UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

WILLIAM SILVA GOMES

LRDB: um Repositório *Online* de Revisões Literárias de Computação

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Profª. Drª. Ingrid Oliveira de Nunes

Porto Alegre
2016

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”

— Albert Einstein

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association for Computing Machinery
CSS	Cascading Style Sheets
DOI	Digital Object Identifier
EBSE	Evidence-based Software Engineering
HTTP	Hypertext Tranfer Protocol
ms	Milissegundo
MVC	Model-View-Controller
SQL	Structured Query Language
URL	Uniform Resource Locator

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

RESUMO

O presente trabalho apresenta a especificação, o projeto e a implementação de um repositório *web* de revisões de literatura relacionadas à computação, com suporte especial a revisões sistemáticas. Devido ao crescente volume de revisões sistemáticas produzidas no âmbito de engenharia de software, enfatizam-se os conceitos e a aplicabilidade das revisões sistemáticas nesse contexto. A revisão sistemática constitui o processo de avaliar e interpretar todos os estudos relevantes sobre um determinado tópico de interesse. A prática desse tipo de estudo revolucionou as pesquisas na área médica e observa-se que aplicar a ideia à engenharia de software e à computação como um todo pode trazer diversos benefícios aos profissionais e pesquisadores da área. O trabalho apresenta os principais conceitos de revisões sistemáticas, contextualiza o tema no cenário de engenharia de software e apresenta o projeto, bem como a implementação, de um sistema que centraliza as revisões sistemáticas já publicadas, chamado LRDB. Também é apresentado o processo de população inicial do repositório, envolvendo revisões sistemáticas de engenharia de software.

Palavras-chave: Revisão sistemática, repositório, engenharia de software, *web*.

LRDB: an Online Repository of Computing Literature Reviews

ABSTRACT

This work presents the design and implementation of a web repository of surveys related to computing with special support to systematic reviews. Due to the increasing amount of systematic reviews produced in the software engineering area, we emphasize the concepts and the applicability of systematic reviews in this context. The systematic review is the process of evaluating and interpreting all relevant studies of a particular topic of interest. The practice of systematic reviews revolutionized medical research and applying this idea in software engineering and in computing as a whole can bring many benefits to practitioners and researchers. The paper presents the main concepts of systematic reviews, contextualizes the topic in the software engineering scope and presents the design and implementation of a web system that makes available systematic reviews that have already been published in a single location. The process of initial population of the repository, involving systematic reviews of software engineering, is also presented.

Keywords: Systematic reviews, repository, software engineering, web.

1 INTRODUÇÃO

A quantidade de estudos publicados envolvendo temas ligados à computação vem crescendo muito nos últimos anos. Em meio a tantas novidades, manter-se atualizado e ter acesso aos estudos mais relevantes pode se tornar uma tarefa bastante complicada. O volume de informação é muito grande, tanto em complexidade quanto em abrangência. Além disso, encontrar trabalhos confiáveis e com qualidade superior é ainda mais difícil. Essa realidade se estende à grande parte da ciência da computação, mas tem se intensificado no âmbito de engenharia de software (TICHY et al., 1995) (SJØBERG et al., 2005) (TRENDOWICZ; MÜNCH; JEFFERY, 2011). Artigos com grande valor científico são publicados em conferências com uma velocidade maior do que em periódicos, agravando o problema.

Como uma das alternativas para organizar, selecionar e sintetizar todo o conhecimento adquirido sobre um dado tópico, surge o conceito de revisão sistemática. Trata-se de uma técnica metódica para identificar, avaliar e analisar estudos primários com o objetivo de investigar uma questão específica (STAPLES; NIAZI, 2007a). Dessa forma, é possível extrapolar achados de estudos independentes, avaliar a consistência de cada um deles e explicar possíveis divergências e conflitos (MULROW, 1994). Essa ideia surgiu na área médica, frente à necessidade de lidar com um número enorme de estudos com resultados conflitantes. Percebeu-se também que tomar decisões baseadas no acúmulo de resultados de experimentos científicos é mais confiável do que se basear somente na opinião de especialistas (KITCHENHAM et al., 2009). Os resultados dos experimentos científicos são chamados de evidência. Na área médica, a adoção da pesquisa baseada em evidências mudou radicalmente o panorama dos estudos. Revisões sistemáticas relacionam-se com a abordagem baseada em evidência porque consistem no melhor método para atingir bons resultados.

Baseando-se no enorme benefício que a aplicação de pesquisas baseadas em evidência trouxe à medicina e a outras ciências, Kitchenham et al. (2004) propôs que a ideia fosse aplicada ao contexto de engenharia de software, produzindo o conceito de engenharia de software baseada em evidências. Desde então, a prática de revisões sistemáticas vem ganhando expressiva popularidade na área. Novas revisões tem sido publicadas constantemente, referindo-se à vários tópicos de engenharia de software (DYBÅ; DINGSØYR, 2008a).

Kitchenham et al. (2004) elaboraram uma série de passos para a construção de

uma revisão sistemática, baseando-se nos moldes estabelecidos para o âmbito médico. Contudo, adequaram o processo à realidade da engenharia de software. Assim, o primeiro item do roteiro proposto consiste em identificar uma real necessidade de fazer uma revisão sistemática. Os pesquisadores devem, no início, encontrar e analisar todas as revisões sistemáticas existentes sobre o fenômeno de interesse, visando identificar o que já foi feito. Contudo, Brereton et al. (2005) revelam um grande problema: *"mecanismos de busca de trabalhos de engenharia de software não são desenvolvidos de modo a suportar revisões sistemáticas"*. Ainda que existam ferramentas que auxiliem na condução de uma revisão sistemática e outras que ajudam a gerenciar referências, não há um local que centralize as revisões sistemáticas existentes.

1.1 Contribuição

Tendo em vista o avanço das revisões sistemáticas no âmbito de engenharia de software e a provável expansão para as demais áreas da computação, neste trabalho propõe-se o projeto e o desenvolvimento de um repositório *web* de revisões sistemáticas de computação, chamado LRDB, sigla que significa *Literature Reviews Database* em inglês. A ideia é criar um ambiente colaborativo no qual os usuários poderão cadastrar e pesquisar por revisões sistemáticas já existentes de todas as áreas da computação. O foco inicial é em engenharia de software, mas propõe-se a implementação de um sistema mais abrangente, podendo beneficiar toda a comunidade relacionada à computação. Dessa forma, além de revisões sistemáticas, o sistema permite o cadastro de qualquer revisão de literatura ligada à área, havendo uma distinção entre revisões tradicionais e revisões sistemáticas, permitindo realizar buscas de acordo com os metadados mais relevantes, o que é fundamental na primeira etapa do desenvolvimento de revisões sistemáticas.

O objetivo principal é que o repositório armazene somente os metadados das revisões. O texto completo de cada trabalho deve estar disponível em algum outro repositório e é somente referenciado no presente sistema. Assim, ao pesquisar uma publicação, o usuário tem acesso a todos os metadados da mesma e, para acessá-la na íntegra, é direcionado à página oficial que a armazena. Essa decisão foi tomada porque o acesso a diversos trabalhos é cobrado e armazenar as publicações propriamente ditas pode trazer problemas de direitos autorais.

Trata-se de um modelo inicial que deverá ser mantido e aperfeiçoado, contando com a ajuda de toda a comunidade envolvida. Espera-se que o sistema evolua e inspire

novas ideias com objetivos similares, beneficiando todos aqueles que buscam conhecimento. Além da implementação do sistema, foi realizado o cadastro inicial de algumas revisões sistemáticas existentes de engenharia de software, o que nos permite gerar algumas métricas sobre o sistema, observar seus benefícios e possíveis dificuldades futuras. Algumas melhorias também são observadas, de modo a construir um sistema útil e prático para todo o meio acadêmico.

1.2 Estrutura do Texto

Além desta introdução, o presente trabalho é composto de cinco capítulos. O Capítulo 2 aborda os conceitos de revisões sistemáticas em geral, sua aplicação em engenharia de software e trabalhos relacionados, o que é fundamental para compreender a importância da solução proposta. O Capítulo 3 aborda as funcionalidades e a arquitetura do repositório. A aplicação desenvolvida é apresentada no Capítulo 4, que contém imagens do sistema sob a perspectiva do usuário. O Capítulo 5 trata da população inicial do repositório, discutindo os critérios de busca e resultados encontrados, além de apresentar algumas métricas de desempenho dos mecanismos de busca implementados. Finalmente, a Conclusão é descrita no Capítulo 6, no qual também são discutidas possibilidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Em diversas áreas da ciência, pesquisas são realizadas para solucionar problemas, entender fenômenos e provar teorias (GOUGH; OLIVER; THOMAS, 2012). Com o grande desenvolvimento científico atual, a tarefa de se manter a par de todo conhecimento produzido sobre determinado tópico é praticamente impossível para uma única pessoa (DYBÅ; DINGSØYR, 2008a). No passado isso era possível, mas atualmente tem se tornado cada vez mais difícil (GOUGH; OLIVER; THOMAS, 2012). Portanto, revisões passaram a ser essenciais para todos que desejam se manter atualizados sobre as evidências acumuladas em seu campo de interesse (DYBÅ; DINGSØYR, 2008a) e, segundo Crombie et al. (1994), *"sua importância cresce diretamente proporcional ao número de trabalhos desenvolvidos no tema"*.

Neste capítulo serão apresentados os principais conceitos de revisões sistemáticas, suas diferenças em relação à revisão narrativa, os motivos que promoveram seu surgimento e sua aplicabilidade em de engenharia de software, o que é fundamental para compreender a importância da proposta do trabalho.

2.1 Revisões Sistemáticas

O termo revisão sistemática é utilizado para denotar uma metodologia específica de pesquisa, cujo objetivo é reunir e avaliar rigorosamente todas as evidências disponíveis sobre um tópico particular (BIOLCHINI et al., 2005). As revisões sistemáticas, bem como as demais classes de revisão, são estudos secundários com abordagem retrospectiva, englobando o conhecimento já existente sobre certo assunto. Como o próprio termo indica, ao revisar o que foi produzido, não existe o objetivo de pesquisar novas informações, concentrando-se apenas naquilo que já foi dito sobre o assunto. Assim, revisões sistemáticas empregam estudos originais, isto é, estudos primários como sua "população" (MULROW, 1987) (COOK; SACKETT; SPITZER, 1995).

Ao contrário do processo usual de revisão de literatura, no qual cada autor decide como conduzir o procedimento de revisão, uma revisão sistemática é feita de maneira metódica e formal. *"Uma boa revisão sistemática preocupa-se em encontrar todos os estudos relevantes, avaliar cada um deles em relação à estrutura e execução, e combinar os resultados dos estudos individuais de maneira imparcial"* (CROMBIE; DAVIES, 2009).

Um protocolo é desenvolvido previamente e todo o processo de condução da revisão sistemática segue uma sequencia de passos bem definidos nele especificados (BIOLCHINI et al., 2005). O protocolo utilizado deve ser explicitamente documentado, permitindo que outros pesquisadores possam avaliá-lo e também possam replicar o processo de revisão (DYBÅ; DINGSØYR, 2008b). Todos os passos do protocolo são projetados em relação a um problema específico, o qual é expressado por meio de uma questão bem estruturada e bem definida (BIOLCHINI et al., 2005). O foco na questão guia a escolha dos métodos para obter as respostas (GOUGH; OLIVER; THOMAS, 2012), as quais posteriormente serão usadas para guiar a tomada de decisões (CRD, 2009).

O rigor na metodologia visa garantir que todas as bases de pesquisa relevantes foram consideradas e que uma análise válida dos estudos primários foi realizada, produzindo um estudo transparente, com o mínimo risco de vieses (CRD, 2009). Entende-se por viés, neste contexto, tendência a uma opinião ou preconceito. Além disso, o cuidado para que o estudos sejam selecionados de maneira cuidadosa e imparcial serve como medida de alta qualidade (CROMBIE; DAVIES, 2009).

O tipo de evidência a ser incluído em uma revisão sistemática deve ser definido antecipadamente no seu protocolo (BIOLCHINI et al., 2005). O critério de seleção deve ser claro (DYBÄ; KITCHENHAM; JORGENSEN, 2005), permitindo que os dados de estudos diferentes sejam normalizados de maneira que seus resultados possam ser comparados em termos de magnitude de efeito (BIOLCHINI et al., 2005). Os dados de vários estudos permitem verificar se os resultados são consistentes e podem ser generalizados ou se variam de acordo com subgrupos específicos (DYBÅ; DINGSØYR, 2008b) e, por isso, a etapa de seleção deve ter sua confiabilidade verificada (KITCHENHAM; DYBA; JORGENSEN, 2004).

Ao comparar evidências de estudos diferentes, é possível analisar em que ponto os resultados correspondem e onde eles se contradizem, indicando os padrões e a estrutura do conhecimento existente (DYBÄ; KITCHENHAM; JORGENSEN, 2005). Ao definir o que se sabe, descobre-se também possíveis “buracos” no conhecimento, ou seja, pontos que precisam ser investigados em pesquisas futuras (KITCHENHAM; DYBA; JORGENSEN, 2004). Quando os resultados dos estudos primários são resumidos, mas não são estatisticamente combinados, tem-se uma revisão sistemática qualitativa. Ao combinar os dados de um ou mais estudos usando técnicas estatísticas, produz-se uma revisão sistemática quantitativa, chamada de revisão sistemática com metanálise (COOK; MULROW; HAYNES, 1997a).

Metanálise ou meta-análise é o termo empregado para definir a combinação estatística de resultados de diversos estudos primários (BERWANGER et al., 2007) e, portanto, tem como pré-requisito uma boa revisão sistemática no tópico de interesse (CROMBIE; DAVIES, 2009). A metanálise proporciona uma abordagem útil e racional para lidar com diversas dificuldades práticas que preocupam qualquer um que esteja tentando entender a eficácia de uma pesquisa (CROMBIE; DAVIES, 2009). Com a metanálise, é possível verificar se os resultados são a favor ou contra determinado tratamento ou abordagem, auxiliando a tomada de decisões. Por esse motivo, revisões sistemáticas que possuem metanálise tem um maior valor científico (BERWANGER et al., 2007). Por outro lado, alguns fatores nos resultados podem impedir a associação dos diferentes estudos sob uma única medida, impossibilitando a realização da metanálise (COOK; MULROW; HAYNES, 1997b).

2.1.1 Diferenças das Revisões Tradicionais

Ainda que todas as revisões tenham como meta resumir e organizar o conhecimento para torná-lo mais acessível, a maneira como elas são conduzidas pode ser bastante diferente. Além disso, qualquer revisão está sujeita a erros randômicos e sistemáticos, de tal forma que a sua qualidade reside nos métodos utilizados para minimizá-los e para evitar tendenciosidade de opinião nos resultados (COOK; MULROW; HAYNES, 1997b). E é exatamente neste ponto que está a principal diferença entre as revisões sistemáticas e as revisões narrativas.

