Suporte à Execução de Técnicas de Priorização de Requisitos

Title: Support for the Execution of Requirements Prioritization Techniques

Diógenes Batista de Oliveira Diniz
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
dbod@ecomp.poli.br

Maria Lencastre Pinheiros de Menezes e Cruz
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
mlpm@ecomp.poli.br

Resumo *A priorização de requisitos é uma das principais etapas do processo de desenvolvimento de um produto de software, pois ela tem como objetivo garantir que as funcionalidades mais importantes e com o maior valor sejam entregues o mais rápido possível. Mesmo com tantos benefícios, observa-se que a indústria não aplica de forma representativa as técnicas de priorização de requisitos. Apesar de existirem várias técnicas consolidadas para a priorização de requisitos, existe uma dificuldade muito grande em encontrar essas técnicas reunidas em um único lugar na web. Assim, o objetivo deste trabalho é a construção de um sistema web que contenha a implementação de técnicas de priorização de requisitos já consolidadas, Hundred Dollar, MoSCoW, AHP e Matriz de Wiegers, facilitando a utilização das mesmas por todas as partes interessadas. Para o desenvolvimento da ferramenta foi utilizada a linguagem de programação PHP e o microframework Lumen. O sistema foi avaliado por um usuário em potencial e as opiniões sobre o sistema foram positivas. A partir dos testes e das análises realizadas percebeu-se a importância de uma ferramenta que dê suporte à priorização de requisitos reunindo diversas técnicas em um só lugar.*

Palavras-Chave: *técnicas de priorização de requisitos, desenvolvimento de software, php, lumen, sistema web, partes interessadas*

Abstract *The prioritization of requirements is one of the main steps in the development process of a software product, because it aims to ensure that the most important and most valuable functionalities are delivered as quickly as possible; even with so many benefits it's noted that the industry does not apply representative requirements prioritization techniques. Although there are several consolidated techniques for prioritizing requirements, there is a very great difficulty in finding them together on the web. Thus, the objective of this work is the construction of a web system that contains the implementation of prioritization requirements techniques that are already consolidated, Hundred Dollar, MoSCoW, AHP and Wiegers Matrix, facilitating their use by all stakeholders. For the development of the tool the PHP programming language and the Lumen microframework were used. The system was evaluated by a potential user and opinions about the system were positive. From the tests and analysis performed, it was realized the importance of a tool that supports the prioritization of requirements combining several techniques in one place.*

Keywords: *Prioritization requirements techniques, software development, PHP, Lumen, web system, stakeholders*

1 Introdução

A cada dia os *softwares* vêm se tornado a principal ferramenta para tomada de decisão em organizações. É notável a sua presença dentro de diferentes áreas como a educação, a medicina e a indústria. Mesmo sendo tão importantes, estudos apontam que 53% dos projetos finalizados não atendem as expectativas do cliente. De acordo com [3] esses números vem diminuindo a cada ano com o avanço da Engenharia de Requisitos.

Outra questão, bastante recorrente, é que os *softwares* precisam ser desenvolvidos de maneira rápida e eficiente. Para isso, precisam garantir que a etapa de priorização de requisitos seja feita de forma criteriosa e precisa, para que os principais requisitos sejam entregues de maneira mais ágil.

Porém, mesmo com toda a sua importância, muitas vezes a priorização é realizada de forma *ad-hoc*, ou seja, sem orientações claras [14], o que pode gerar uma priorização mal realizada, fazendo com que os respectivos *softwares* não atendam às expectativas dos seus usuários, ou seja, os sistemas não ficam alinhados com as reais necessidades da empresa.

O processo de priorização de requisitos é um grande desafio no processo de desenvolvimento de *software*, e caso não seja feito de maneira satisfatório pode ocasionar danos na reputação, perda de pedidos e redução dos lucros [15]. Já [9] afirma ser difícil para um cliente decidir quais dos seus requisitos são mais importantes, com diversos *stakeholders* essa decisão é ainda mais complicada, visto que cada um possui um ponto de vista diferente.

Este trabalho tem como objetivo criar e disponibilizar um conjunto de implementações, envolvendo diversas técnicas de priorização de requisitos, para que estas fiquem disponíveis para a uso por outros sistemas. O objetivo é facilitar a automatização do processo de priorização de requisitos, para que este seja cada vez mais utilizado, potencializando o desenvolvimento de *software* com melhor qualidade.

O restante deste artigo está organizado da seguinte forma: a Seção 2, apresenta a metodologia adotada. A seção 3, denominada Trabalhos Relacionados apresenta um breve histórico sobre trabalhos que estão relacionados a priorização de requisitos. A Seção 4, denominada Referencial Teórico, descreve a Engenharia de Requisitos e as técnicas de priorização de requisitos. A Seção 5 apresenta as técnicas e tecnologias utilizadas no desenvolvimento de aplicações *Web*, proposta neste artigo. A Seção 6 descreve a proposta do sistema, incluindo uma explicação das funcionalidades presentes, detalhando como e onde foram utilizadas as tecnologias no sistema. A Seção 7 mostra os resultados obtidos após a avaliação dos usuários. Por fim, na Seção 8 são feitas as considerações finais sobre todo o trabalho.

2 Metodologia

Para gerar o conjunto de técnicas propostas, este trabalho foi desenvolvimento em cinco etapas.

