR/TR	Complex.	PF
Pesquisar usuário informando nome ou login ou e-mail	EE	1	1	Baixa	3
Consultar usuários que contém o termo informado no nome ou login ou e-mail	SE	3	1	Baixa	4
					7

Pesquisar usuário informando nome ou login ou e-mail:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome ou login ou senha

Consultar usuários que contém o termo informado no nome ou login ou e-mail:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome, Login, E-mail

Meus vídeos

Tela para gerenciar os vídeos do usuário. É possível visualizar os vídeos previamente cadastrados e excluí-los. Além de possibilitar cadastrar novos vídeos.



Meus vídeos	Tipo	TD	AR/TR	Complex.	PF
Consultar vídeos que sou autor	CE	1	1	Baixa	3
Consultar categorias que poderão ser escolhidas para o novo vídeo	CE	1	1	Baixa	3
Cadastrar novo vídeo	EE	4	1	Baixa	3
Excluir vídeo	EE	1	1	Baixa	3
					12

Consultar vídeos que sou autor:
Arquivos referenciados (AR/TR): Vídeo
Tipos de dados (TD): Html

Consultar categorias que poderão ser escolhidas para o novo vídeo:
Arquivos referenciados (AR/TR): Categoria
Tipos de dados (TD): Descrição

Cadastrar novo vídeo:
Arquivos referenciados (AR/TR): Vídeo
Tipos de dados (TD): Título, Descrição, Html, Categoria

Excluir vídeo:
Arquivos referenciados (AR/TR): Vídeo
Tipos de dados (TD): Id do vídeo

Meus dados

Tela para edição dos dados do usuário. Os únicos dados editáveis serão o nome e o e-mail.

Meus dados	Tipo	TD	AR/TR	Complex.	PF
Consultar meus dados	CE	2	1	Baixa	3
Alterar meus dados	EE	2	1	Baixa	3
					6

Consultar meus dados:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome, E-mail

Alterar meus dados:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome, E-mail

Detalhes do vídeos

Tela para visualizar detalhes e comentários de um vídeo. Além disso o usuário pode inserir um novo comentário e curtir o vídeo.

Detalhes do vídeo	Tipo	TD	AR/TR	Complex.	PF
Consultar dados do vídeo	SE	7	4	Alta	7
Consultar comentários do vídeo	CE	3	2	Baixa	3
Criar novo comentário	EE	1	1	Baixa	3
Curtir o vídeo	EE	1	1	Baixa	3
					16

Consultar dados do vídeo:
Arquivos referenciados (AR/TR): Vídeo, Curtir, Usuário, Categoria
Tipos de dados (TD): Titulo, Descrição, Html, Data de criação, Nome do autor, Nome da categoria, Total de curtir

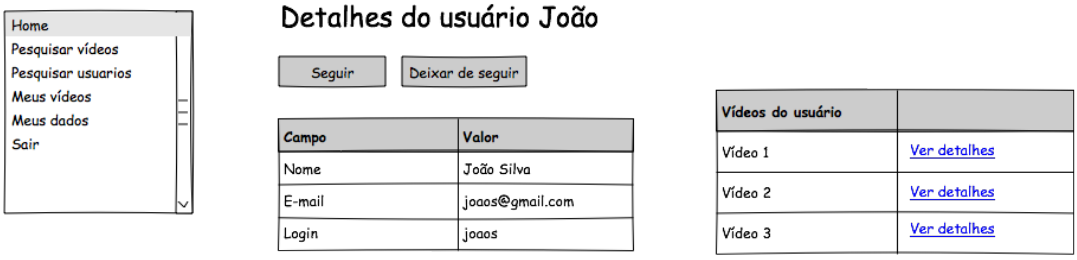
Consultar comentários do vídeo:
Arquivos referenciados (AR/TR): Comentário, Usuário
Tipos de dados (TD): Data de criação, Texto, Nome do autor

Criar novo comentário:
Arquivos referenciados (AR/TR): Comentário
Tipos de dados (TD): Texto

Curtir o vídeo:
Arquivos referenciados (AR/TR): Curtir
Tipos de dados (TD): Id do vídeo

Detalhes do usuário

Tela para visualizar detalhes e vídeos de um usuário. Também é possível seguir ou deixar de seguir o usuário.