Revisões narrativas são estudos amplos, indicados para descrever e discutir o "estado da arte" de um certo tópico sob a ótica teórica ou conceitual. Compõem a abordagem tradicional de revisão literária e são úteis para a educação continuada, permitindo ao leitor situar-se sobre certo tema em um curto espaço de tempo (ROTHER, 2007). Por outro lado, revisões narrativas não informam o método a partir do qual foram produzidas, nem o critério de seleção dos estudos primários (BERWANGER et al., 2007). Elas geralmente baseiam-se na experiência e opiniões do autor, o qual é, na maioria das vezes, um especialista na área (CIPRIANI; GEDDES, 2003). A falta de um método explícito para a realização da revisão pode gerar diversas falhas na metodologia e, consequentemente, influenciar as conclusões (COOK; MULROW; HAYNES, 1997a), que podem ser motivadas inclusive por interesses pessoais do autor na área (GOUGH; OLIVER; THOMAS, 2012). Além disso, o leitor dificilmente consegue identificar quais recomendações são

baseadas na experiência do autor, nem saber a amplitude literária alvo da revisão, dificultando a compreensão de porquê alguns estudos ganharam mais ênfase do que outros. Uma revisão narrativa não permite saber com clareza se o autor selecionou alguns trabalhos porque reforçam seu ponto de vista ou excluiu outros porque contradizem seu pensamento (GARG; HACKAM; TONELLI, 2008).

Segundo Biolchini et al. (2005), para a abordagem tradicional de revisão de literatura, variações entre os estudos tendem a representar uma fonte de ruído, ou seja, um fator que atrapalha a interpretação e julgamento dos resultados. Para as revisões sistemáticas, a variedade de resultados é um agente estimulante para entender o cenário completo do problema que está sendo investigado, possibilitando aos pesquisadores avaliar as influências de cada estudo individual.

Devido à baixa qualidade das revisões narrativas (DYBÅ; DINGSØYR, 2008b) e visando reduzir erros e distorções de opinião, surge a necessidade de criar revisões que divulgam claramente os resultados, obtidos de acordo com métodos rigorosos e explícitos (DYBÅ; DINGSØYR, 2008b). Com o objetivo de satisfazer tal necessidade, surgem as revisões sistemáticas.

Como já mencionado, uma revisão sistemática baseia-se em um processo para identificar, de forma abrangente, todos os estudos relacionados a uma questão específica, avaliar a metodologia de cada estudo, resumir os dados, identificar razões para divergências e citar as limitações do conhecimento atual. Todas as decisões utilizadas para compilar as informações são explicitadas, permitindo ao leitor avaliar a qualidade do processo de revisão e do potencial viés nas conclusões (GARG; HACKAM; TONELLI, 2008). A Tabela 2.1 apresenta uma comparação entre revisões sistemáticas e revisões narrativas, facilitando a compreensão.

Kitchenham et al. (2004) defendem ainda que uma revisão sistemática deve, idealmente, ser realizada por mais de um pesquisador e que um processo de verificação cruzada esteja incluído no protocolo, permitindo que pesquisadores independentes analisem e validem o trabalho de seus colegas, reduzindo o risco de vieses.

2.1.2 Origem e Expressividade das Revisões Sistemáticas

A busca por evidência para suportar decisões não é algo novo. Ainda que possa parecer algo recente, os primeiros indícios formais de síntese e avaliação sistemática de resultados científicos datam da década de 1970. Inicialmente, a ideia de tomar decisões

Tabela 2.1: Diferença entre revisões narrativas e sistemáticas.

Item	**Revisão Narrativa**	**Revisão Sistemática**
Questão	Escopo Amplo	Escopo Específico
Fontes e Método de Busca	Geralmente não especificado, com potencial viés	Fontes abrangentes e estratégia de busca explícita
Seleção	Geralmente não especificado, com potencial viés	Baseada em critérios aplicados uniformemente
Avaliação	Variável	Criteriosa e reprodutível
Síntese	Geralmente qualitativa	Geralmente Quantitativa (Meta-análise)
Inferências	Às vezes baseadas em evidências	Totalmente baseadas em evidências

baseadas em provas científicas foi aplicada à área médica, quando Archie Cochrane, em 1972, orientou os profissionais de saúde a praticarem a medicina baseada em evidências (EPPI CENTRE, 2016). Era necessário modificar a forma como eram feitas a seleção e captação dos estudos, visando melhorar a qualidade das ações de saúde e do ensino (TORRE-UGARTE; GUANILO; BERTOLOZZI, 2011). Sackett et al. (1996) definiu a medicina baseada em evidências como *"o uso consciente, explícito e criterioso das melhores evidências atuais na tomada de decisão sobre o cuidado de pacientes individuais"*.

No início do ano de 1980, em Oxford, um grupo de pesquisadores de serviços em saúde criou as condições iniciais para suportar a medicina baseada em evidências, criando um programa de revisões sistemáticas sobre a eficácia das intervenções em saúde. Tratava-se do princípio da Colaboração Cochrane, cujo centro foi inaugurado em Oxford em 1992 (EPPI CENTRE, 2016). Atualmente, a Colaboração Cochrane possui mais de 37 mil membros em 130 países e se autodefine como uma rede global independente de pesquisadores, profissionais, pacientes e pessoas em geral interessadas em saúde, trabalhando juntos para produzir informação confiável e acessível sobre saúde, transformando a maneira de tomar decisões (COCHRANE, 2016). No Brasil, o Centro de Colaboração Cochrane foi inaugurado em São Paulo no ano de 1996.

Observar a aplicabilidade das revisões sistemáticas na medicina e, por conseguinte, compreender seu mérito é relativamente simples. Em algumas áreas terapêuticas, principalmente naquelas relacionadas ao uso de medicamentos, existem diversos experimentos tentando responder questões parecidas em relação à efetividade clínica dos efeitos. Por exemplo, "Esse tratamento produz benefícios significantes quando aplicado a

esse grupo de pacientes?". O problema começa em considerar toda a literatura espalhada por periódicos ao longo de vários anos e agrava-se quando os experimentos apresentam grande nível de incerteza ou informam resultados totalmente conflitantes, ou seja, um experimento diz que determinado medicamento produziu melhoras e outro ensaio indica que o mesmo medicamento causou danos aos pacientes (CROMBIE; DAVIES, 2009).

Segundo Gouth et al. (2012), *"a análise cuidadosa das informações revelou enormes lacunas no conhecimento. Mostrou também que as chamadas melhores práticas foram, algumas vezes, mortalmente imperfeitas. E, ao não fazer nada além de peneirar metodicamente dados pré-existentes, tem salvado muito mais vidas do que podemos imaginar"*.

Logo que se percebeu os benefícios das revisões sistemáticas no campo da saúde, surgiu então, em 1999, a Colaboração Campbell com sede em Londres, adaptando a metodologia da Colaboração Cochrane para garantir o mesmo nível de qualidade de revisões sistemáticas no contexto de políticas públicas (EPPI CENTRE, 2016). A partir daí, diversas outras áreas passaram a usar abordagens similares como forma de melhorar a qualidade de suas pesquisas, tais como nutrição, políticas sociais e educação (KITCHENHAM; DYBA; JORGENSEN, 2004)

2.1.3 Vantagens e Desvantagens das Revisões Sistemáticas

A prática de revisões sistemáticas como forma de pesquisa apresenta diversas vantagens e, também, algumas desvantagens. Ao encontrar, avaliar e sumarizar todas evidências disponíveis sobre uma questão específica, uma revisão sistemática pode fornecer um nível muito mais alto de confiabilidade do que seria obtido avaliando qualquer um dos estudos primários separadamente (STAPLES; NIAZI, 2007b). Segundo o Ministério da Saúde (2012), as revisões sistemáticas permitem solucionar controvérsias em estudos com divergências nas estimativas, identificam a necessidade de realizar estudos maiores e definitivos, aumentam o poder estatístico e respondem questões não abordadas individualmente nos trabalhos primários.

Devido a sua natureza formal, metodológica e transparente, revisões sistemáticas tem uma papel fundamental em determinar se os achados científicos são consistentes e podem ser generalizados, o que tem enorme importância na área de saúde. Pesquisadores usam revisões sistemáticas para identificar, justificar e redefinir hipóteses, além de evitar pontos de falhas de trabalhos anteriores (MULROW, 1994). Em caso de inconsistências

nos resultados dos estudos cobertos pela revisão, razões podem ser encontradas e novas suposições podem ser feitas para os subgrupos (GREENALGH, 1997). Outro ponto importante é que, sem revisões sistemáticas, os pesquisadores podem iniciar trabalhos desnecessários para responder questões que já foram solucionadas ou podem perder pistas importantes a respeito de um determinado fenômeno (COOK; MULROW; HAYNES, 1997b).

Se combinarmos os benefícios da metanálise, as revisões sistemáticas ganham mais força. Normalmente, experimentos realizados com amostras pequenas tem pouca representatividade e tendem a ser inconclusivos, ou seja, não mostram grandes diferenças estatísticas entre os subgrupos. A metanálise permite agregar os frutos de pequenos experimentos, clarificando os efeitos, em média, de determinado tratamento em um subgrupo específico (CROMBIE; DAVIES, 2009). Então, ao combinar quantitativamente os resultados de diversos pequenos estudos, a metanálise permite conclusões mais convincentes e precisas (COOK; MULROW; HAYNES, 1997b).

Por outro lado, é inegável que produzir uma revisão sistemática requer muito mais esforço do que a criação de revisões literárias tradicionais (STAPLES; NIAZI, 2007b), principalmente porque, frequentemente, envolve mais de uma pessoa. Além disso, o resumo oferecido por uma revisão sistemática é tão confiável quanto o método utilizado para estimar os efeitos em cada estudo primário (GARG; HACKAM; TONELLI, 2008). Apenas adicionar o termo "revisão sistemática" ao título de uma publicação não garante que a mesma foi desenvolvida com o devido rigor (YUSUF, 1997).

2.2 Revisões Sistemáticas em Engenharia de Software

Nos últimos anos, diversas pesquisas têm sido realizadas na área de engenharia de software, ao mesmo tempo em que as técnicas e conceitos vêm sendo aprimorados consideravelmente (BIOLCHINI et al., 2005). Contudo, é muito comum que a construção de determinado software seja suportada por uma tecnologia para a qual existem poucas ou nenhuma evidência sobre sua eficácia (ZELKOWITZ; WALLACE; BINKLEY, 2003). Mais do que isso, algumas vezes não se sabe, ao menos, se aquela tecnologia é minimamente adequada ao contexto do projeto. Zelkowitz et al. (2003) complementam ainda que outras tecnologias, por sua vez, são ignoradas mesmo existindo dados publicados evidenciando que elas serão úteis. A adoção de uma técnica ou tecnologia inadequada representa enormes riscos e custos.

A revisão sistemática de literatura, de maneira similar àquela aplicada em outras ciências, pode ser a chave para solucionar tais problemas. Observando os benefícios que a abordagem baseada em evidências trouxe à medicina, Kitchenham et al. (2004) sugeriram que os profissionais adotassem a Engenharia de Software Baseada em Evidências (EBSE) como mecanismo para suportar e melhorar suas decisões quanto à adoção de tecnologias. A principal motivação para adotar a prática baseada em evidências é que sistemas intensivos de software tem se tornado parte do cotidiano das pessoas, o que traz diversos benefícios, mas também pode provocar danos graves. Desta forma, Kitchenham et al. (2004) afirmam que é preciso adotar controles para minimizar os riscos de tais prejuízos.

Sob total influência da medicina baseada em evidência, cujo princípio é *"agregar as melhores evidências de pesquisa com experiência clínica e com valores dos pacientes"* (SACKETT et al., 2000), a engenharia de software baseada em evidências tem por objetivo *"prover os meios pelos quais as melhores evidências atuais de pesquisa possam ser integradas com experiência prática e com valores humanos no processo de tomada de decisão sobre desenvolvimento e manutenção de software"* (KITCHENHAM; DYBA; JORGENSEN, 2004). De acordo com os mesmos autores, a engenharia de software baseada em evidências deve ainda contemplar os seguintes a seguir:

- criar um objetivo comum para grupos de pesquisa e para pesquisadores individuais, para garantir que suas pesquisas tenham como alvo a necessidade da indústria e dos grupos interessados;
- fornecer uma forma que permita aos profissionais tomar decisões racionais sobre a adoção de tecnologias;
- prover meios para aumentar a confiança de sistemas intensivos de software, como resultados da melhor escolha das tecnologias de desenvolvimento;
- fornecer mecanismos que facilitem a aceitação de sistemas de software que interagem com pessoas;
- conceder uma entrada ao processo de certificação.

Observando os itens acima, é possível notar o quanto a engenharia de software baseada em evidências preocupa-se em reduzir a distância entre o mundo acadêmico e a realidade da indústria, o que é um grande desafio. A ideia é impulsionar a ênfase no rigor metodológico, tendo como foco a pertinência à aplicação prática (DYBÄ; KITCHENHAM; JORGENSEN, 2005). Além disso, da mesma forma que a medicina baseada em evidências é embasada por rigorosos estudos de tratamentos dados a pacientes reais, a en-

genharia de software não deve confiar somente em experimentos de laboratório, tentando obter o máximo de evidências de projetos industriais por meios de estudos de observação e experiências em campo (DYBÄ; KITCHENHAM; JORGENSEN, 2005).

Para obter as melhores evidências é necessário realizar revisões sistemáticas (BIOLCHINI et al., 2005). A engenharia de software baseada em evidências herdou grande parte do modelo aplicado à medicina. Contudo, há diferenças consideráveis entres as duas áreas, o que representa desafios à prática na engenharia de software. Um primeiro ponto, observado por Kitchenhan et al. (2004) diz respeito à cooperação entre os médicos, o que é mais difícil de encontrar entre empresas de software concorrentes. Na área da saúde, por exemplo, médicos são responsáveis por divulgar os efeitos de determinado medicamento, o que contribui para a evidência associada a um tratamento e pode gerar novos estudos primários. Na engenharia de software não há incentivo para que as empresas de software auxiliem umas às outras, informando suas boas e más experiências acerca de novas tecnologias.

Outra grande diferença da área médica é que, na engenharia de software, a maior parte das tarefas deve ser realizada por profissionais experientes e capacitados que conhecem as técnicas e métodos que estão sendo usados, o que requer a presença efetiva dos mesmos. Na medicina, ainda que os médicos sejam experientes, os tratamentos prescrevidos por eles aos pacientes podem ser aplicados com sucesso por enfermeiros, o que não exige a presença dos próprios médicos. O fato de exigir a presença de um engenheiro de software capacitado impede o mascaramento dos praticantes do estudo. É comum na área de saúde a realização de experimentos randômicos, nos quais nem o médico, nem o paciente sabem qual tratamento está sendo aplicado, visando eliminar as possibilidades de pressionar os resultados. O mesmo não pode ser realizado nos experimentos de engenharia de software (BIOLCHINI et al., 2005).