Na primeira etapa foi realizada uma revisão da literatura, que serviu como base para criar um entendimento para o refinamento da proposta. A revisão da literatura foi dividida em duas fases, na primeira fase o foco foi em trabalhos relacionados à priorização de requisitos, na segunda fase o foco foi na busca foi em trabalhos e ferramentas que já disponibilizassem técnicas de priorização implementadas.

Na segunda etapa foi montada a arquitetura do sistema, onde ficou definido que seria uma API REST, com o objetivo de ter várias técnicas de priorização de requisitos implementadas em um único lugar e tirar do usuário a complexidade dessas implementações.

Na terceira etapa foi constituída da implementação, propriamente dita, das diferentes técnicas. Após a codificação o sistema foi submetido a testes, e a partir dos resultados obtidos ele sofreu algumas alterações.

Na quarta etapa o sistema foi disponibilizado para um grupo de usuários utilizarem. Foi divulgado em um grupo de amigos. O sistema ficou disponível por cinco dias. Apenas duas pessoas usaram o sistema (uma para testes e outra para chamada em uma aplicação [11].

Na quinta etapa foi feita a análise dos dados, na qual foi observada se a utilização do sistema atendeu a expectativa dos usuários.

3 Trabalhos Relacionados

Para a elaboração do trabalho foi realizada uma etapa de pesquisa com o objetivo de identificar plataformas que estivessem diretamente ligadas às técnicas de priorização de requisitos e que, consequentemente, serviram como base para o desenvolvimento da ferramenta que será aqui proposta.

Em 2013, C. S. B. Letícia [1], desenvolveu uma ferramenta *web* para priorização de requisitos com foco nas necessidades do usuário, ferramenta esta que implementa as técnicas *Hundred Dollar* e Análise Importância-Desempenho (IPA).

O software reqT, cujo desenvolvimento iniciou em 2010 na *Lund University* pelo professor Bjorn Regnell com o objetivo inicial de prover uma ferramenta gratuita para ilustrar os conceitos da Engenharia de Requisitos

para os estudantes de Mestrado. Desde então , o reqT tornou-se uma laboratório de pesquisa de Engenharia de Requisitos para capturar ideias de modelagem de requisitos com base em experiências da prática e resultados de pesquisa.

A ferramenta Controla foi desenvolvida como trabalho final do Curso Bacharelado em Sistemas de Informação da Faculdade de Viçosa (FDV - MG) e tem como objetivo apoias as atividades inerentes a Engenharia de Software.

4 Referencial Teórico

De acordo com Thayer [2], a Engenharia de Requisitos é a primeira etapa dentro de todo o processo da engenharia de *software*, a qual estuda como coletar, entender, armazenar, verificar e gerenciar os requisitos. Como especificado em [2], a Engenharia de Requisitos é subdivida em cinco processos, sendo eles:

- Elicitação: fase onde ocorre a identificação das fontes dos requisitos e o levantamento dos requisitos;
- Análise: processo onde os requisitos levantados são analisados para se chegar na definição deles;
- Especificação: é criado um documento que contém a definição de todos requisitos analisados;
- Verificação: fase que busca assegurar que os requisitos especificados estão em concordância com o desejo dos usuários;
- Gerenciamento: é o planejamento e controle de todas as atividades citadas anteriormente.

Dentro do contexto da Engenharia de Requisitos existe a priorização de requisitos, que tem como objetivo identificar os requisitos mais importantes para o *software* em desenvolvimento. De acordo com [3], na priorização de requisitos faz-se necessário considerar questões como a importância dos requisitos, seus custos, riscos, tempo de desenvolvimento, entre outras. A priorização de requisitos deve estar presente em todos os processos da engenharia de requisitos.

A seguir, pode-se observar como a priorização de requisitos está presente em cada um dos processos [12]:

- Elicitação: determinar quais *stakeholders* e quais requisitos devem ser considerados primeiro;
- Análise: analisar primeiramente os requisitos que foram identificados com maior prioridade;
- Documentação: ao escrever o documento, os requisitos devem ser escritos de maneira ordenada, para satisfazer as regras de documentação;
- Verificação: determinar a ordem que os requisitos devem ser validados, para assim ter a ordem que os requisitos vão ser resolvidos;
- Gerenciamento: definir as solicitações de mudanças que devem ser realizadas em primeiro lugar.

De acordo com [4], as técnicas de priorização de requisitos podem ser classificadas em grupos de escalas: nominal, ordinal e racional, descritas a seguir:

- Nominal: os requisitos são separados em grupos de prioridades, onde dentro de cada grupo todos os requisitos tem a mesma prioridade; exemplo, técnica de MosCoW
- Ordinal: tem como objetivo ordenar a lista de requisitos; técnica de Ranking
- Racional: produz resultados que podem trazer diferença relevante entre os requisitos; *Hundred Dollar* ($100) e AHP;

Para o desenvolvimento do trabalho foram escolhidas quatro técnicas de priorização de requisitos que serão descritas nas subseções a seguir.

4.1 *Hundred Dollar* ($100)

Também conhecido como método dos 100 (cem) pontos. Como mencionado por [5], é uma técnica simples para priorização de requisitos.

