UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN

FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT

DEPARTAMENTO DE INFORMÁTICA – DI

Renato Monteiro Nicacio

WEB-SAM - Desenvolvimento de um sistema web para classificação do nível cognitivo do discente com base em Mapas Conceituais.

MOSSORÓ - RN

2021

Renato Monteiro Nicacio

WEB-SAM - Desenvolvimento de um sistema web para classificação do nível cognitivo do discente com base em Mapas Conceituais.

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do Prof. Dr. Rommel Wladimir de Lima.

MOSSORÓ - RN

2021

Renato Monteiro Nicacio

WEB-SAM - Desenvolvimento de um sistema web para classificação do nível cognitivo do discente com base em Mapas Conceituais.

Relatório apresentado como pré-requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovado em: 13 / 05 / 2021

Banca Examinadora

Prof. Dr. Rommel Wladimir de Lima (Orientador)
Universidade do Estado do Rio Grande do Norte – UERN

Prof. Ms. Raimundo Nonato Bezerra Neto (Coorientador)
Universidade do Estado do Rio Grande do Norte – UERN

Prof. Dr. Francisco Chagas de Lima Júnior
Universidade do Estado do Rio Grande do Norte – UERN

Prof. Dr. Isaac de Lima Oliveira Filho
Universidade do Estado do Rio Grande do Norte – UERN

SUMÁRIO

1 INTRODUÇÃO 5

2 OBJETIVOS 6

3 METODOLOGIA 7

4 DESCRIÇÃO DO SISTEMA 8

5 RESULTADOS 21

REFERÊNCIAS 22

1 INTRODUÇÃO

Com o passar dos anos a informação torna-se algo cada vez mais acessível, trazendo para o ensino tradicional o desafio constante de manter os alunos motivados em sala de aula bem como otimizar o processo de ensino para os professores. Neste sentido é possível fazer uso de ferramentas educacionais e tecnológicas a fim de simplificar atividades do cotidiano.

Os mapas conceituais são estratégias usadas para medir o nível de conhecimento do aluno, podendo agilizar a realização de tarefas e mostrar novos meios de ensino para o docente. Segundo (NETO, 2019, p.19) “[...] mapas conceituais caracterizam-se como artifício ativo que faz com que os alunos relacionem o conteúdo e assimilem o conhecimento, conectando novos saberes com os que já possuem, o que possibilita uma aprendizagem mais significante.”

2 OBJETIVOS

OBJETIVO GERAL: Desenvolvimento de um sistema web para a identificação do nível de cognição do aluno.

OBJETIVOS ESPECÍFICOS:

- Estudar técnicas de desenvolvimento web integradas com a de programação linguagem python;
- Estudar e implementar técnicas de processamento e segmentação de texto e análise lexical;
- Realizar testes para legitimar o sistema;
- Avaliar e efetivar os resultados obtidos.

3 METODOLOGIA

Para o desenvolvimento da proposta, a metodologia foi dividida em quatro etapas:

i) revisão bibliográfica sobre a arquitetura de (NETO, 2019), para determinar as abordagens a serem utilizadas na proposta;

ii) realizar levantamento de requisitos para o desenvolvimento de uma ferramenta web server-side;

iii) desenvolvimento do protótipo;

iv) Validação do sistema e análise dos resultados.