O ciclo de vida de um software é impactado por diversas técnicas que interagem entre si. Por exemplo, métodos de projeto dependem de uma prévia análise de requisitos, que leva em conta limitações impostas pelas plataformas de software e hardware, bem como restrições de tempo e orçamento. Dessa forma, é complicado afirmar que a técnica de projeto tem um impacto significante na confiabilidade do produto final. Esse é outro fator que afasta os estudos da engenharia de software dos trabalhos de medicina. É difícil determinar o impacto individual de uma técnica no resultado final do produto (KITCHENHAM; DYBA; JORGENSEN, 2004).

Devido às peculiaridades do contexto de engenharia de software, Kitchenhan et

al. (2004) propuseram orientações para desenvolver revisões sistemáticas adequadas à realidade dos profissionais e pesquisadores da área. Tais instruções são divididas em três fases e tem forte influência do modelo praticado na medicina. Tais fases estão descritas a seguir.

- **Planejar a revisão:** o resultado desta etapa é o protocolo da revisão sistemática, o qual definirá o propósito e todos os passos da revisão. Essa fase é subdividida em 3 partes:
  - *Identificar a necessidade de realizar a revisão sistemática:* primeiramente, é necessário observar se existe uma real necessidade de fazer uma nova revisão sistemática, pois o esforço requerido é grande. Antes de começar uma nova revisão, deve-se tentar encontrar e analisar todas revisões existentes sobre o fenômeno de interesse. Assim, pode-se encontrar alguma revisão que já atende à necessidade ou que forneça exemplos que podem ajudar no desenvolvimento do protocolo da nova revisão.
  - *Definir as questões de pesquisa:* essa é a parte mais importante de uma revisão sistemática, porque são as questões que guiam toda a metodologia da revisão. Uma questão apropriada deve ser importante tanto para pesquisadores quanto para profissionais da indústria, permitindo guiar as mudanças nas práticas atuais ou aumentar a confiabilidade das mesmas. Além disso, deve tentar identificar discrepâncias entre crenças comuns e a realidade. Outro ponto que deve ser definido é a estrutura da questão, que se refere aos seguintes critérios:
    - População: refere-se a uma área de aplicação, um segmento da indústria, uma categoria de engenheiro de software (relativo à experiência do profissional) ou um cargo (gerente, por exemplo);
    - Intervenção: refere-se à metodologia de software, determinada ferramenta ou tecnologia;
    - Comparação: define o que (tecnologia, ferramenta ou técnica) está sendo comparado com a intervenção;
    - Efeitos: deve relatar os fatores mais importantes para os pesquisadores e profissionais, tais como melhora na confiabilidade, redução de custo e tempo de desenvolvimento;
    - Contexto: define em que contexto a comparação está sendo realizada, por exemplo, academia ou indústria.

  - *Desenvolver o protocolo da revisão:* é necessário ter um protocolo pré-estabelecido contendo um plano completo para a execução da revisão sistemática. Tal protocolo é necessário para evitar vieses dos pesquisadores. O protocolo pressupõe a estrutura do relatório final, descrevendo o contexto, as questões específicas de pesquisa, a estratégia de busca dos trabalhos primários (incluindo os termos de busca e as bases de dados a serem pesquisadas), o critério de seleção dos estudos primários a serem incluídos ou excluídos da revisão, o método de avaliação dos estudos primários, a estratégia de extração de dados, entre outros.
  - *Revisar o protocolo:* como o protocolo tem extrema importância, ele mesmo deve ser revisado. Se possível, a revisão do protocolo deve ser feita por outros pesquisadores.

- **Conduzir a revisão:** nesta fase, são gerados os resultados intermediários e finais da revisão. Após a aprovação do protocolo, a revisão pode começar de fato, seguindo as seguintes etapas.

  - *Identificar estudos:* uma estratégia de busca formal, descrita no protocolo, é usada para encontrar a população de estudos que pode ser relevante para as questões de pesquisa. É necessário ter a descrição explícita da estratégia de busca utilizada, permitindo que a mesma seja replicada. Para evitar vieses, deve-se tentar encontrar publicações que reportam efeitos negativos.
  - *Selecionar os estudos primários:* identificar os estudos primários que fornecem evidência direta para as questões de pesquisa. O processo de seleção deve estar descrito no protocolo.
  - *Avaliar a qualidade dos estudos:* depois de selecionar os estudos primários, é necessário realizar uma avaliação mais criteriosa quanto à qualidade, permitindo aos pesquisadores identificar diferenças entre as implementações dos trabalhos. Além disso, procura-se identificar vieses nos resultados dos trabalhos, o que caracteriza um fator de exclusão.
  - *Extração de dados:* a extração dos dados produzidos nos estudos primários também deve estar documentada no protocolo. Devem ser coletados todos os dados necessários para responder às questões de pesquisa. Essa etapa é um pré-requisito para a realização de metanálise.
  - *Síntese dos dados:* depois que os dados foram extraídos, eles devem ser agru-

pados e resumidos a fim de clarificar as questões de pesquisa da revisão sistemática. Novamente, essa etapa deve estar descrita no protocolo.

- **Reportar a revisão:** após concluir a revisão, ela deve ser publicada. Geralmente, as revisões sistemáticas são publicadas na forma de relatórios técnicos ou artigos de periódicos ou conferências. Se os resultados da revisão sistemática forem de interesse dos profissionais da indústria, é interessante que outros meios de disseminação sejam examinados.

Observando o número de passos recomendados para a realização de uma revisão sistemática, seja em engenharia de software ou em qualquer outra área, nota-se o quão criteriosa e, consequentemente, trabalhosa é a sua condução. É por isso que a qualidade da revisão sistemática reside nos métodos que foram utilizados para conduzi-la. Nota-se também a constante preocupação em evitar vieses nos resultados, o que não ocorre nas revisões narrativas.

2.3 Trabalhos Relacionados

Com a popularidade das revisões sistemáticas em engenharia de software, muitos trabalhos vêm sendo realizados com a intenção de aprimorar os métodos já existentes e dar suporte à condução das revisões. Marshall et al. (2014) realizaram uma análise das ferramentas existentes para suportar todo o processo envolvido em uma revisão sistemática de engenharia de software (MARSHALL; BRERETON; KITCHENHAM, 2014). Foram identificadas quatro ferramentas que cobrem a condução de revisões sistemáticas em engenharia de software.

- *Systematic Literature unified Review Program (SLuRp)*: trata-se de uma ferramenta *web* desenvolvida para auxiliar o gerenciamento de uma revisão sistemática. Todas as fases de desenvolvimento da revisão são suportadas e resultados confiáveis podem ser gerados (BOWES; HALL; BEECHAM, 2012).
- *State of the Art through systematic review (StArt)*: objetiva suportar cada estágio da realização da revisão sistemática, facilitando sua produção (HERNANDES et al., 2012).
- *SLR-Tool*[1]: disponibilizada gratuitamente, também objetiva suportar cada estágio do desenvolvimento de uma revisão sistemática. Além disso, algumas funcionali-

[1] http://alarcosj.esi.uclm.es/SLRTool/

dade adicionais são suportadas, tais como exportação de metadados.

- *SLRTool*[2]: plataforma *web* e de código aberto, suporta o processo de condução de uma revisão sistemática por múltiplos usuários e não limita-se ao escopo de engenharia de software.

Outro trabalho relacionado é o estudo realizado por Kitchenham et al. (2009), o qual consiste em uma revisão sistemática de revisões sistemáticas de engenharia de software, isto é, trata-se de um estudo terciário com o objetivo de revisar todos os artigos relacionados à engenharia de software baseada em evidências. Foram analisados artigos publicados no período de 2004 a 2008, concentrando-se naqueles que descrevem revisões sistemáticas (KITCHENHAM et al., 2009). Tal trabalho merece destaque porque avalia o impacto da prática de revisões sistemáticas como forma de agregar evidências. Os autores concluem que a qualidade das revisões sistemáticas, no período em questão, estava aumentando, porém havia poucos tópicos cobertos até então.

Embora existam estudos desenvolvidos para avaliar a qualidade das revisões sistemáticas e ferramentas importantes criadas para suportá-las, encontrar aquelas que já foram produzidas ainda é uma tarefa difícil. Biolchini et al. (2005) evidenciam que uma diferença notável entre engenharia de software e a medicina é que os pesquisadores da saúde contam com uma grande estrutura organizacional e tecnológica para ampará-los. Além da já mencionada *Colaboração Cochrane*, existem outras grandes bases de dados de revisões sistemáticas que auxiliam a criação de novos estudos e divulgam o conhecimento adquirido, como é o caso do serviço *PubMed Health*[3], da biblioteca nacional de medicina dos Estados Unidos.

Conforme discutido na Seção 2.2, o primeiro passo na elaboração de uma revisão sistemática é verificar se ela é realmente necessária, o que envolve investigar se um estudo com o mesmo objetivo já foi desenvolvido. Infelizmente, não existe uma ferramenta que permita encontrar facilmente as revisões sistemáticas produzidas em engenharia de software (e na computação como um todo), dificultando o trabalho dos pesquisadores e ocultando conhecimentos que poderiam ser muito bem aproveitados. Os estudos de engenharia de software baseados em evidencia estão fragmentados e limitados (KITCHENHAM; DYBA; JORGENSEN, 2004).

[2]http://www.slrtool.org/

[3]http://www.ncbi.nlm.nih.gov/pubmedhealth/

2.4 Considerações Finais

Baseando-se no benefício que os repositórios de revisões sistemáticas trazem aos pesquisadores da área médica e, diante da ausência de ferramentas similares para apoiar as pesquisas de engenharia de software, propõe-se a criação de um mecanismo análogo para revisões sistemáticas de computação, com ênfase inicial para àquelas produzidas no contexto de engenharia de software devido à sua maior popularidade. O projeto visa acomodar revisões sistemáticas de toda a computação porque acredita-se que a quantidade de tais estudos tende a aumentar nas demais áreas da informática. Além de revisões sistemáticas, o sistema suportará também a catalogação de qualquer revisão ligada à computação, isto é, revisões tradicionais também serão suportadas pelo repositório.

As bases de dados de trabalhos de computação disponíveis atualmente cumprem muito bem o papel de armazenar os estudos desenvolvidos, entretanto, não há nenhuma distinção entre trabalhos primários e revisões sistemáticas. E é esse um dos maiores problemas, porque os mecanismos de busca disponíveis não permitem aplicar filtros que retornem apenas revisões sistemáticas. Todos os trabalhos são considerados estudos primários. O sistema proposto foi desenvolvido para suportar qualquer publicação que seja uma revisão de literatura, mas há uma diferenciação entre os tipos de publicações. Isso permite que um tratamento especial seja dado às revisões sistemáticas, permitindo que os mecanismos de busca possam selecionar os resultados de acordo com os metadados mais relevantes.

O sistema foi projetado para atender às principais necessidades de quem deseja pesquisar por revisões sistemáticas e divulgar suas publicações. Isso facilita o trabalho de encontrar uma revisão sistemática específica, verificar o que já foi desenvolvido sobre certo tópico e permite que novas revisões sejam divulgadas de forma fácil, direcionadas a um público que busca este tipo de trabalho. Assim, tanto quem busca o conhecimento, quanto quem o produz são beneficiados.

3 LRDB: FUNCIONALIDADES E ARQUITETURA

O LRDB, sigla que em inglês denota *Literature Reviews Database*, é um sistema *web* que permite o cadastro de publicações relacionadas à computação, oferecendo suporte particular ao cadastro de revisões sistemáticas da área. Inicialmente, o sistema foi idealizado para contemplar somente revisões sistemáticas, mas depois optou-se por aproveitar a plataforma para abrigar revisões tradicionais também. Neste capítulo, serão apresentadas as principais funcionalidades do LRDB, bem como sua arquitetura e tecnologias utilizadas.

3.1 Funcionalidades

As principais funcionalidades da aplicação compreendem cadastrar e pesquisar publicações, sendo possível filtrar os resultados de acordo com os termos de interesse. Além disso, o sistema proporciona um ambiente colaborativo, onde usuários podem editar publicações de outros usuários, com o intuito de manter os dados corretos e atualizados. Administradores avaliam as sugestões de edição, podendo aprová-las ou não. Para cada publicação, é preciso informar se a mesma trata-se de uma revisão sistemática ou não, o que permite realizar buscas considerando apenas revisões sistemáticas.

A seguir, apresentamos os recursos do sistema com um nível maior de detalhes.

- *Tipos de usuários:* o sistema possui três tipos de usuários, sendo que qualquer pessoa pode se cadastrar para ter acesso a um maior número de recursos. Os usuários visitantes, isto é, não cadastrados, podem apenas visualizar as publicações mais recentes, bem como realizar pesquisas simples e avançadas. Já os usuários cadastrados podem adicionar novas publicações, além de solicitar edição e exclusão de registros. O terceiro grupo de usuários é composto pelos administradores do sistema, o quais possuem a missão de avaliar todas solicitações de novas publicações, edição ou exclusão. Sempre que um usuário cadastrado insere uma nova publicação no sistema, a mesma é submetida à aprovação de um administrador. O mesmo ocorre para os pedidos de edição e exclusão.
- *Cadastro de Publicações:* Usuários cadastrados e administradores podem inserir novas publicações. Ao cadastrar uma publicação, é necessário informar previamente se o estudo é uma revisão sistemática e também qual o tipo da publicação

(artigo, livro, relatório técnico, entre outros). Praticamente todos os tipos de entrada de bibliografia da ferramenta *BibTeX* são suportados. Ao selecionar o tipo de publicação, o usuário é redirecionado para uma página onde deve preencher os campos associados aos metadados do trabalho, tais como título, classificação e autores. É necessário informar uma URL (*Uniform Resource Locator*) ou DOI (*Digital Object Iditifier*[1]), que permitirá acessar a publicação completa. Ao concluir a inserção da nova publicação, a mesma é submetida à aprovação dos administradores, tornando-se pública aos demais usuários somente se for aprovada.