A ideia é que cada *stakeholder* possui cem pontos para distribuir entre os requisitos. O resultado é apresentado em uma escala de proporção, onde os valores indicam o quanto um requisito é mais ou menos importante que o outro. Após a distribuição dos cem pontos entre todos os requisitos, os requisitos estão priorizados e não podem ser repriorizados, pelo menos não na versão para esse *stakeholder*. No quadro abaixo pode-se observar o ranking dos requisitos priorizados.

Figura 1: Disposição de pontos

Requisitos	**$100**
Requisito 1	20
Requisito 2	25
Requisito 3	50
Requisito 4	5

Fonte: O Autor

Figura 2: Requisitos Priorizados

Requisitos	**$100**	**Priorização Final**
Requisito 1	20	3

Requisito 2	25	2
Requisito 3	50	1
Requisito 4	5	4

Fonte: O autor

O *hundred dollar* não leva em consideração a quantidade relativa de esforço necessário para implementar um dos requisitos. A técnica se baseia apenas no valor recebido para um determinado conjunto de *stakeholders*.

4.2 MoSCoW

Esta técnica tem como objetivo chegar a um entendimento comum entre os *stakeholders* sobre a importância que cada um dá a um determinado requisito. A técnica divide os requisitos em quatro categorias:

- *Must*: são os requisitos que possuem a maior prioridade. Sem eles o projeto não será bem sucedido
- *Should*: também são requisitos de alta prioridade, tão importantes quanto o *must*, porém, podem existir soluções alternativas para eles, ou não são tão críticos
- *Could*: são requisitos desejaveis e de menor prioridade, eles não afetam a solução, apenas são incluídos se o tempo e os recursos assim permitirem.
- *Want*: são requisitos que os *stakeholders* gostariam de ter, não precisam ser implementados em uma determinada versão do *software*. Podem ser inseridos em uma versão futura ou nunca serem inseridos.

A técnica MoSCoW funciona melhor quando é executada em grupo. Porém, os *stakeholders* precisam compreender que nem todos os requisitos devem ser *must*.

4.3 Analytic Hierarchy Process (AHP)

A AHP foi desenvolvida por [6], e foi criada para tomadas de decisões complexas. Esta técnica compara todos os possíveis pares de requisitos no mesmo nível de hierarquia, com o objetivo de determinar a prioridade, utilizando valores em uma escala de 1 (um) a 9 (nove), onde um representa requisitos com a mesmo nível de importância e nove representa o que é mais importante.

De acordo com [7], pessoas que priorizam com AHP tendem a desconfiar dos resultados, já que se torna difícil ter em mente a prioridade individual de cada requisito o controle fica focado na comparação dos requisitos em pares. O passo-a-passo da técnica é descrito na Figura 1.

```
1. Separar os requisitos em níveis hierárquicos;
2. Para cada nível realizar os seguintes passos:
 A) Colocar os requisitos na matriz AHP n x n;
 B) Realizar a comparação em pares, utilizando a escala de pontuação
     para preencher as colunas;
 C) Somas as colunas;
 D) Normalizar a soma das colunas;
 E) Calcular a média das linhas.
```

Figura 1: Passos AHP Fonte: (Adaptado J. M. S. Júnior) [8]

Os quadros subsequentes exemplificam o processo de execução da técnica em questão.

Figura 31: Matriz Comparativa de Requisitos.

Requisitos	**Requisito 1**	**Requisito 2**	**Requisito 3**	**Requisito 4**
Requisito 1	1	1/5	1/9	1
Requisito 2	5	1	1	5
Requisito 3	9	1	1	5
Requisito 4	1	1/5	1/5	1

Fonte: (Adaptado Vargas, 2010)

Figura 4: Somatório das colunas.

Requisitos	**Requisito 1**	**Requisito 2**	**Requisito 3**	**Requisito 4**
Requisito 1	1	1/5	1/9	1
Requisito 2	5	1	1	5
Requisito 3	9	1	1	5
Requisito 4	1	1/5	1/5	1
Total	16,00	2,40	2,31	12,00

Fonte: (Adaptado Vargas, 2010)

Figura 5: Normalização das colunas.

Requisitos	Requisito 1	Requisito 2	Requisito 3	Requisito 4
Requisito 1	1/16 = 0,063	(1/5)/2,40 = 0,083	(1/9)/2,31 = 0,048	1/12 = 0,083
Requisito 2	5/16 = 0,313	1/2,40 = 0,417	1/2,31 = 0,433	5/12 = 0,417
Requisito 3	9/16 = 0,563	1/2,40 = 0,417	1/2,31 = 0,433	5/12 = 0,417
Requisito 4	1/16 = 0,063	(1/5)/2,40 = 0,083	(1/5)/2,31 = 0,087	1/12 = 0,083

Fonte: (Adaptado Vargas, 2010)

Figura 6: Requisitos priorizados AHP

Requisitos	Cálculo	Priorização Final
Requisito 1	[0,063+0,083+0,048+0,083]/4 =0,0693	6,93%
Requisito 2	[0,313+0,417+0,433+0,417]/4 =0,3946	39,46%
Requisito 3	[0,563+0,417+0,433+0,417]/4 =0,4571	45,71%
Requisito 4	[0,063+0,083+0,087+0,083]/4 =0,0789	7,89%

Fonte: (Adaptado Vargas, 2010)