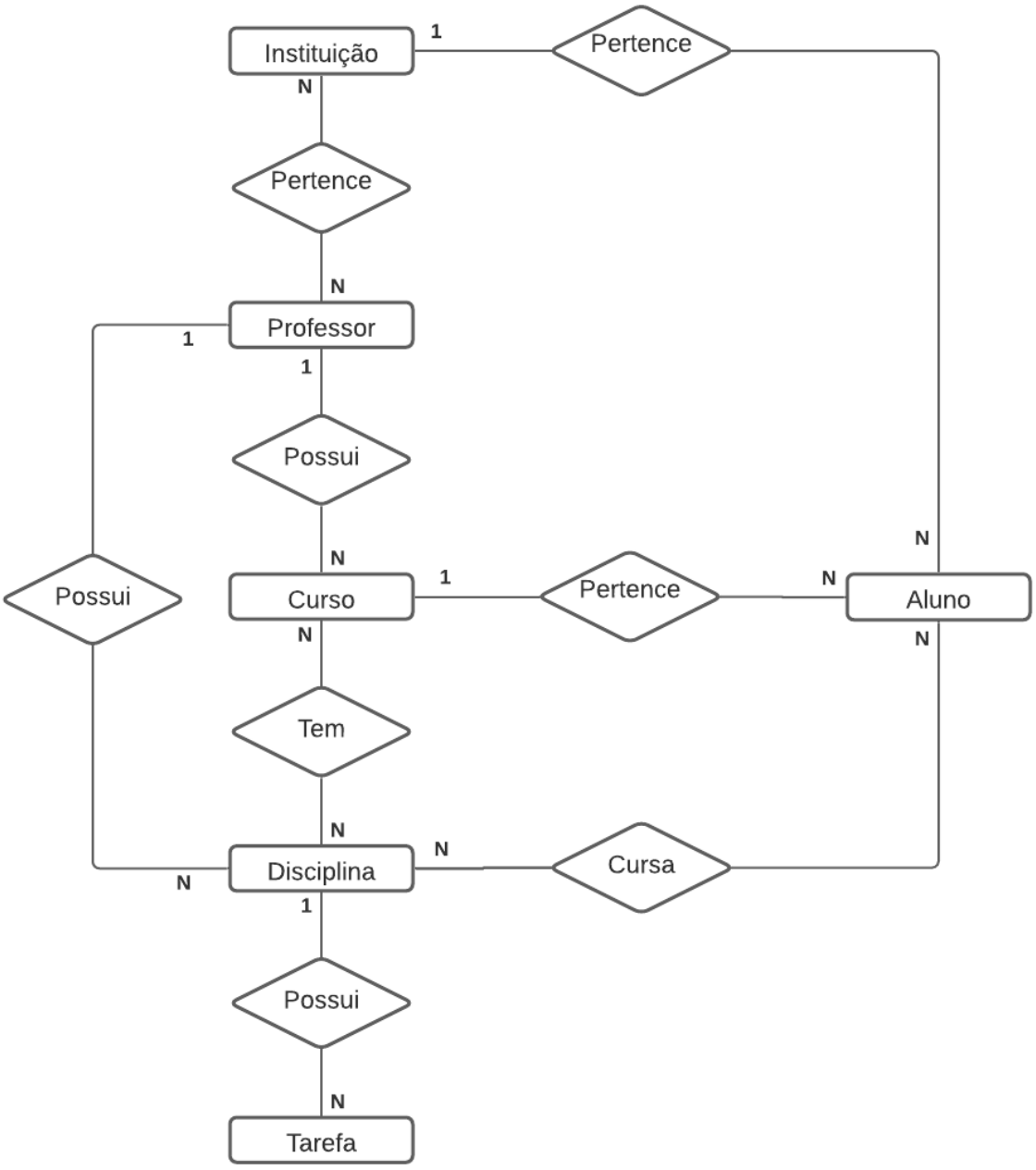
4 DESCRIÇÃO DO SISTEMA

O sistema refere-se a um ambiente virtual de aprendizagem capaz de utilizar os relacionamentos textuais produzidos pelos Mapas Conceituais(AGUIAR, 2013) dos discentes para classificar seu nível de cognição. Para isso, o sistema permite ao professor disponibilizar tarefas aos alunos, que serão respondidas através de Mapas Conceituais. O professor define a validade das proposições do Mapa Conceitual do aluno e através do uso de técnicas de Processamento de Linguagem Natural(BARBOSA, 2017), o sistema define um score para o Mapa Conceitual que poderá ser utilizado pelo professor para identificar o nível de conhecimento do aluno sobre o conteúdo da tarefa. Dessa forma, o acompanhamento dos resultados online pelo professor trará um ganho significativo nesse processo e possibilitará análises mais rápidas e aprofundadas sobre o aprendizado dos discentes.

Para implementar todo o sistema e seus relacionamentos, foi utilizado o ***Django Framework*** (DJANGO, 2021), um framework de web server-side escrito em ***Python*** (PYTHON, 2021). *Django* facilita bastante a parte de manipular os dados, uma vez que o desenvolvedor tem acesso ao *django-admin*, uma página onde pode-se manipular todo o banco de forma fácil e prática.