- *Edição de Publicações:* Usuários cadastrados e administradores podem editar publicações. A ideia é que todos contribuam para que tenhamos uma plataforma mais rica e confiável. Assim, ao se deparar com algum dado incorreto ou que não foi informado, é possível editá-lo e solicitar a edição da publicação. O pedido será recebido pelos administradores, que poderão comparar a nova versão com a antiga e então autorizar a modificação.
- *Exclusão de Publicações:* Usuários cadastrados e administradores podem excluir publicações. Se um usuário perceber que um trabalho não deveria ter sido publicado, porque, por exemplo, sua temática não está ligada à computação, ele pode solicitar que o mesmo seja removido do sistema. Ao requisitar a exclusão, é necessário fornecer uma justificativa. Administradores avaliarão o argumento e aplicarão a ação adequada.
- *Busca:* Qualquer usuário, incluindo visitantes podem realizar buscas. A busca simples retorna todas as publicações que possuem qualquer dado associado ao termo de busca.
- *Busca Avançada:* Qualquer usuário, inclusive visitantes podem utilizar a ferramenta de busca avançada. É possível filtrar os resultados informando os autores desejados, palavras-chave, classificação, entre outros. É possível especificar também se a pesquisa deve retornar apenas revisões sistemáticas.
- *Atualização de dados cadastrais:* Usuários cadastrados tem acesso a uma área onde podem atualizar seus dados cadastrais. É possível alterar nome, sobrenome, senha, entre outros. Caso o usuário esqueça sua senha de acesso ao sistema, ela também pode ser recuperada.
- *Gerência de Usuários:* Administradores possuem acesso ao módulo de gerência de usuários. É possível excluir um usuário cadastrado do sistema ou conceder-lhe

[1] https://www.doi.org/

direitos administrativos. Para facilitar, existe uma ferramenta simples de pesquisa de usuários.

Além dos itens apresentados, outro ponto que merece destaque é a relação entre usuários e autores. Um usuário pode cadastrar trabalhos que não são de sua autoria, isto é, pode popular o repositório com qualquer publicação, mediante aprovação. Assim, no momento em que uma pessoa realiza o cadastro no sistema, é verificado se existem publicações associadas a ela, ou seja, de sua autoria. Caso existam, todos os trabalhos são exibidos como sendo suas próprias publicações. Isso permite que o verdadeiro autor seja notificado sempre que um novo estudo seu for adicionado por um outro usuário. Além disso, caso o usuário tenha publicado trabalhos com e-mails diferentes, situação comum quando o mesmo pertence a mais de uma instituição, é possível informar endereços de e-mail adicionais, permitindo ao sistema associar todos os trabalhos a um único usuário.

3.2 Arquitetura e Tecnologias Utilizadas

A arquitetura da aplicação, bem como as tecnologias empregadas, foram arranjadas de forma a contemplar as melhores práticas de desenvolvimento e manter os padrões mais utilizados. Por possuir experiências prévias, optou-se por adotar o conjunto de tecnologias provido pela plataforma *.NET*[2] e todos os conceitos que são empregados como padrão por desenvolvedores experientes em tal plataforma.

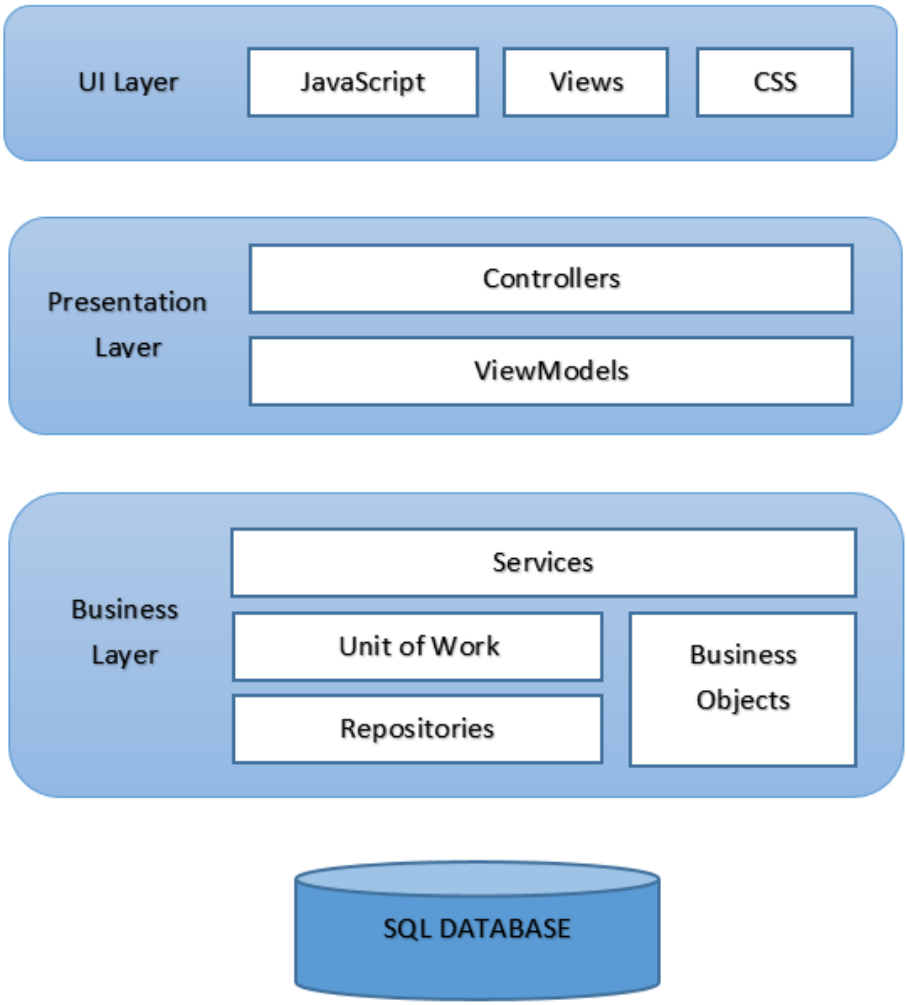
3.2.1 Arquitetura

A arquitetura da aplicação está divida em três camadas, conforme ilustrado na Figura 3.1. A primeira camada representa a interface com o usuário, isto é, a porção do sistema que é visível de fato para quem o utiliza. A camada de apresentação, por sua vez, engloba toda manipulação necessária para produzir os insumos necessários à interface com o usuário, além de tratar suas requisições. A camada de negócios é o núcleo da aplicação, envolvendo todas as entidades e regras de negócio, além de controlar o acesso aos dados.

O projeto foi desenvolvido de acordo com o padrão arquitetural MVC, sigla que em inglês significa *Model-View-Controller*. É um padrão que incentiva o isolamento entre

[2]https://www.microsoft.com/net

Figura 3.1: Arquitetura do sistema



as partes da aplicação, isto é, favorece um baixo acoplamento (CHADWICK; SNYDER; PANDA, 2012). Desta forma, a aplicação é decomposta em três módulos: modelo, visão e controle, descritos abaixo.

- *Modelo:* representa o núcleo da aplicação. É responsável por gerenciar o comportamento e os dados do modelo de domínio. Engloba toda a lógica de negócio, validações e acesso a dados.
- *Visão:* é a camada responsável pela exibição das informações ao usuário. Realiza formatações e pequenos ajustes relacionados apenas à maneira como os dados serão apresentados. Nenhuma decisão de negócio deve ser tomada nesse nível da aplicação.
- *Controle:* é uma camada intermediária que controla o fluxo da aplicação. É respon-

sável por receber as requisições dos usuários e informá-las à camada de modelo. Além disso, deve fornecer respostas ao usuário, definindo o que será exibido na camada de visão.

Os principais benefícios do padrão MVC, que também motivaram sua escolha para o presente trabalho, derivam do fato de que este padrão arquitetural induz um fraco acoplamento. Chadwick et al. (2012) afirmam que as maiores vantagens estão no desenvolvimento, testabilidade e manutenção. O desenvolvimento é facilitado porque um componente não depende diretamente dos demais, o que permite que ele seja desenvolvido isoladamente ou substituído sem afetar o desenvolvimento de outro elemento com o qual interage. Os testes também são beneficiados pelo mesmo motivo, isto é, pode-se trocar um componente inteiro por algo que o simule, permitindo testar outras áreas da aplicação. A manutenção, por sua vez, leva vantagem porque mudanças geralmente afetam um pequeno grupo de elementos isolados.

A Figura 3.1 mostra a arquitetura em camadas, mas nela também podemos observar os elementos do padrão MVC. Na verdade, é apenas uma forma de ilustrar como a arquitetura da aplicação está organizada. Assim, o módulo visão do MVC corresponde à camada mais acima no diagrama da Figura 3.1, que equivale, em termos práticos, a páginas *HTML* (*HyperText Markup Language*) com formulários e botões, através dos quais o usuário informa dados e requisita ações. Essas páginas são chamadas de *views*, pois representam a visão do usuário. Além de *HTML*, que define somente o conteúdo das páginas, na camada de interface de usuário existem outras tecnologias, tais como *CSS* (*Cascading Style Sheets*), cuja função é definir a aparência dos elementos *HTML* e *JavaScript*, cujo objetivo é alterar o comportamento das páginas dinamicamente, evitando que as mesmas sejam recarregadas.

Já os controladores do MVC aparecem na camada de apresentação da arquitetura ilustrada e estão assim localizados porque são os responsáveis por tratar as entradas dos usuários, informadas na camada de cima, notificar à camada de negócios e devolver uma resposta adequada ao usuário. São os controladores que decidem o que será exibido ao usuário e como suas requisições devem ser tratadas. É por isso que, como o próprio nome sugere, controlam o fluxo da aplicação.

Na camada de apresentação existem também as *viewmodels*. A função das *viewmodels* é representar um conjunto de dados que serão utilizados por uma *view* para exibir determinadas informações (PIRES, EDUARDO, 2016). Isso é particularmente útil quando queremos exibir as informações dos objetos de domínio de uma maneira mais

amigável ou mais completa. Essa é a maior vantagem do uso das *viewmodels*: não é necessário alterar uma classe de domínio para atender as necessidades de uma *view*. (PIRES, EDUARDO, 2016). Além disso, uma *viewmodel* pode conter dados de mais de uma classe de domínio, oferecendo tudo que uma *view* precisa em uma única requisição. Em geral, os controladores recebem os dados brutos da camada de negócios e os convertem no formato de *viewmodel*, fornecendo o que a *view* associada necessita para exibir a resposta ao usuário.

A camada de negócios apresentada faz analogia ao *model* do padrão MVC. É nesta parte que se encontra toda lógica da aplicação. Todas decisões, bem como o processamento de informações, são realizados nesse nível. Como mencionado anteriormente, os controladores recebem as requisições dos usuários e as informam à camada de negócios. Em um nível mais detalhado, os controladores requisitam serviços da camada de negócios. Neste ponto, temos a camada de serviços, a qual fornece um conjunto de operações disponíveis às camadas superiores e encapsula a lógica de negócios da aplicação, isto é, a camada de serviços é um padrão para organizar a lógica de negócios (FOWLER, 2002). Se o usuário solicita uma ação que é composta por diversas operações, os controladores podem acionar um único método da camada de serviços, que por sua vez tratará toda sequência de operações envolvidas naquela ação e retornará uma resposta ao controlador. Desta maneira, o código dos controladores fica claro, enxuto e facilmente compreensível.

Abaixo da camada de serviços, temos os objetos de negócio. Todas as classes que representam as entidades modeladas na aplicação são consideradas objetos de negócio. Por exemplo, temos classes que representam artigos, livros e autores. Os serviços são responsáveis pela lógica de negócio que envolve esses objetos, por exemplo, eles coordenam o processo de publicação de um novo artigo. Além disso, utilizamos um *framework* de mapeamento objeto-relacional para mapear os objetos de negócio para tabelas no banco de dados, o que nos permite focar mais nas regras de negócio da aplicação do que na persistência dos objetos propriamente dita, produzindo um código mais reduzido, elegante e de fácil manutenção. (RANIERI, BARBARA, 2016).

Na camada de negócios, temos ainda outros dois elementos relacionados à persistência de dados: repositórios e unidade de trabalho. O padrão repositório adiciona uma abstração no topo da camada de persistência e ajuda a eliminar lógica duplicada na implementação de códigos para consultas ao banco de dados (MARCORATTI, JOSE CARLOS, 2016a). Um repositório encapsula a lógica necessária para persistir os objetos de domínio, fornecendo uma visão mais orientada a objetos da camada de persistência e

a separação de interesses no código (FOWLER, 2002). Na prática, existe um repositório para cada entidade que deve ser persistida no banco de dados. Por exemplo, podemos ter um repositório de usuários, outro repositório de publicações e assim por diante. Cada repositório engloba todas as operações de persistência associadas à entidade a qual se refere. Como muitas das operações de persistência possuem praticamente a mesma função em repositórios diferentes, tais como inserção ou deleção, utilizamos o conceito de repositório genérico. A principal ideia de um repositório genérico é evitar redundância de código e centralizar a lógica de acesso aos dados. Dessa forma, cada entidade do modelo de domínio utiliza uma instância do repositório genérico.

O padrão unidade de trabalho tem como propósito combinar um conjunto de interações e aplicar todas as mudanças no banco de dados em apenas uma transação (TECHBRIJ, 2016). *"Uma unidade de trabalho pode ser vista como um contexto, sessão ou objeto que acompanha as alterações das entidades de negócio durante uma transação, sendo também responsável pelo gerenciamento dos problemas de concorrência que podem ocorrer oriundos dessa transação"* (MARCORATTI, JOSE CARLOS, 2016b). A camada de serviços acessa os repositórios através da unidade de trabalho. Dessa forma, é possível garantir que todas as operações que envolvem múltiplos repositórios são realizadas usando a mesma instância do contexto e, portanto, são efetivadas em uma única transação.

Ainda que a maioria dos *frameworks* de mapeamento objeto-relacional implementem nativamente algum mecanismo semelhante ao provido pela unidade de trabalho e pelos repositórios, implementar uma unidade de trabalho e repositórios próprios é muito útil para melhorar a testabilidade do sistema, efetuar *logs* e ajustes de desempenho, além de tornar possível a substituição do *framework* de mapeamento objeto-relacional sem qualquer modificação na interface de acesso aos dados (MARCORATTI, JOSE CARLOS, 2016b) (DATATELL, 2016). Se usássemos diretamente o mecanismo fornecido pelo *framework* na camada de serviços e, por algum motivo, precisássemos que trocá-lo por outro no futuro, boa parte do código da camada de serviços teria de ser reescrito. Dessa maneira mantemos as responsabilidades bem separadas e menos dependentes de tecnologias.

3.2.2 Tecnologias

Para a implementação do sistema, optamos pela tecnologia *Microsoft ASP .NET MVC*[3], que pode ser vista como um grande *framework* para desenvolvimento de aplicações *web*, construído sobre o popular e maduro *.NET Framework* (CHADWICK; SNYDER; PANDA, 2012). O *.NET Framework* é uma enorme plataforma projetada para facilitar o desenvolvimento orientado a objetos de aplicações para a internet (LIBERTY, 2005).