4.4 Matriz de Wiegers

A matriz de priorização de Wiegers possui uma visão analítica de priorização de requisitos. A técnica se baseia em montar uma matriz, tomando como critérios para a formação dessa matriz: benefícios, riscos, prejuízos e custos. Por padrão todos os critérios possuem o mesmo peso, podendo ser ajustados, inclusive deixar todas com valor zero. A aplicação desta técnica é formada por nove passos descritos na Figura 2.

```
1. Determinar o peso ponderado do benefício, prejuízo, custo e risco;
2. Determinar os requisitos a serem priorizados;
3. Estimar o benefício relativo;
4. Estimar o prejuízo relativo;
5. Calcular os valores toais e percentuais de cada requisito:
   Valor%(Ri) = [Benefício(Ri) * PesoPonderadoBenefício] +
                [Prejuízo(Ri) * PesoPonderadoPrejuízo];
6. Estimar o custo relativo e calcular o percentual do custo de cada
   requisito;
7. Estimar o risco relativo e alcular o percentual do custo de cada
   requisito;
8. Calcular as prioridades dos requisitos individuais:
   Prioridade(Ri) = Valor%(Ri) / [Risco(Ri) * PesoPonderadoRisco ] +
   [Custo(Ri) * PesoPonderadoCusto];
9. Determinar a posição no ranking de prioridades de cada requisito.
```

Figura 2 : Passos Matriz Wiegers. Fonte: (Adaptado Wiegers e Beatty, 2013) [9]

O quadro abaixo exemplifica a execução da técnica em questão.

Figura 7: Priorização de Wiegers.

Pesos Relativos:	1	1			1		1		
Requisitos	Benefícios Relativos	Prejuízos Relativos	Valor Total	Valor %	Custo Relativo	Custo %	Risco Relativo	Risco %	Prioridade
Requisito 1	5	7	12	24%	4	25%	5	21,7%	51%
Requisito 2	7	6	13	26%	3	18,8%	7	30,4%	53%
Requisito 3	9	9	18	36%	4	25%	9	39,1%	56%
Requisito 4	3	4	7	14%	5	31,3%	2	8,7%	35%
Totais	24	26	50	100%	16	100%	23	100%	195%

Fonte: Passos Matriz de Wiegers (Adaptado J. M. S. Maurício)

A limitação desta técnica vem da obrigatoriedade de estimar o benefício, prejuízo, custo e risco para cada item, fazendo com que seja necessário revisitar a lista de requisitos após o *ranking* ser calculado para fazer ajustes necessários afim de calibrar os valores.

5 Sistemas Web

Sistemas *Web* são aqueles que são projetados para serem utilizados em um navegador, tipo *Google Chrome*, *Mozilla Firefox*, *Safari*, etc. através da *internet* ou aplicativos desenvolvidos utilizando tecnologias *Web*. Podem ser executados a partir de um servidor *Hypertext Transfer Protocol* (HTTP) ou localmente, no dispositivo do usuário. As aplicações *Web* costumam ser divididas em duas partes: *client-side* e *server-side*.

O *client-side* é a parte visual da aplicação, ou seja, a parte da aplicação que de fato é apresentada ao usuário. Costuma ser desenvolvida em tecnologias como *Hyper-*

text Markup Language (HTML), *Cascade Style Sheet* (CSS) e *JavaScript*.

No *server-side* é onde ocorre o processamento das informações geradas no *client-side*.Costuma ser desenvolvido em tecnologias como *Java*, *Python*, *PHP: Hyperterx Preprocessor* (PHP). No *server-side* é onde está localizado o banco de dados.

A comunicação entre *client-side* e *server-side* é realizada através do protocolo HTTP, que se baseia em requisições e respostas, que por sua vez são descritas em um conjunto de métodos que são: GET, POST, DELETE, etc.

Para a plataforma foram utilizadas duas soluções de integração de sistemas que são citadas nas subseções abaixo.

5.1 *Web services*

De acordo com [10], *web services* são aplicações modularizadas que podem ser descritas, publicadas e invocadas sobre uma rede, normalmente *WEB*, ou seja, *web services* são interfaces que descrevem uma coleção de operações que são acessíveis por uma rede através de mensagens no formato *eXtensible Markup Language* (XML) padronizadas, permitindo assim uma integração rápida e efetiva. Isso permite que diferentes sistemas se comuniquem utilizando uma linguagem padrão, como o *JavaScript Object Notation* (JSON) ou XML.

Segundo HANSEN et. al., o *web service* é um componente de software que não depende de implementação e plataforma. Pode ser descrito utilizando uma linguagem de descrição de serviço, publicado, registrado e descoberto por um mecanismo padrão. Portanto, ele pode ser invocado através de uma *Application Program Interface* (API) pela rede, construído em volta de outros serviços.

Web services costumam ser desenvolvidos baseados em dois padrões: *Simple Object Access Protocol* (SOAP) e *Representational State Transfer* (REST).

Aplicações com o padrão SOAP utilizam XML para trocas de mensagens e tem sem formato definido por um XML *schema*. O XML é constituído por três elementos:

- Envelope: define a estrutura da mensagem e como processá-la;
- *Header*: possui um conjunto de regras de codificação para expressar instâncias de tipos e dados definidos pelo aplicativo;
- *Body*: convenção para representação de requisições e respostas a procedimentos.