Detalhes do usuário	Tipo	TD	AR/TR	Complex.	PF
Consultar dados do usuário	CE	3	1	Baixa	3
Consultar vídeos do usuário	CE	1	1	Baixa	3
Seguir usuário	EE	1	1	Baixa	3
Deixar de seguir usuário	EE	1	1	Baixa	3
					12

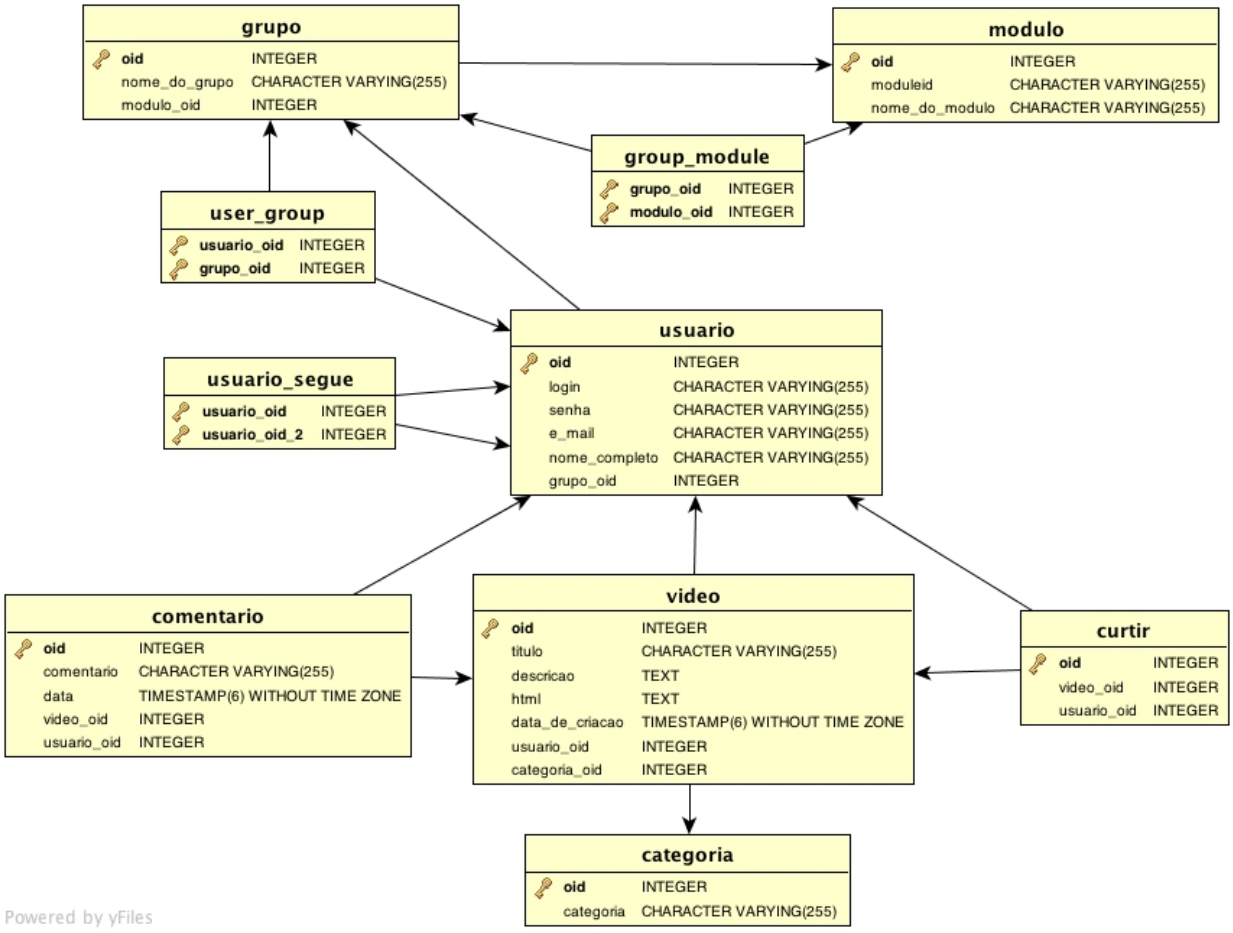
Consultar dados do usuário:
Arquivos referenciados (AR/TR): Usuário
Tipos de dados (TD): Nome, E-mail, Login