O *framwork ASP .NET MVC* herda todas as vantagens do padrão arquitetural MVC, tendo como foco uma aplicação com fraco acoplamento e altamente manutenível (CHADWICK; SNYDER; PANDA, 2012), sendo flexível e customizável. As partes do *ASP.NET MVC* são projetadas com o intuito de serem facilmente alteradas ou substituídas. Todos os principais contratos da estrutura do aplicação são baseados em interfaces, o que permite a realização de testes unitários sem a necessidade de colocar os controladores em execução (CMARIX, 2016). Outra vantagem é que o *ASP .NET MVC* fornece uma integração fácil com *frameworks* de *JavaScript*, o que pode ser bastante útil no futuro (JONATHAN DANYLKO, 2016). Além disso, a título de curiosidade, o *ASP .NET MVC* possui código aberto [4] e as aplicações podem ser desenvolvidas em plataformas *Windows*, *Linux* ou *Mac*.

Por ser construído no topo do *.NET Framework*, o *ASP .NET MVC* herda também diversos recursos poderosos e essenciais ao desenvolvimento de uma aplicação *web*: envio e recebimento de mensagens *HTTP*, acesso simultâneo à aplicação por vários usuários de maneira eficiente, bem como geração dinâmica das páginas da aplicação. Outros recursos como provedores de identidade e *frameworks* de mapeamento objeto-relacional também fazem parte da plataforma *.NET*, o que fornece um ambiente completo para o desenvolvimento de aplicações *web*.

Além desses fatores, outro ponto que encorajou a escolha da tecnologia *ASP .NET MVC* foi o fato de que a mesma utiliza a linguagem de programação *C#*, com a qual já possuía experiência prévia.

Para os programadores, C# é uma linguagem de uso simples e fácil, que abstrai grande parte dos detalhes de baixo nível, permitindo que os desenvolvedores concentrem-se na programação da aplicação propriamente dita (CODEMENTOR, 2016). Atualmente,

[3] http://www.asp.net/mvc
[4] https://github.com/aspnet

C# é uma das linguagens de programação mais utilizadas no mundo (SCOTT EDWARDS, 2016).

Quanto ao ambiente de desenvolvimento, utilizamos a ferramenta *Visual Studio Community 2015* [5]. A *Microsoft* a define como *"um ambiente de desenvolvimento integrado gratuito, completo e extensível para criação de aplicativos modernos para Windows, Android*[6] *e iOS*[7]*, bem como aplicativos web e serviços de nuvem"*. O fato da ferramenta já nos ser familiar e por ser voltada para a linguagem *C#* e para o *framework .NET* (incluindo *ASP .NET MVC*) determinou sua escolha.

Para gerenciamento do código fonte, adotamos o conjunto de ferramentas *Team Foundation Server* [8], que já vem integrado ao *Visual Studio* e fornece diversos serviços, tais como controle de versão, integração contínua e ferramentas de métodos ágeis (VISUAL STUDIO, 2016b). Para controle de versão, o *Team Foundation Server* permite que sejam criados repositórios *Git* [9] gratuitos ilimitados e privados, que podem ser acessados de qualquer cliente *Git* em qualquer plataforma (VISUAL STUDIO, 2016a). Assim, para o controle de versão propriamente dito, usamos um repositório *Git* (suportado pelo *Team Foundation Server*) devido a sua grande aceitação e popularidade.

As tecnologias e ferramentas mencionadas acima permeiam por todo o projeto da aplicação. A seguir estão listadas outras tecnologias empregadas na construção do sistema, mas que atuam em apenas parte do mesmo, isto é, proveem funcionalidades específicas à partes da arquitetura.

- *ASP .NET Razor* [10]: o *Razor* é um mecanismo que permite a geração de *views* de forma simples e dinâmica, o que simplifica muito a codificação da camada de interface do usuário. (DEVMEDIA, 2016). O *Razor* facilita e agiliza a criação de código *HTML*, além de fornecer diversos comandos úteis, os quais são processados no servidor antes da página ser enviada ao navegador cliente (W3SCHOOLS, 2016).
- *Bootstrap*[11]: o *Bootstrap* é um *framework* de *HTML*, *CSS* e *JavaScript* que permite a criação de interfaces *web* de forma elegante, simples e rápida. Ele fornece um conjunto de componentes extremamente úteis como botões, tabelas e paineis,

[5] https://www.visualstudio.com/products/visual-studio-community-vs
[6] https://www.android.com/
[7] http://www.apple.com/br/ios/
[8] https://www.visualstudio.com/pt-br/products/tfs-overview-vs.aspx
[9] https://git-scm.com/
[10] https://msdn.microsoft.com/pt-br/library/gg675215.aspx
[11] http://getbootstrap.com/

permitindo criar páginas visualmente agradáveis e modernas com pouco esforço. Outra enorme vantagem é que o *Bootstrap* tem como foco produzir *websites* responsivos, isto é, um único código faz com que as páginas se adaptem a *desktops*, *smartphones* e *tablets*. O *Bootstrap* vem integrado aos projetos *ASP .NET MVC* criados com o *Visual Studio* e pode ser utilizado juntamente com *Razor*, tornando a criação das *views* ainda mais produtiva e elegante. O *Boostrap* possui código aberto e atualmente é o *framework* do gênero mais popular (SMITH, 2016).

- *Entity Framework*[12]: o *Entity Framework* é, como o próprio nome diz, um *framework* de mapeamento objeto-relacional para a plataforma *.NET*, que permite o mapeamento dos dados das tabelas relacionais para objetos de domínio (RANIERI, BARBARA, 2016). Isso elimina a necessidade de escrever a maior parte do código de acesso a dados, abstraindo a camada que teria essa responsabilidade (MICROSOFT, 2016). Observando a arquitetura apresentada, o *Entity Framework* situa-se entre os repositórios e o banco de dados *SQL*. Utilizamos a abordagem *Code First*, na qual inicia-se definindo as classes do modelo de domínio sem nenhuma preocupação com acesso a dados e sem dependência alguma do *framework*. Ao seguir algumas convenções simples, o *Entity Framework* é capaz inferir muitas informações do modelo de domínio puramente a partir do formato das classes, passando a mapeá-las em tabelas do banco de dados (LERMAN; MILLER, 2011).

  A mesma classe que é utilizada para definir os objetos do domínio da aplicação também é usada para definir a estrutura da tabela que a representa no banco de dados (KAMALASANAN, 2016). Outro benefício é que, como o banco de dados é totalmente gerado a partir do código das classes do modelo de domínio, ao termos um controle de versão do código, também temos um controle de versão do banco de dados (KAMALASANAN, 2016). O *Entity Framework* também está integrado à plataforma *.NET* e foi utilizado por ser o mais popular entre os desenvolvedores *.NET*. A título de curiosidade, o Entity Framework também possui código aberto [13].
- *ASP .NET Identity*[14]: integrante do *ASP .NET*, o *framework ASP .NET Identity* tem por objetivo facilitar a autenticação e autorização de usuários. Possui arquitetura limpa e modularizada, o que proporciona facilidade de customização e testabilidade (PIRES, 2016). Decidimos utilizar o *ASP .NET Identity* porque ele já vem integrado ao modelo de projeto padrão *ASP .NET MVC* e porque o *framework* sim-

[12] http://www.asp.net/entity-framework
[13] https://github.com/aspnet/EntityFramework
[14] http://www.asp.net/identity

plifica muito tarefas que custariam um tempo considerável para ser implementadas manualmente, como confirmação de conta via *E-mail* e recuperação de senhas. Além disso, o *framework* oferece recursos poderosos que podem agregar valor ao sistema no futuro, como autenticação através de redes sociais (*Facebook, Google+, Twitter* e contas da *Microsoft*). Todos aspectos referentes à gerencia de usuários foram implementados usando recursos providos pelo *ASP .NET Identity*, o que agilizou consideravelmente o desenvolvimento. O *framework* também trata diversas questões de segurança, como senhas criptografadas.

3.2.3 Modelo de Domínio

O modelo de domínio da aplicação tem por objetivo apresentar uma perspectiva conceitual e com alto nível de abstração sobre o domínio do problema. São identificadas as principais entidades do sistema e são definidos os relacionamentos entre elas. A ideia é levantar os dados que darão suporte à construção da aplicação.

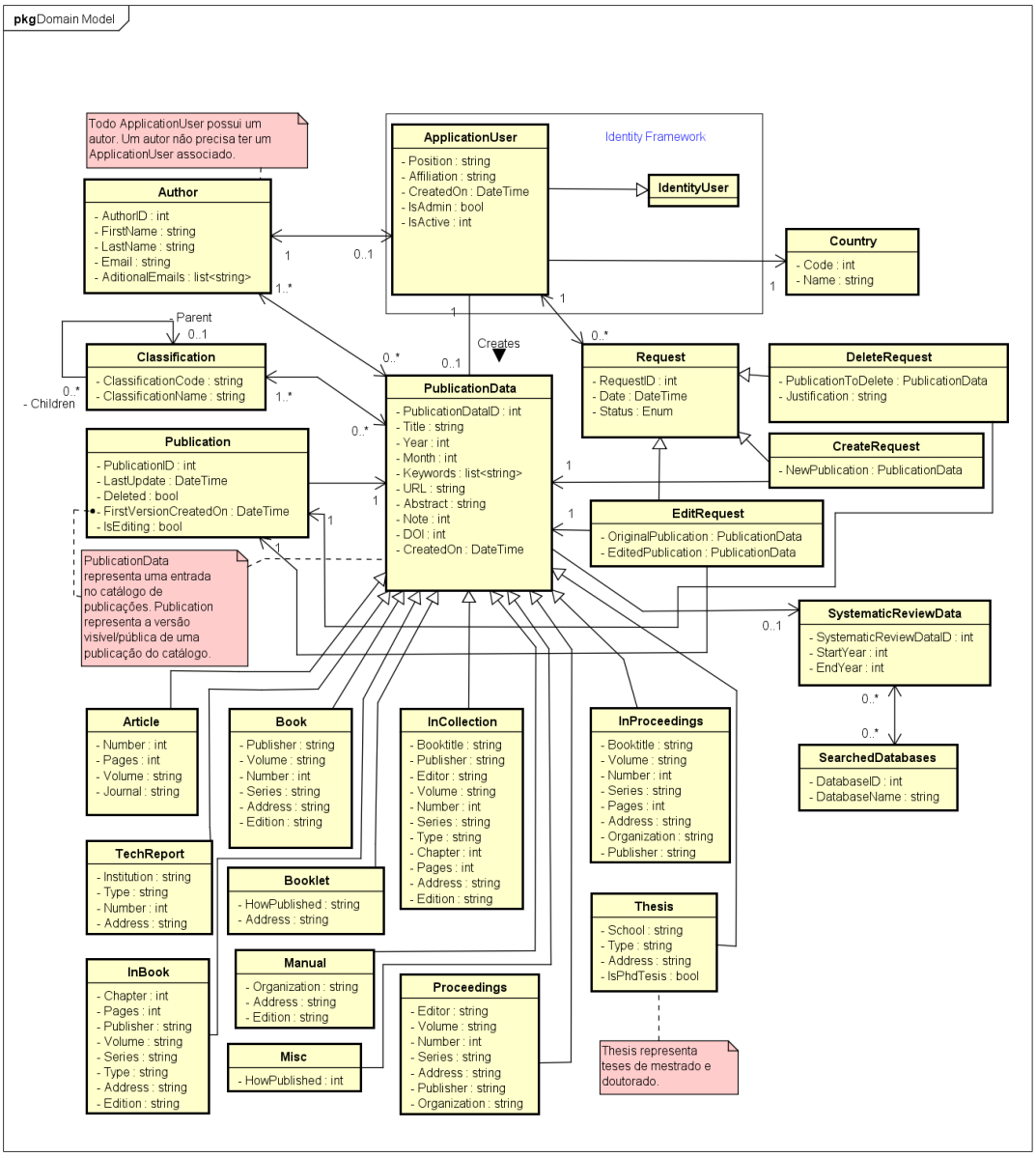
A Figura 3.2 ilustra o modelo de domínio da aplicação. Estão representadas as principais entidades do sistema, juntamente com os atributos mais relevantes. A construção do sistema foi completamente baseada em tal diagrama.

Observando a Figura 3.2, podemos inferir a maior parte do comportamento do sistema. A entidade `PublicationData` corresponde a uma entrada no catálogo de publicações. É essa entidade que contém os dados comuns a todos os tipos de trabalhos. Uma publicação deve ser de qualquer um dos onze tipos das entidades que herdam de `PublicationData`, ou seja, uma publicação pode ser uma entidade `Article`, `Book`, `TechReport`, entre outros.

A entidade `Publication` representa uma publicação visível a todos os usuários do sistema, isto é, ela aponta para uma entidade `PublicationData` aprovada pelos administradores, podendo ser retornada, portando, pelos mecanismos de busca. Quando um usuário cria uma nova publicação (ou seja, insere uma nova entidade que herda de `PublicationData` no catálogo de publicações), a mesma não encontra-se disponível aos demais usuários até que seja aprovada por um administrador. Somente quando um administrador aprova tal trabalho é que ele passa a ser apontado por uma entidade `Publication`, tornando-se validado e acessível aos demais usuários.

Essa separação conceitual entre `Publication` e `PublicationData` se justifica quando voltamos nossa atenção às solicitações dos usuários. Um solicitação pode ser

Figura 3.2: Modelo de domínio.



de qualquer entidade que herda de `Request`, isto é, pode ser uma solicitação de inserção de um novo estudo (`CreateRequest`), edição de uma publicação (`EditRequest`) ou exclusão de um determinado trabalho (`DeleteRequest`). Um pedido de inserção aponta somente para a entidade `PublicationData` associada, porque a nova entrada ainda precisa ser validada por um administrador. Já uma solicitação de edição contém tanto a publicação ativa que se está editando (`OriginalPublication`), quanto a nova versão da mesma com as edições (`EditedPublicationData`). Isso permite aos administradores compararem as duas versões. Ao aprovar a versão mais recente, ou seja, a versão com edições, basta que a entidade `Publication` associada passe a apontá-la. Por fim, um pedido de exclusão indica somente qual publicação ativa deve ser excluída, fornecendo uma justificativa.

No diagrama, é possível visualizar também a relação entre usuários, representados pela entidade `ApplicationUser` e autores retratados pela entidade `Author`. Conforme mencionado anteriormente, todo usuário tem um autor associado, entretanto, um autor não precisa estar ligado a um usuário. A relação ocorre desta forma porque um usuário cadastrado, ao adicionar uma nova publicação, informa os seus respectivos autores. Cada autor só será vinculado a um usuário no momento em que uma pessoa se cadastrar no sistema informando o mesmo endereço de *E-mail* atribuído ao autor. Enquanto isso não ocorre, o autor não está vinculado a nenhum usuário. Por outro lado, sempre que uma pessoa se inscreve no sistema, caso não exista um autor que possa ser vinculado a ela, cria-se então um novo autor associado. Neste ponto, nota-se também a presença do *framework ASP. NET Identiy*. Todos os dados relacionados à autenticação de um usuário são suportados pelo *framework* e estão presentes na entidade `IdentityUser`.