Diferentemente do padrão SOAP, os *web services* REST não têm como objetivo possuir um padrão, e sim utilizar de maneira correta o protocolo HTTP. Uma grande vantagem do REST é que o usuário não precisa de conhecimento de como a aplicação a ser consumida funciona, isso é graças ao *Hypermedia As The Engine of Application State* (HATEOAS), que define que todos os links de um *web service* devem informar qual ação será tomada a partir do ponto que o usuário está.

5.2 API

Application Program Interface (API) ou interface de programação de aplicativos é um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do *software*, mas apenas usar seus serviços.

O modo como uma API é construída tem bastante impacto na usabilidade, ela precisa ser construída de um modo que oculte a complexidade presente dentro dos módulos, assim o usuário não precisa entender a complexidade existente dentro dele. O *design* de interfaces de programação representa uma parte importante da arquitetura de *software* das API's.

6 Descrição do sistema

O sistema tem o objetivo de apresentar uma plataforma onde o usuário possa utilizar várias técnicas de priorização de requisitos em um único lugar. Para isso foi desenvolvida uma API que visa auxíliar na criação dessa plataforma, bem como remover a complexidade da implementação das técnicas de priorização de requisitos no desenvolvimento da plataforma.

Nas próximas subseções será apresentado o uso das técnicas implementadas. Em seguida, serão descritas as tecnologias usadas para o desenvolvimento da API.

6.1 Detalhamento do uso da API

Na API o usuário poderá acessar a técnica de priorização de requisitos desejada através de uma *Uniform Resource Identifier* (URI) que identificará a técnica escolhida, para cada técnica o usuário precisará fazer a entrada dos dados através de um JSON que possui um padrão para cada técnica e terá como resposta um JSON que possui os requisitos dispostos em ordem crescente tomando como base a prioridade.

Para utilizar a API o usuário vai acessar o endereço: https://priorizacao.herokuapp.com/public/ na sua aplicação através do método POST complementando o endereço com a URI desejada, conforme descrito a seguir.

URI	Hundred Dolar	*/dollar/ranking*
	MosCoW	*/moscow/ranking*

	AHP	*/ahp/ranking*
	Matriz de Wigers	*/wiegers/ranking*

6.1.1 *Hundred Dollar* ($100)

Ao acessar a URI */dollar/ranking* o usuário opta por utilizar a técnica *hundred dollar; assim*, precisa enviar os dados no formato JSON com o padrão descrito na Figura 4.

```
{
    "requisitos" : [{
        "idrequisito" : 1,
        "valor" : 20
    },{
        "idrequisito" : 2,
        "valor" : 25
    },{
        "idrequisito" : 3,
        "valor" : 50
    },{
        "idrequisito" : 4,
        "valor": 5
    }]
}
```

Figura 8: JSON de entrada da técnica *Hundred Dollar*.

O JSON é constituído de uma lista de requisitos nomeada. Cada requisito é identificado por um *id* (nomeado de *idrequisito*) e um campo (nomeado *valor*) que representa o quanto aquele requisito vale para o *stakeholder*. O valor varia entre zero e cem, e a soma de todos os valores deve ser um múltiplo de cem. O mesmo requisito pode ser enviado mais de uma vez, pois os dados de todos os *stakeholders* podem ser enviados de uma única vez.

Como resposta o usuário recebe um JSON com um padrão definido que é mostrado na Figura 5.

```
{
    "success": true,
    "data": [
        {
            "idrequisito": 3,
            "valor": 50,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 2,
            "valor": 25,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 1,
            "valor": 20,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 4,
            "valor": 5,
            "quantidade_entradas": 1
        }
    ]
}
```

Figura 9: JSON de saída da técnica *hundred dollar*.

O JSON de saída é composto por um campo *success*, que indica se a operação obteve sucesso, e um campo *data* que é uma lista ordenada dos requisitos que foram enviados como entrada. Cada requisito possui uma identificação que é representada pelo campo *idrequisito*; a soma dos valores enviados por cada *stakeholder* que é representada no campo *valor* e a quantidade de vezes que o requisito foi enviado que é representada pelo campo *quantidade_entradas*.

6.1.2 *MoSCoW*

Caso a URI acessada seja */moscow/ranking* a técnica a ser aplicada será a MoSCoW, onde o usuário terá que enviar os dados de entrada no formato JSON com o padrão descrito na Figura 6:

```
{
    "requisitos" : [{
        "idrequisito" : 1,
        "entrada" : "must"
    },{
        "idrequisito" : 2,
        "entrada" : "could"
    },{
        "idrequisito" : 3,
        "entrada" : "should"
    },{
        "idrequisito" : 4,
        "entrada": "want"
    }]
}
```

Figura 10: JSON de entrada da técnica MoSCoW

A entrada é formada por uma lista de requisitos cujo

identificador é *requisitos*, cada requisito é formado por um identificador nomeado *idrequisito* e uma entrada onde os valores são os descritos na técnica MoSCoW e estão no formato de *string*. O mesmo requisito pode ser enviado mais de uma vez, visto que o sistema deve ser utilizado por mais de um *stakeholder*.

A saída é enviada como um JSON com padrão definido na imagem abaixo (Figura. 7)


```
{
    "success": true,
    "data": [
        {
            "idrequisito": 1,
            "must": 1,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 3,
            "should": 1,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 2,
            "could": 1,
            "quantidade_entradas": 1
        },
        {
            "idrequisito": 4,
            "want": 1,
            "quantidade_entradas": 1
        }
    ]
}
```


Figura 11: JSON de saída da técnica MoSCoW.