Consultar vídeos do usuário:
Arquivos referenciados (AR/TR): Vídeo
Tipos de dados (TD): Data de criação, Título

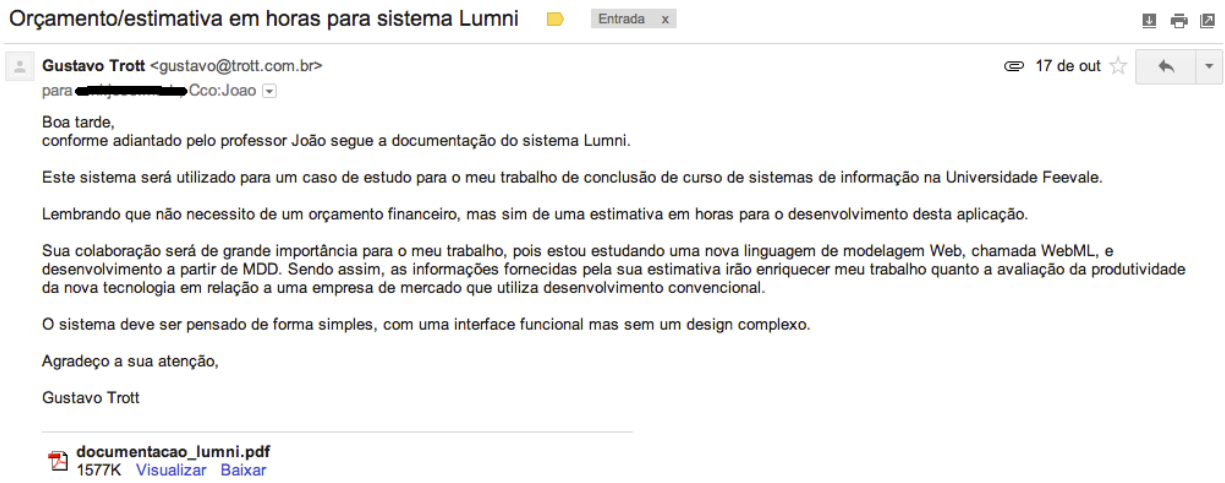
Seguir usuário:
Arquivos referenciados (AR/TR): Usuario_segue
Tipos de dados (TD): Id do usuário

Deixar de seguir usuário:
Arquivos referenciados (AR/TR): Usuario_segue
Tipos de dados (TD): Id do usuário

APÊNDICE C – ESTRUTURA DE TABELAS DO SISTEMA LUMNI



APÊNDICE D – MODELO DE E-MAIL PARA ESTIMATIVA DE ESFORÇO



Orçamento/estimativa em horas para sistema Lumni Entrada x

Gustavo Trott <gustavo@trott.com.br> 17 de out

para [redacted] Cco:Joao

Boa tarde,
conforme adiantado pelo professor João segue a documentação do sistema Lumni.

Este sistema será utilizado para um caso de estudo para o meu trabalho de conclusão de curso de sistemas de informação na Universidade Feevale.

Lembrando que não necessito de um orçamento financeiro, mas sim de uma estimativa em horas para o desenvolvimento desta aplicação.

Sua colaboração será de grande importância para o meu trabalho, pois estou estudando uma nova linguagem de modelagem Web, chamada WebML, e desenvolvimento a partir de MDD. Sendo assim, as informações fornecidas pela sua estimativa irão enriquecer meu trabalho quanto a avaliação da produtividade da nova tecnologia em relação a uma empresa de mercado que utiliza desenvolvimento convencional.

O sistema deve ser pensado de forma simples, com uma interface funcional mas sem um design complexo.

Agradeço a sua atenção,

Gustavo Trott

documentacao_lumni.pdf
1577K Visualizar Baixar