Observa-se também o fato de que uma publicação pode ou não ser uma revisão sistemática. Isso é percebido por meio da associação entre as entidades `PublicationData` e `SystematicReviewData`, a qual pode ter multiplicidade zero ou um. Os dados particulares de uma revisão sistemática ficam contidos na entidade `SystematicReviewData`, que atualmente engloba o período de abrangência da revisão sistemática. Além disso, é possível informar também as bases de busca nas quais foram pesquisados os trabalhos primários. A entidade `SearchedDatabases` contém essa informação. No futuro, provavelmente, serão adicionados novos atributos pertinentes a revisões sistemáticas, o que permitirá filtros mais complexos nos mecanismos de busca.

3.3 Considerações Finais

Observando o modelo de domínio da aplicação, pode-se notar a preocupação em suportar um grande número de formatos de publicações. A modelagem foi assim realizada porque as revisões de literatura podem ser disponibilizadas em diversas formas, tais como artigos, livros e relatórios técnicos. Isso torna o sistema flexível e adequado a qualquer publicação. Além disso, nota-se a preocupação com os atributos pertencentes às revisões sistemáticas, arranjados de uma maneira que seja fácil de estendê-los.

Outro ponto que merece destaque é a atenção à usabilidade e estética do software. A aplicação foi desenvolvida de modo a proporcionar uma utilização fácil e agradável, motivando o usuário através de interfaces simples e diretas.

4 LRDB: IMPLEMENTAÇÃO

O presente capítulo objetiva apresentar o sistema desenvolvido. Serão exibidas imagens que ilustram as principais funcionalidades do LRDB.

4.1 Apresentação do Sistema

A aplicação é composta, do ponto de vista do usuário, por diversas páginas *web*. A Figura 4.1 apresenta a página que contém o formulário de inscrição de um novo usuário.

Qualquer usuário pode visualizar as publicações mais recentes, ilustradas na Figura 4.2. Nessa imagem, podemos observar também o menu do sistema e barra de busca simples. Neste caso, o usuário ainda não realizou o *login*. A lista de publicações exibe alguns metadados de cada item. Se o usuário desejar visualizar a publicação com um detalhamento um pouco maior, pode clicar no botão *Quick View* em destaque na imagem.

Quando o usuário clica no botão *Quick View*, um pequeno resumo da publicação lhe é apresentado, conforme mostrado na Figura 4.3. Se o usuário desejar visualizar os demais dados da publicação, ele deve clicar em *Show Full Publication*, onde lhe será apresentada uma tabela com todos os metadados informados até o momento. Caso decida acessar diretamente o texto da publicação, ele pode clicar no *link Access Full Text*.

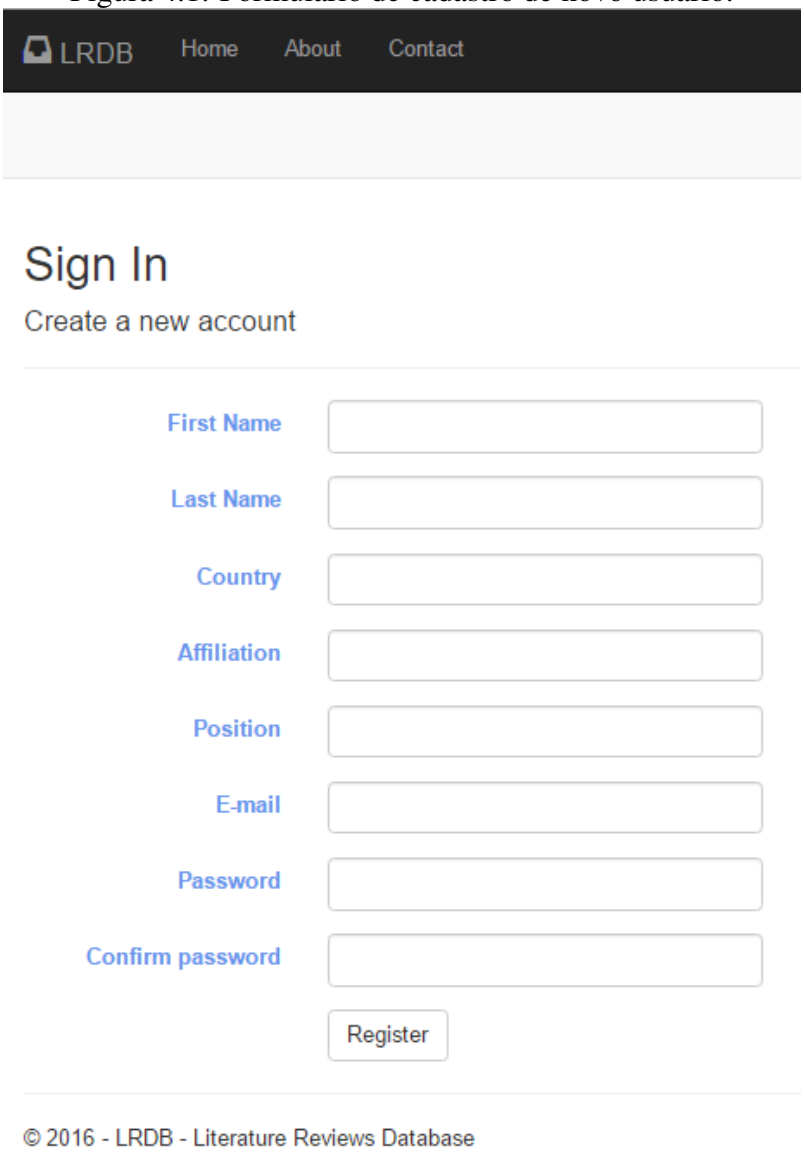
Ao adicionar uma publicação, um usuário deve selecionar qual seu tipo e informar se o trabalho que está sendo catalogado é uma revisão sistemática. A Figura 4.4 ilustra essa funcionalidade.

A Figura 4.5 mostra a página de edição de um artigo. Em geral, a tela de edição de um trabalho é praticamente a mesma usada para adição. Além disso, todos os tipos de publicação possuem um modelo similar, ou seja, o que difere são somente os metadados associados a cada tipo de publicação. Nessa imagem, é possível observar alguns pontos mencionados anteriormente, descritos a seguir.

- Adição ou remoção de classificação: um usuário pode adicionar ou remover classificações associadas a uma publicação. O sistema utiliza as classificações da ACM[1]. Para inserir uma nova classificação, ele deve digitar seu nome na caixa de texto indicada. Ao começar a digitar o nome de uma classificação, será apresentada uma lista de classificações que contém o termo digitado, permitindo ao usuário escolher

[1] http://www.acm.org/about/class/2012

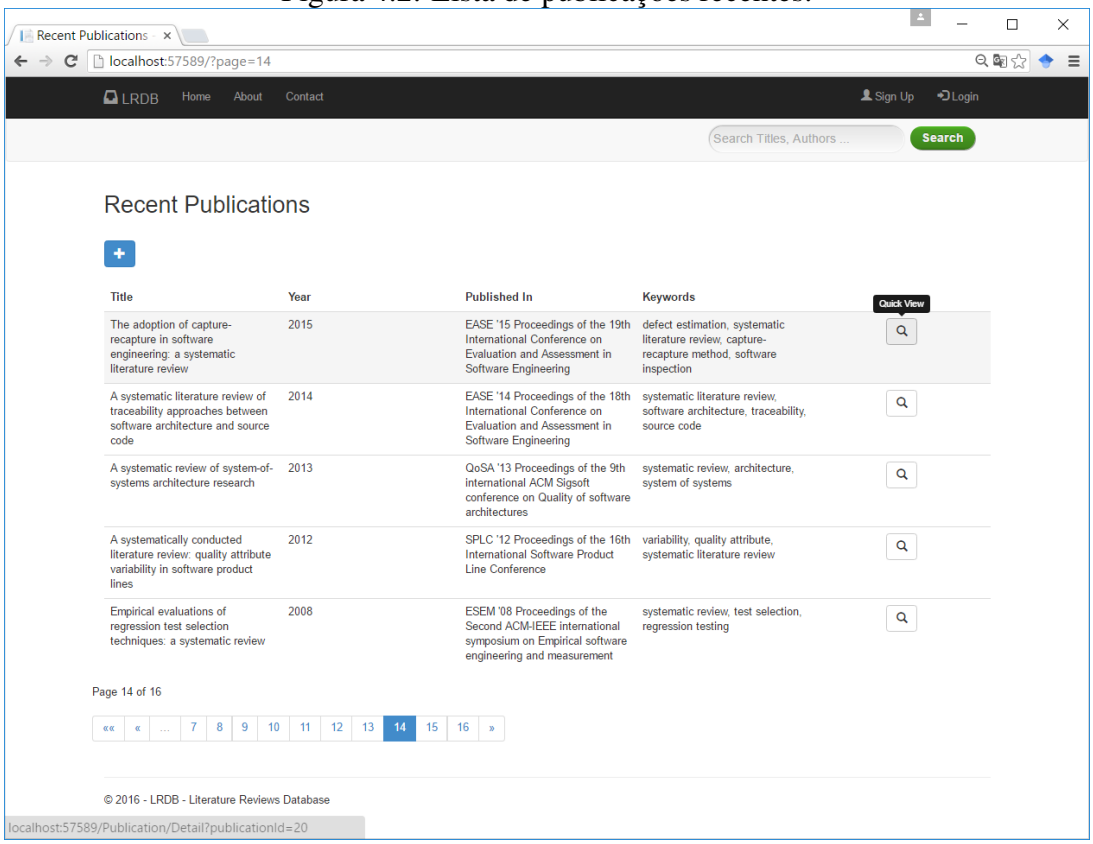
Figura 4.1: Formulário de cadastro de novo usuário.



uma opção. Isso garante que só serão adicionadas classificações válidas.

- Adição ou remoção de autores: é possível associar autores a publicações, conforme exibido na porção inferior da página. Para adicionar um novo autor, é necessário informar seu *E-mail*, nome e sobrenome.
- Itens pertinentes a revisões sistemáticas: na porção direita da página, temos os dados associados a revisões sistemáticas. É possível especificar o período de cobertura da revisão, isto é, o intervalo de datas que contém os trabalhos primários. E, analogamente às classificações, é possível adicionar as bases de buscas pesquisadas para a realização da revisão sistemática. Em breve, serão suportados mais metadados específicos de revisões sistemáticas.

Figura 4.2: Lista de publicações recentes.



Quando uma publicação é editada por um usuário, é gerada uma solicitação de edição, que será avaliada por um administrador. A Figura 4.6 apresenta a visualização de um pedido de edição. O administrador pode comparar as duas versões e aprovar ou não o pedido. Se for necessário, o administrador também pode fazer pequenos ajustes nos metadados e aprovar a publicação.

A funcionalidade de busca avançada é exibida na Figura 4.7. Acredita-se que as possibilidades de busca também serão aumentadas em breve.

A Figura 4.8 apresenta os recursos providos pelo *framework Bootstrap*. A página exibida na Figura 4.2 foi redimensionada automaticamente e pode-se observar que todos seus componentes adequaram-se ao novo tamanho. O mesmo ocorre em telas de *smartphones* e *tablets*, fornecendo ao usuário uma ótima usabilidade. É possível notar inclusive, no canto superior direito, que o formato de menu foi alterado.

Figura 4.3: *Quick view*.

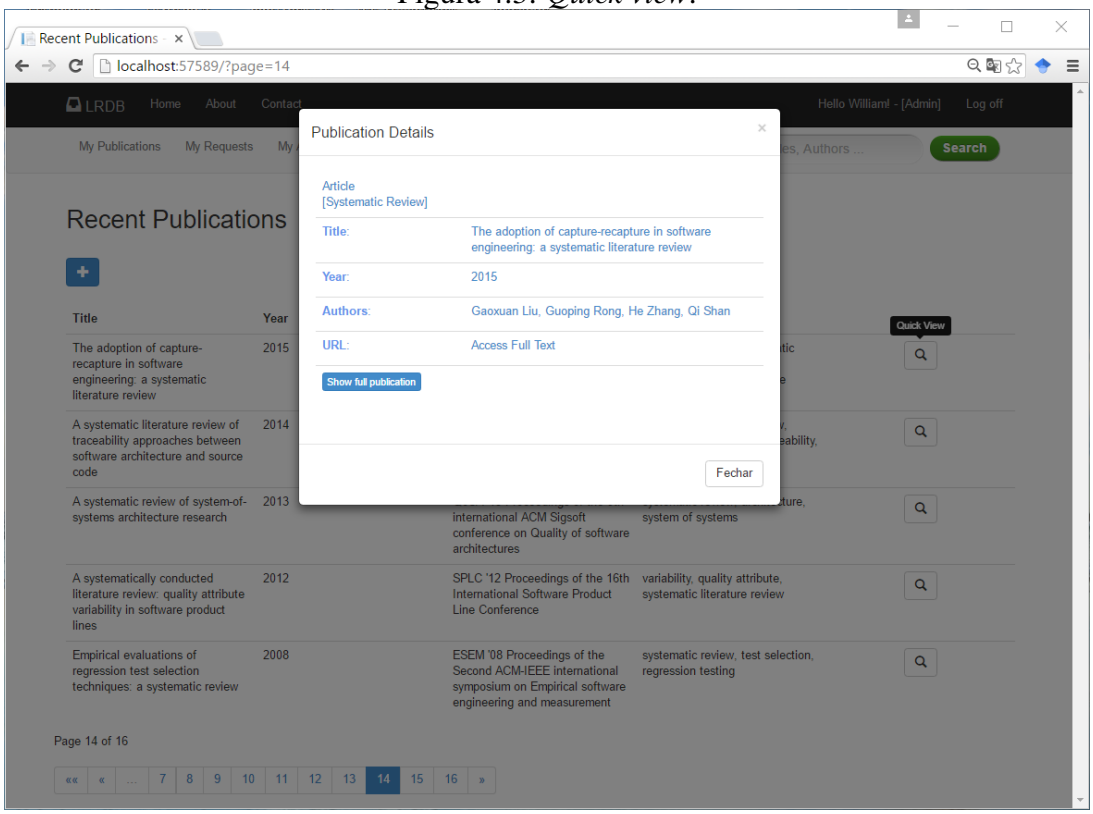


Figura 4.4: Seleção do tipo de publicação.