A saída é constituída por um JSON composto por um campo *success* que determina o resultado da aplicação da técnica e um campo *data* que é uma lista formada por todos os requisitos que foram enviados na entrada. Os requisitos possuem um campo de identificação que é representada pelo campo *idrequisito*, um campo *quantidade_entradas* que representa a quantidade de entradas que o requisito teve. Para cada tipo de entrada da técnica MoSCoW vai ser gerada uma saída com o nome da entrada e a quantidade de vezes que ela foi utilizada, ou seja, se um requisito teve uma entrada *must* e duas *could*, a saída vai ter um campo *must* cujo valor é um, e um campo *could*, cujo valor é dois.

6.1.3 AHP

Ao acessar a URI */ahp/ranking* a técnica AHP será aplicada e a entrada será feita por um JSON com padrão estabelecido na imagem da Figura 8.


```
{
    "requisitos" : [{
        "idrequisito" : 1,
        "valores" : [1, 0.2, 0.111, 1]
    },{
        "idrequisito" : 2,
        "valores" : [5, 1, 1, 5]
    },{
        "idrequisito" : 3,
        "valores" : [9, 1, 1, 5]
    },{
        "idrequisito" : 4,
        "valores" : [1, 0.2, 0.2, 1]
    }]
}
```


Figura 12: JSON de entrada da técnica AHP.

O JSON de entrada é formado pelo campo *requisitos* que representa uma lista de requisitos, cara requisito possui um campo de identificação nomeado de *idrequisito* e um campo *valores* que é composto pelos valores que foram atribuídos na linha do requisito na matriz AHP. A operação é realizada por um único *stakeholder*, para múltiplos *stakeholders* deve ser realizada uma requisição para cada *stakeholder* e as respostas devem ser comparadas.

Como resposta o usuário recebe um JSON que possui padrão identificado na imagem a seguir (Figura9).


```
{
    "success": true,
    "data": [
        {
            "idrequisito": 1,
            "prioridade": 6.9299455502668392
        },
        {
            "idrequisito": 2,
            "prioridade": 39.463661113515073
        },
        {
            "idrequisito": 3,
            "prioridade": 45.713661113515073
        },
        {
            "idrequisito": 4,
            "prioridade": 7.8927322227030139
        }
    ]
}
```


Figura 13: JSON de saída da técnica AHP.

O JSON de saída é formado por um campo *success*, que indica o sucesso da aplicação da técnica, e um campo *data* que é a lista dos requisitos enviados na entrada. Por sua vez. Cada requisito é formado por um campo *idrequisito* que identifica o requisito e um campo *prioridade* que é a porcentagem do quanto ele é prioritário dentro da lista de requisitos.

6.1.4 Matriz de Wiegers

No caso da URI */wiegers/ranking* ser acessada a técnica a ser aplicada será a matriz de Wiegers, onde os dados de entrada são enviados no formato JSON cujo padrão é descrito na imagem a seguir (Figura 10)

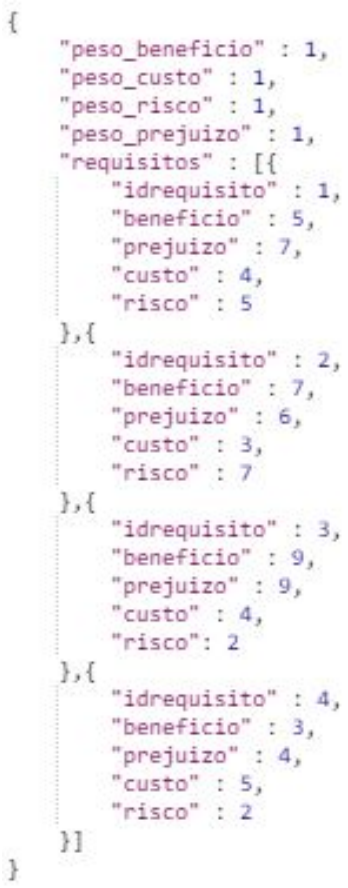

Figura 14: JSON de entrada da técnica matriz de Wiegers.

Para a matriz de Wiegers o JSON de entrada é constituído de quatro valores que identificam as variáveis de configuração da técnica, que são: *peso_benefício*, *peso_custo*, *peso_risco* e *peso_prejuizo*. Os valores são utilizados para definir os pesos que cada tipo de entrada terá nos cálculos. Dentro do JSON também consta uma lista de requisitos identificada por *requisitos*, onde cada requisito presente na lista possui um identificador cujo nome é *idrequisito* e para cada operador da matriz de Wiegers existe um campo que tem como identificador o nome do operador e um valor que será utilizado nos cálculos da priorização, ou seja, para o operador risco, existe um campo identificado por *risco* e um valor atribuído a ele. A operação é realizada por um único *stakeholder*, para múltiplos *stakeholders* deve ser realizada uma requisição para cada *stakeholder* e as respostas devem ser comparadas.

Como saída a operação retorna um JSON cujo padrão é definido a seguir (ver Figura 11).

O JSON de saída é formado por um campo *success* que identifica o sucesso da operação e um campo *data* que é uma lista dos requisitos que foram utilizados na entrada. Cada requisito tem como identificador o campo *idrequisito* e um atributo *prioridade* que representa a porcentagem do quanto ele é prioritário dentro da lista de requisitos.

Figura 15: JSON de saída da técnica matriz de Wiegers.

6.1.5 Mensagem de erro

Caso qualquer uma das operações descritas acima falhe, a API retornará um JSON que tem o padrão descrito abaixo:

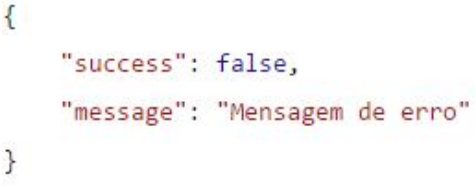

Figura 16: JSON de erro.

O JSON de erro possui um campo *success* que terá *false* como valor, visto que ocorreu um erro na operação e um campo *message* que trará uma mensagem indicando qual erro ocorreu.

6.2 Tecnologias utilizadas

Para o desenvolvimento da API foi considerado a utilização de tecnologias atuais, de fácil manutenção, baixo

custo e alta produtividade.

Na construção da API foi escolhido a montagem de um *Web service* seguindo o padrão REST, com o objetivo de não ter a complexidade de se montar uma interface visual para o usuário final. No *server-side* foi utilizado o *micro-framework* Lumen, que é baseado em PHP.

Lumen é um *micro-framework* que tem como objetivo a construção de *web services*. O que torna ele um *micro-framework* é o fato de não existir *views*, ou seja, o Lumen não apresenta interface com o usuário, deixando a cargo do desenvolvedor utilizar a tecnologia que mais lhe agradar.

O código fonte do sistema é aberto e está armazenado em um repositório do tipo *Git*, cujo endereço é https://github.com/diogenesdiniz/priorizacao.

7 Resultados obtidos

Desde o início da ferramenta, a mesma foi utilizada para o auxílio na construção de um sistema por S. V. Danilo [11], e após a sua finalização foi aplicado um questionário com o objetivo de reunir informações sobre a utilização e comportamento da ferramenta.

De modo geral o usuário reportou que achou a ideia pertinente, de fácil manuseio e útil. Foi observado que o acesso aos serviços foram feitos de maneira satisfatória, visto que as documentações para entrada e saída dos requisitos estavam bem descritas.

O ponto negativo ficou na ausência de uma página inicial que contenha a documentação da ferramenta, além da descrição das técnicas implementadas nela. Essa melhoria permitiria que usuários com conhecimento de programação, mas com pouco conhecimento das técnicas de priorização de requisitos, conseguiriam utilizar a ferramenta.

O trabalho tenta deixar claro a importância de uma ferramenta para priorização de requisitos que contenha a implementação de diversas técnicas em um lugar, melhorando assim a atividade de priorização e tornando-a cada vez mais presente na indústria.

8 Conclusões e trabalhos futuros

Este trabalho apresenta uma API construída para a implementação de várias técnicas de priorização de requisitos em um único lugar, auxiliando assim a construção de ferramentas que se utilizem dessas implementações.

A partir dos testes realizados foi observado o quão útil é uma API que possua as técnicas de priorização implementadas.

Também durante os testes foi analisada a eficiência da API, mostrando que ela é bastante rápida, realizando cálculos complexos em menos de um segundo, o que é considerado extremamente rápido para requisições em uma API.

Como trabalhos futuros propõem-se a implementação de outras técnicas de priorização de requisitos e de uma ferramenta que consuma a API implementada neste trabalho, com o objetivo de ser uma ferramenta que facilite a priorização de requisitos e torne a atividade um hábito dentro das mais diversas equipes de desenvolvimento de software.

Referências

[1] C. S. B. Letícia, Ferramenta Web para priorização de requisitos funcionais com foco nas necessidades dos usuários, 2013.

[2] Thayer, R. H. e Dorfman, M. Introduction to Tutorial Software Requirements Engineering in Software Requirements Engineering. IEEE-CS Press, Second Edition. Pag 1-2, 1997

[3] K.Kohl, C. Rupp. Fundamentos da Engenharia de Requisitos, 1 ed, vol 3, Santa Barbara, CA, 2011. Página 27.

[4] Qiao, M. The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review. Dissertação de Mestrado, Auckland University of Technology, 2009.

[5] Berander, P. e Andrews, A. Requirements Prioritization in Engineering and managing software requirements. Página 69-94.

[6] Saaty, T. L The analytic hierarchy process. McGraw-Hill, New York, 1980.

[7] Aurum, A. Wohlin, C. Prioritizing Legal Requirements, RELAW, Atlanta, USA, 2009.

[8] J. M. S. Júnior, G-4REPrioritization: Um Guia para Apoio à Escolha de Técnicas de Priorização de Requisitos, Dissertação de Mestrado, Universidade de Pernambuco, Outubro, 2015.

[9] Wiegers, K.; Beatty, J. Software Requirements, 3 ed., Redmond, Washington, 2013.

[10] HANSEN, Roseli Person et al. Web Services: na architectural overview. In: First International Seminar on Advanced Research in E-business-EBR, 2002.

[11] S. V. Danilo, Ferramenta de auxílio ao guia de apoio à escolha de técnicas de priorização de requisitos (G4REPrioritization), 2017.

[12] Klaus Pohl. Requirements Engineering: Fundamentals, Principles, and Techniques, 1st ed Springer Publishing Company, Incorporated, 2010.