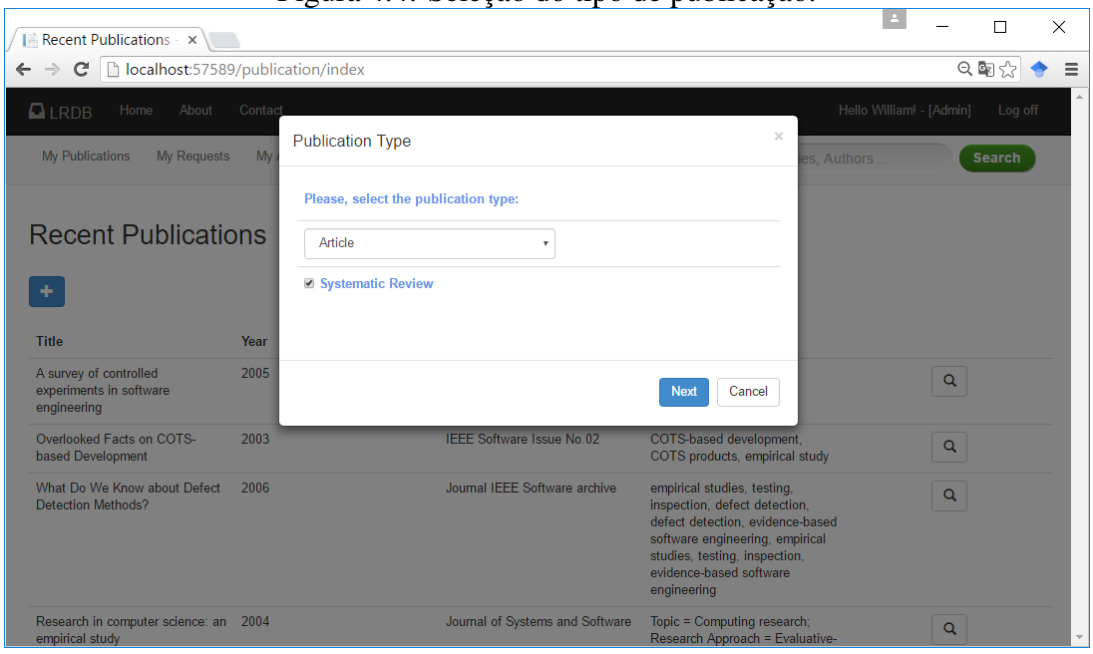


Figura 4.5: Edição de artigo.

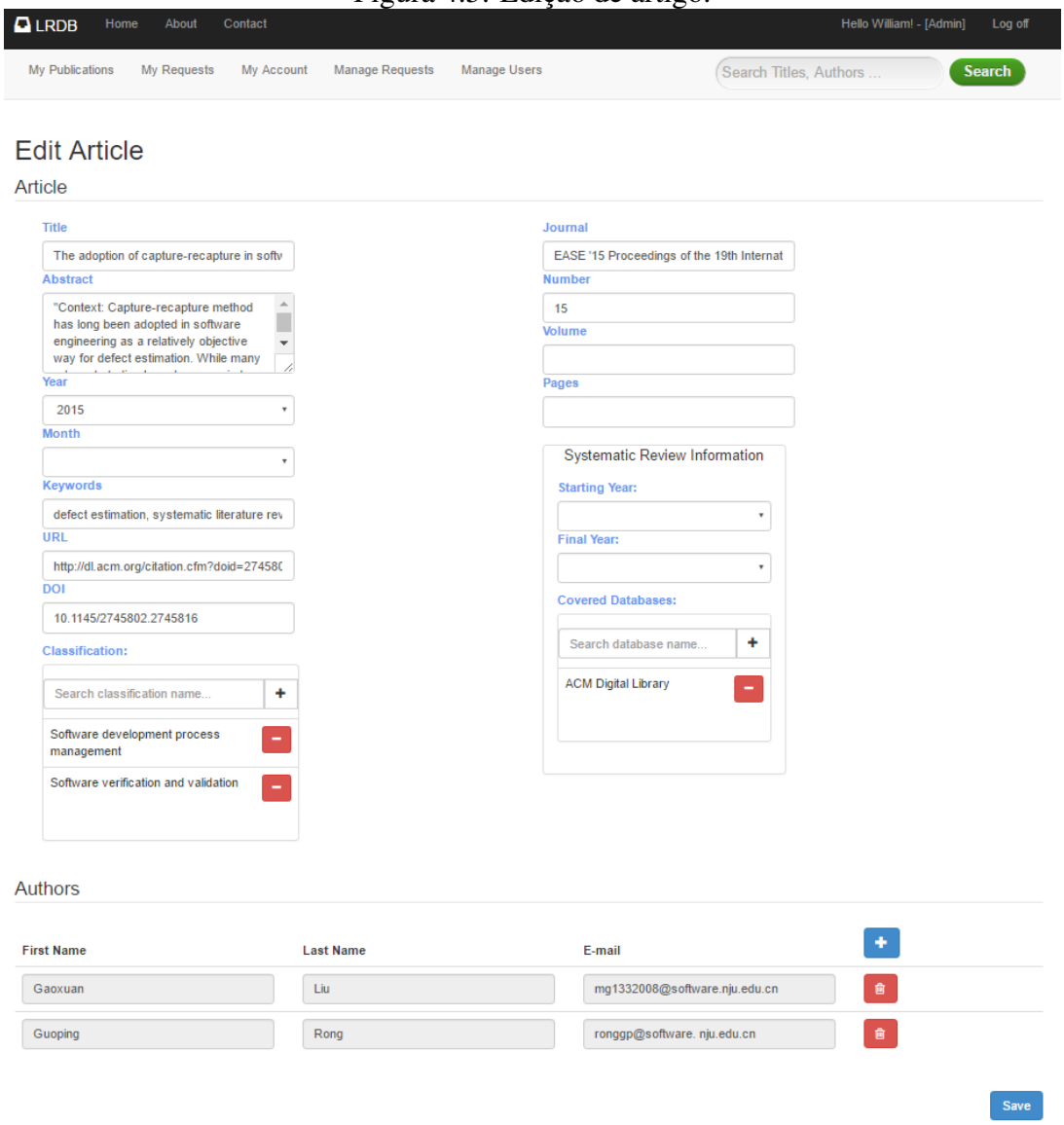


Figura 4.6: Solicitação de edição de publicação.

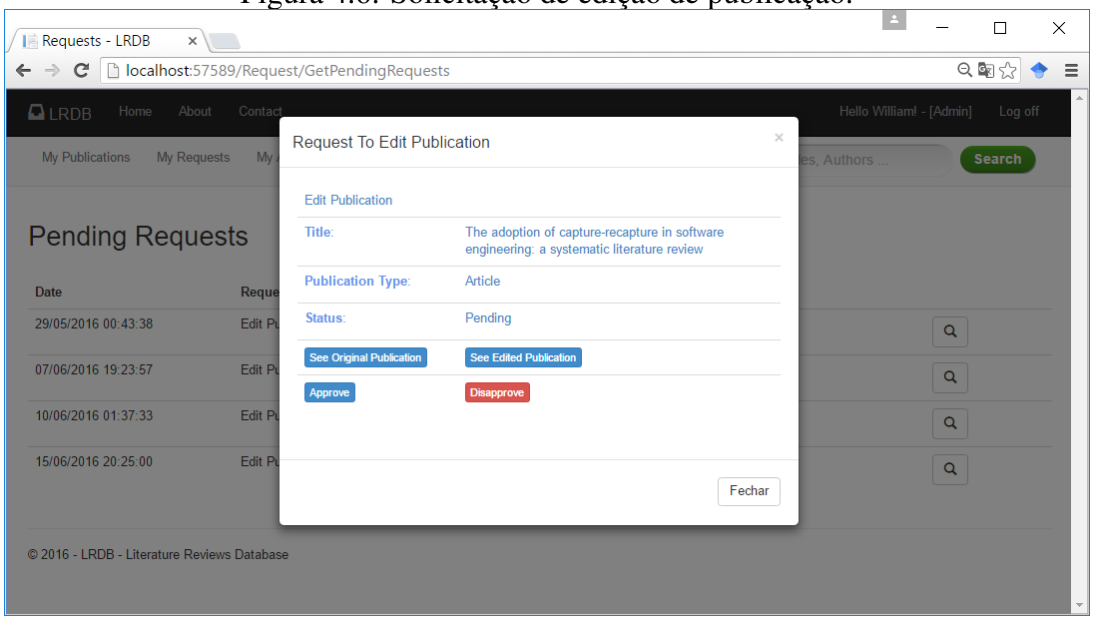


Figura 4.7: Busca avançada.

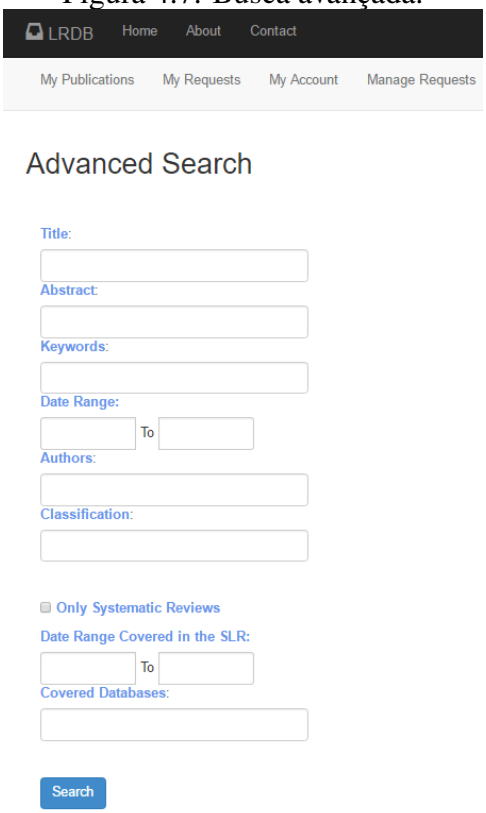
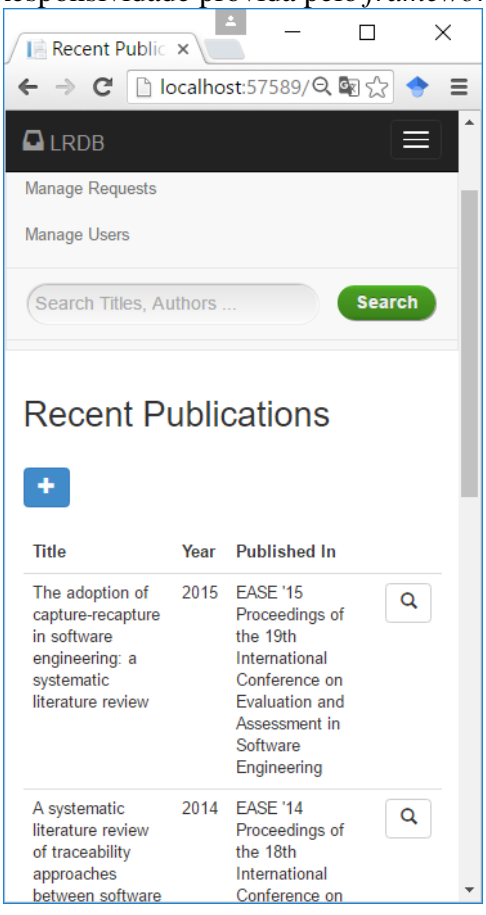


Figura 4.8: Responsividade provida pelo *framework Bootstrap*.



5 DADOS INICIAIS

Além da implementação do sistema, foi realizada uma população inicial do repositório apresentado. O objetivo é inserir dados iniciais, de modo a incentivar a adição de mais trabalhos e o uso imediato do repositório. O presente capítulo apresenta a maneira como tal população foi conduzida e fornece também alguns exemplos de consultas realizadas utilizando as ferramentas de busca da própria aplicação.

5.1 População Inicial do Repositório

Com o objetivo único de avaliar o comportamento do sistema e torná-lo útil logo após sua implementação, foram catalogadas algumas revisões sistemáticas de engenharia de software. Devido ao grande número de revisões publicadas e ao esforço necessário para encontrar e avaliar cada uma delas, foi criado um critério de busca restritivo o suficiente para permitir a catalogação de todos trabalhos selecionados em um curto prazo.

Basicamente, foram utilizados dois critérios de busca:

- Pesquisa em repositório existente: foi realizada uma busca avançada na plataforma *ACM Digital Library*[1], onde foi especificado que um ou mais termos da publicação deveriam conter a expressão *"Software AND Engineering AND Systematic AND Review"*. Tal pesquisa retornou 272 resultados. Após análise, verificamos que apenas 65 referiam-se, de fato, a revisões sistemáticas de engenharia de software. Destas publicações, 59 foram adicionadas ao sistema. Alguns trabalhos não foram inseridos porque não foram encontrados todos os metadados necessários.
- Adição dos itens cobertos em um estudo terciário: foi adicionada a grande maioria dos estudos selecionados na revisão sistemática de revisões sistemáticas de engenharia de software desenvolvida por Kitchenham et al. (2009). No total, 17 estudos foram adicionados. Não foi possível adicionar todos os trabalhos porque algumas informações necessárias para o cadastro não foram localizadas.

Para cada estudo catalogado, tentou-se obter o máximo de metadados associados possível. Essa tarefa é trabalhosa porque nem sempre os metadados estão disponíveis claramente. A Tabela 5.1 totaliza as revisões sistemáticas adicionadas ao sistema, agrupadas de acordo com a classificação atribuída.

[1] http://dl.acm.org/

Tabela 5.1: Revisões sistemáticas catalogadas.

Classificação	**Total de revisões sistemáticas**
Software and it's engineering	35
Software creation and management	21
Software organization and properties	10
Management of computing and information systems	9
Software engineering education	1

5.2 Exemplos de Consultas

Com o intuito de analisar o comportamento do sistema, foram realizados alguns exemplos de consultas. Certamente os mecanismos de pesquisa serão largamente utilizados e é preciso garantir que eles funcionam conforme o esperado. Os exemplos realizados são simples e envolvem uma população pequena de revisões sistemáticas, mas ainda assim é possível aferir o tempo de resposta da aplicação, o que é fundamental do ponto de vista do usuário.

A população considerada foi de 76 publicações e foram efetuadas consultas usando a busca simples e a busca avançada. Utilizou-se o navegador *Google Chrome*[2] para realização dos testes. Os tempos totais de repostas foram aferidos utilizando as ferramentas para desenvolvedor que o próprio navegador oferece[3]. O tempo total apresentado considera a média de 10 execuções. A seguir estão descritos as consultas realizadas e os resultados.

- Busca Simples
    - Pesquisa pelo sobrenome de um autor específico, associado a apenas uma publicação: foi retornada apenas 1 publicação, conforme esperado, e o tempo de resposta médio foi de 162 ms.
    - Pesquisa pelo termo "review": foram retornadas 66 publicações e o tempo de resposta médio foi de 198 ms.
    - Pesquisa por um termo que não possui nenhuma publicação associada: não foi retornada nenhuma publicação e o tempo de resposta médio foi de 167 ms.