[13] Ricardo Vargas. Utilizando a Programação Multicritério (Analytic Hierarchy Process -AHP) para Selecionar e Priorizar Projetos na Gestão de Portfólio, Disponível em: http://www.ricardo-vargas.com/pt/articles/analytic-hierarchy-process/, 10/10/2010, Acessado em 20/06/2017

[14] Balushi, T. H. A.; Sampaio, P. R. F.; Loucopoulos, P. Eliciting and prioritizin quality requirements supported by ontologies: a case study using the ElicitO framework and tool, Expert Systems, Vol 30(2). Mai. 2013.

[15] Karlson, J.; Ryan, K. Supporting the selection of Software Requirements. In: INTERNATIONAL WORKSHOP ON SOFTWARE SPECIFICATION AND DESIGN (IWSSD '96), 8th Proceedings , 1996, p. 146-149.

Apêndice A

Questionário de Avaliação da Usabilidade da Ferramenta de Suporte à Técnicas de Priorização de Requisitos

- Perfil do Usuário
  - (1) Nome
  - (2) E-mail
  - (3) Instituição/Empresa
  - (4) Qual é a sua experiência em desenvolvimento de software?
    - (a) 1 ano ou menos
    - (b) 2 a 4 anos
    - (c) 5 a 10 anos
    - (d) Mais de 10 anos
  - (5) Qual é atualmente a sua atividade profissional?
    - (a) Business / Requirements Analyst
    - (b) Project Management
    - (c) Product Development / Engineering
    - (d) QA Testing
    - (e) Product Management
    - (f) Executive Management
    - (g) Outside Consultant
    - (h) Usability Design
    - (i) Student

- Hundred Dollar
  - (1) Qual foi a sua experiência em acessar o webservice da técnica em questão?
    - (a) Péssimo
    - (b) Ruim
    - (c) Regular
    - (d) Bom
    - (e) Ótimo
  - (2) A descrição dos dados de entrada estavam claros?
    - (a) Péssimo
    - (b) Ruim
    - (c) Regular
    - (d) Bom
    - (e) Ótimo
  - (3) A descrição dos dados de saída estavam claros?
    - (a) Péssimo
    - (b) Ruim
    - (c) Regular
    - (d) Bom
    - (e) Ótimo
  - (4) A aplicação das técnicas atendeu as suas expectativas?
    - (a) Péssimo
    - (b) Ruim
    - (c) Regular
    - (d) Bom
    - (e) Ótimo

- MoSCoW
  - (1) Qual foi a sua experiência em acessar o webservice da técnica em questão?
    - (a) Péssimo
    - (b) Ruim
    - (c) Regular
    - (d) Bom
    - (e) Ótimo
  - (2) A descrição dos dados de entrada estavam claros?
    - (a) Péssimo

(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(3) A descrição dos dados de saída estavam claros?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(4) A aplicação das técnicas atendeu as suas expectativas?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

- AHP

(1) Qual foi a sua experiência em acessar o webservice da técnica em questão?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(2) A descrição dos dados de entrada estavam claros?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(3) A descrição dos dados de saída estavam claros?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(4) A aplicação das técnicas atendeu as suas expectativas?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

- Matriz de Wiegers

(1) Qual foi a sua experiência em acessar o webservice da técnica em questão?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(2) A descrição dos dados de entrada estavam claros?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(3) A descrição dos dados de saída estavam claros?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(4) A aplicação das técnicas atendeu as suas expectativas?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

- Ferramenta de suporte à técnicas de priorização de requisitos

(1) Você conseguiu utilizar as técnicas de priorização de requisitos?
(a) Sim
(b) Não

(2) Você considera útil esta ferramenta para seu trabalho?
(a) Sim
(b) Não

(3) Qual sua avaliação para a ferramenta?
(a) Péssimo
(b) Ruim
(c) Regular
(d) Bom
(e) Ótimo

(4) Qual sua sugestão para melhorar esta ferramenta?

Autorização de publicação de PFC

Eu, **DIOGENES BATISTA DE OLIVEIRA DINIZ** autor do projeto de final de curso intitulado: **SUPORTE À EXECUÇÃO DE TÉCNICAS DE PRIORIZAÇÃO DE REQUISITOS;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Diógenes Batista de Oliveira Diniz
DIOGENES BATISTA DE OLIVEIRA DINIZ

Orientador(a) Maria Lencastre Pinheiro de Menezes Cruz

coorientador(a)

Professor de TCC Sérgio Campello Oliveira

Data

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 7 de Julho de 2017, às 10:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente **DIOGENES BATISTA DE OLIVEIRA DINIZ**, orientado pelo professor **Maria Lencastre Pinheiro de Menezes Cruz**, sob título **SUPORTE À EXECUÇÃO DE TÉCNICAS DE PRIORIZAÇÃO DE REQUISITOS**, a banca composta pelos professores:

Maria Lencastre Pinheiro de Menezes Cruz
João Henrique Correia Pimentel

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 8,5 (oito e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 07 dias para entrega da versão final da monografia a contar da data deste documento.

MARIA LENCASTRE PINHEIRO DE MENEZES CRUZ

JOÃO HENRIQUE CORREIA PIMENTEL

* Este documento deverá ser encadernado juntamente com a monografia em versão final.