[2]https://www.google.com.br/chrome/

[3]https://developers.google.com/web/tools/chrome-devtools/profile/network-performance/resource-loading#view-network-timing-details-for-a-specific-resource

- Busca Avançada
    - Pesquisa por trabalhos publicados entre os anos 2010 e 2015 que contém alguma palavra-chave igual a "*Web*": 2 publicações foram retornadas e o tempo de resposta total foi de 145 ms.
    - Pesquisa por um autor específico, associado a apenas uma publicação: apenas uma publicação foi retornada e o tempo de resposta médio foi de 136 ms.
    - Pesquisa por publicações que contém o termo "review" no título: foram retornadas 53 publicações e o tempo de resposta foi de 170 ms.
    - Pesquisa por uma palavra-chave não contida em nenhum trabalho: nenhuma publicação foi retornada e o tempo de resposta foi de 103 ms.

6 CONCLUSÃO

Ao longo deste trabalho, foram contextualizados os conceitos de revisão sistemática, sua importância na medicina e sua influência no âmbito de engenharia de software. Observou-se que a prática de revisões sistemáticas pode trazer diversos benefícios à área, como por exemplo, auxiliar na escolha das tecnologias adequadas para um novo projeto, reduzindo os riscos de problemas futuros. Além disso, constatou-se uma grande preocupação em aproximar os profissionais da indústria de software e os pesquisadores do meio acadêmico, promovendo uma cooperação baseada em evidências, isto é, permitindo a geração de resultados concretos e úteis para todos envolvidos. Contudo, percebe-se também que existem poucas ferramentas cujo objetivo principal é suportar as revisões sistemáticas em engenharia de software e, ainda mais crítico do que isso, as bases de dados atuais não diferenciam revisões sistemáticas de trabalhos primários, o que torna a pesquisa por revisões sistemáticas algo muito trabalhoso.

Foi apresentado o projeto e a implementação de uma solução colaborativa para centralizar as revisões sistemáticas já existentes em todas as áreas da computação, com enfoque principal em engenharia de software. A solução proposta oferece, também, suporte a revisões tradicionais como alternativa para agregar mais conhecimento.

A proposta apresentada serve como um primeiro passo para que se tenha, em um futuro próximo, um aparato tecnológico planejado de maneira a suportar revisões sistemáticas de computação, estimulando a prática desse tipo de estudo. Acredita-se que, com a colaboração e empenho de todos, é possível obter um sistema prático, confiável e disponível a qualquer pessoa que esteja em busca de conhecimento.

Foi realizada a população inicial do repositório, de forma a analisar o comportamento do sistema e gerar algumas métricas simples de desempenho, as quais nos permitem acreditar que a aplicação é viável e realmente pode ser utilizada.

Como trabalhos futuros, a curto prazo, é preciso realizar novos testes no sistema e melhorar sua estética e usabilidade em alguns pontos, além de produzir um material informativo para os visitantes, de modo a incentivá-los a utilizar o sistema. Além disso, deve ser realizada a implantação do sistema, permitindo que qualquer pessoa possa acessá-lo. A longo prazo, recursos existentes devem ser lapidados e novas funcionalidades devem ser adicionadas.

REFERÊNCIAS

BERWANGER, O. et al. Como avaliar criticamente revisões sistemáticas e metanálises. **Revista Brasileira de Terapia Intensiva**, SciELO Brasil, v. 19, n. 4, p. 475–480, 2007.

BIOLCHINI, J. et al. Systematic review in software engineering. **System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES**, v. 679, n. 05, p. 45, 2005.

BOWES, D.; HALL, T.; BEECHAM, S. Slurp: a tool to help large complex systematic literature reviews deliver valid and rigorous results. In: ACM. **Proceedings of the 2nd international workshop on Evidential assessment of software technologies**. [S.l.], 2012. p. 33–36.

CHADWICK, J.; SNYDER, T.; PANDA, H. **Programming ASP. NET MVC 4: Developing Real-World Web Applications with ASP. NET MVC**. [S.l.]: "O'Reilly Media, Inc.", 2012.

CIPRIANI, A.; GEDDES, J. Comparison of systematic and narrative reviews: the example of the atypical antipsychotics. **Epidemiologia e Psichiatria Sociale**, v. 12, p. 146–153, 9 2003. ISSN 2045-7979. Available from Internet: <http://journals.cambridge.org/article_S1121189X00002918>.

CMARIX. **5 Reasons to use ASP.Net MVC Framework**. 2016. Available from Internet: <http://www.cmarix.com/5-Reasons-to-use-ASP-Net-MVC-Framework>. Accessed in: 01 jun. 2016.

COCHRANE. **About us**. 2016. Available from Internet: <http://www.cochrane.org/about-us>. Accessed in: 03 jun. 2016.

CODEMENTOR. **Why Learn C?** 2016. Available from Internet: <http://www.bestprogramminglanguagefor.me/why-learn-c-sharp>. Accessed in: 02 jun. 2016.

COOK, D.; SACKETT, D. L.; SPITZER, W. O. Methodologic guidelines for systematic reviews of randomized control trials in health care from the potsdam consultation on meta-analysis. **Journal of clinical epidemiology**, Elsevier, v. 48, n. 1, p. 167–171, 1995.

COOK, D. J.; MULROW, C. D.; HAYNES, R. B. Systematic reviews: Synthesis of best evidence for clinical decisions. **Annals of Internal Medicine**, v. 126, n. 5, p. 376–380, 1997. Available from Internet: <+http://dx.doi.org/10.7326/0003-4819-126-5-199703010-00006>.

COOK, D. J.; MULROW, C. D.; HAYNES, R. B. Systematic reviews: synthesis of best evidence for clinical decisions. **Annals of internal medicine**, Am Coll Physicians, v. 126, n. 5, p. 376–380, 1997.

CRD. **Systematic reviews: CRD's guidance for undertaking reviews in health care**. [S.l.]: Centre for Reviews and Dissemination, 2009.

CROMBIE, I. K.; DAVIES, H. T. What is meta-analysis. **What is**, p. 1–8, 2009.

DATATELL. **Unit of Work, Repository, Entity Framework, and Persistence Ignorance**. 2016. Available from Internet: <https://datatellblog.wordpress.com/2015/02/11/unit-of-work-repository-entity-framework-and-persistence-ignorance/>. Accessed in: 01 jun. 2016.

DEVMEDIA. **ASP.NET MVC 3 com Razor – Revista .net Magazine 91**. 2016. Available from Internet: <http://www.devmedia.com.br/asp-net-mvc-3-com-razor-revista-net-magazine-91/22872>. Accessed in: 01 jun. 2016.

DYBÅ, T.; DINGSØYR, T. Strength of evidence in systematic reviews in software engineering. In: **Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. New York, NY, USA: ACM, 2008. (ESEM '08), p. 178–187. ISBN 978-1-59593-971-5. Available from Internet: <http://doi.acm.org/10.1145/1414004.1414034>.

DYBÅ, T.; DINGSØYR, T. Strength of evidence in systematic reviews in software engineering. In: ACM. **Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement**. [S.l.], 2008. p. 178–187.

DYBÄ, T.; KITCHENHAM, B. A.; JORGENSEN, M. Evidence-based software engineering for practitioners. **Software, IEEE**, IEEE, v. 22, n. 1, p. 58–65, 2005.

EPPI CENTRE. **History of systematic reviews**. 2016. Available from Internet: <http://eppi.ioe.ac.uk/cms/Default.aspx?tabid=68>. Accessed in: 03 jun. 2016.

FOWLER, M. **Patterns of enterprise application architecture**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2002.

GARG, A. X.; HACKAM, D.; TONELLI, M. Systematic review and meta-analysis: when one study is just not enough. **Clinical Journal of the American Society of Nephrology**, Am Soc Nephrol, v. 3, n. 1, p. 253–260, 2008.

GOUGH, D.; OLIVER, S.; THOMAS, J. **An introduction to systematic reviews**. [S.l.]: Sage, 2012.

GREENALGH, T. How to read a paper: Papers that summarise other papers: systematic reviews and meta-analysis. **British Medical Journal**, v. 315, p. 672–5, 1997.

HERNANDES, E. et al. Using GQM and TAM to evaluate StArt - a tool that supports Systematic Review. **CLEI Electronic Journal**, scielouy, v. 15, p. 3 – 3, 04 2012. ISSN 0717-5000. Available from Internet: <http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002012000100003&nrm=iso>.

JONATHAN DANYLKO. **10 Reasons To Start Using ASP.NET MVC**. 2016. Available from Internet: <http://www.danylkoweb.com/Blog/10-reasons-to-start-using-aspnet-mvc-E9>. Accessed in: 02 jun. 2016.

KAMALASANAN, A. **Different Approaches of Entity Framework**. 2016. Available from Internet: <https://www.simple-talk.com/dotnet/.net-framework/different-approaches-of-entity-framework/>. Accessed in: 03 jun. 2016.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering–a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009.

KITCHENHAM, B. A.; DYBA, T.; JORGENSEN, M. Evidence-based software engineering. In: IEEE COMPUTER SOCIETY. **Proceedings of the 26th international conference on software engineering**. [S.l.], 2004. p. 273–281.

LERMAN, J.; MILLER, R. **Programming Entity Framework: Code First**. [S.l.]: "O'Reilly Media, Inc.", 2011.

LIBERTY, J. **Programming C#: Building. NET Applications with C#**. [S.l.]: "O'Reilly Media, Inc.", 2005.

MARCORATTI, JOSE CARLOS. **.NET - Apresentando o padrão de projeto Repository**. 2016. Available from Internet: <http://www.macoratti.net/11/10/net_pr1.htm>. Accessed in: 03 jun. 2016.

MARCORATTI, JOSE CARLOS. **.NET - Padrão Repository e Unit of Work com EF 6 (revisitado)**. 2016. Available from Internet: <http://www.macoratti.net/16/01/net_uow1.htm>. Accessed in: 01 jun. 2016.

MARSHALL, C.; BRERETON, P.; KITCHENHAM, B. Tools to support systematic reviews in software engineering: A feature analysis. In: ACM. **Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering**. [S.l.], 2014. p. 13.

MICROSOFT. **Entity Framework**. 2016. Available from Internet: <http://www.asp.net/entity-framework>. Accessed in: 03 jun. 2016.

MULROW, C. D. The medical review article: state of the science. **Annals of Internal Medicine**, Am Coll Physicians, v. 106, n. 3, p. 485–488, 1987.

MULROW, C. D. Rationale for systematic reviews. **BMJ: British Medical Journal**, BMJ Group, v. 309, n. 6954, p. 597, 1994.

PIRES, E. **ASP.NET Identity – Tutorial Completo – Demos, Vídeo, Slides**. 2016. Available from Internet: <http://eduardopires.net.br/2014/08/asp-net-identity-tutorial-completo/>. Accessed in: 02 jun. 2016.

PIRES, EDUARDO. **ASP.Net MVC – View Model Pattern – Quando e como utilizar?** 2016. Available from Internet: <http://eduardopires.net.br/2013/08/asp-net-mvc-view-model-pattern-quando-e-como-utilizar/>. Accessed in: 02 jun. 2016.

RANIERI, BARBARA. **Veja o que é ORM e os frameworks disponíveis para .NET**. 2016. Available from Internet: <http://www.princiweb.com.br/blog/programacao/aspnet/veja-o-que-e-orm-e-os-frameworks-disponiveis-para-net.html>. Accessed in: 02 jun. 2016.

ROTHER, E. T. Revisão sistemática x revisão narrativa. **Acta Paulista de Enfermagem**, Escola Paulista de Enfermagem, v. 20, n. 2, p. v–vi, 2007.

SACKETT, D. et al. Evidence-based medicine: how to practice and teach ebm churchill livingston. **London, England**, 2000.

SCOTT EDWARDS. **Six Reasons Why You Should Learn C**. 2016. Available from Internet: <https://www.opensesame.com/blog/six-reasons-why-you-should-learn-c>. Accessed in: 02 jun. 2016.

SJØBERG, D. I. et al. A survey of controlled experiments in software engineering. **Software Engineering, IEEE Transactions on**, IEEE, v. 31, n. 9, p. 733–753, 2005.

SMITH, S. **Building Beautiful, Responsive Sites with Bootstrap**. 2016. Available from Internet: <https://docs.asp.net/en/latest/client-side/bootstrap.html>. Accessed in: 01 jun. 2016.

STAPLES, M.; NIAZI, M. Experiences using systematic review guidelines. **J. Syst. Softw.**, Elsevier Science Inc., New York, NY, USA, v. 80, n. 9, p. 1425–1437, sep. 2007. ISSN 0164-1212. Available from Internet: <http://dx.doi.org/10.1016/j.jss.2006.09.046>.

STAPLES, M.; NIAZI, M. Experiences using systematic review guidelines. **Journal of Systems and Software**, Elsevier, v. 80, n. 9, p. 1425–1437, 2007.

TECHBRIJ. **Generic Repository and Unit of Work Pattern, Entity Framework, Unit Testing, Autofac IoC Container and ASP.NET MVC [Part 1]**. 2016. Available from Internet: <http://techbrij.com/generic-repository-unit-of-work-entity-framework-unit-testing-asp-net-mvc>. Accessed in: 03 jun. 2016.

TICHY, W. F. et al. Experimental evaluation in computer science: A quantitative study. **Journal of Systems and Software**, Elsevier, v. 28, n. 1, p. 9–18, 1995.

TORRE-UGARTE, M. C. De-la; GUANILO, R. F. T.; BERTOLOZZI, M. R. Revisão sistemática: noções gerais. **Revista da Escola de Enfermagem, São Paulo**, SciELO Brasil, v. 45, n. 5, p. 1260–6, 2011.

TRENDOWICZ, A.; MÜNCH, J.; JEFFERY, R. State of the practice in software effort estimation: a survey and literature review. In: **Software Engineering Techniques**. [S.l.]: Springer, 2011. p. 232–245.

VISUAL STUDIO. **Controle de versão**. 2016. Available from Internet: <https://www.visualstudio.com/pt-br/products/tfs-overview-vs.aspx>. Accessed in: 02 jun. 2016.

VISUAL STUDIO. **Team Foundation Server**. 2016. Available from Internet: <https://www.visualstudio.com/pt-br/products/tfs-overview-vs.aspx>. Accessed in: 02 jun. 2016.

W3SCHOOLS. **ASP.NET Razor - Markup**. 2016. Available from Internet: <http://www.w3schools.com/aspnet/razor_intro.asp>. Accessed in: 01 jun. 2016.

YUSUF, S. Meta-analysis of randomized trials: looking back and looking ahead. **Controlled clinical trials**, Elsevier, v. 18, n. 6, p. 594–601, 1997.

ZELKOWITZ, M. V.; WALLACE, D. R.; BINKLEY, D. W. Experimental validation of new software technology. **Series on Software Engineering and Knowledge Engineering**, v. 12, p. 229–263, 2003.