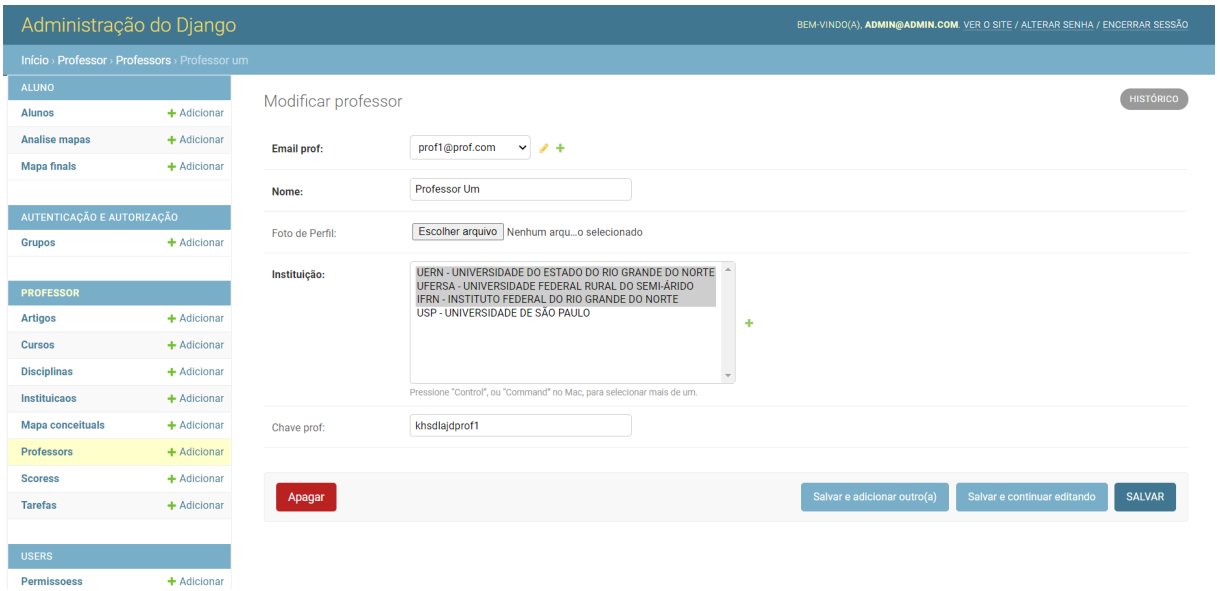
Com o intuito de ilustrar e ter uma melhor noção do funcionamento do sistema, foi criado um *Diagrama de Entidade Relacionamento (ER),* **figura 1,** que ilustra a modelagem do banco de dados. Seguindo essa linha, a **figura 2,** mostra o *django-admin* e como é a visualização dos atributos das tabelas do Banco de Dados, neste exemplo está sendo apresentado a tabela *professor*.

Figura 1: Diagrama ER



Fonte: Autoria própria

Figura 2: Django-admin - Tabela Professor



Fonte: Autoria própria

Como se observa na **figura 1**, o *professor* poderá ter mais de uma instituição vinculada à sua conta, enquanto o *aluno* mantém-se atrelado a apenas uma. Tanto as *instituições* quanto os *cursos* pertencentes podem ser cadastrados por um *professor* caso ainda não existam no sistema, e ficam disponíveis para qualquer outro usuário.

Já o processo de *disciplinas* da-se de forma diferente, onde a disciplina cadastrada pelo *professor* aparece disponível para edição somente para o mesmo. A partir de sua criação, o *professor* deve inserir os alunos que irão participar desta disciplina, observando que somente os alunos cadastrados no sistema irão aparecer para inclusão.

Com o *django*, chamamos de *view* cada função de uma página, sendo que nem sempre é necessário ter uma página visível para utilizarmos a função de uma *view* na aplicação web. Por exemplo, a **figura 3,** ilustra a view que trata o cadastro da disciplina.

Figura 3: View de cadastro de disciplina

```python
@login_required(login_url = '/users/login')
def cadastrardisciplina(request, id):
    if permissao(request):
        my_curso=Curso.objects.get(id=id)
        # se o metodo for post ele insere dados no banco
        form = DisciplinaForm(request.POST) # preenche o formulario com os dados do post
        if form.is_valid(): # verifica se o formulario é valido
            # se for valido vai comecar a salvar o cadastro da disciplina
            cadastrardisciplina = form.save(commit=False) # commit=False, vai parar a inserção de dados e esperar até que mande salvar
            cadastrardisciplina.professor = request.user.professor # a disciplina será cadastrada adicionando o id do usuario logado
            cadastrardisciplina.curso=my_curso
            string=aleatorio(10)+'{}'.format(str(cadastrardisciplina.nome_disciplina))
            cadastrardisciplina.chave_disc = string
            cadastrardisciplina.save()
            return redirect('professor:lista_disciplinas', id)
        else:
            form = DisciplinaForm()
            return render(request, 'professor/cad_disciplina.html', {'form': form, 'my_curso':my_curso})
    else:
        return render(request, 'professor/sem_permissao.html')
```

Fonte: Autoria própria

Toda *view* do sistema possui seu primeiro parâmetro como *request*, que é um objeto HttpRequest. Já o segundo parâmetro é opcional, dependendo do que deseja ser feito. neste exemplo da **figura 3**, temos o parâmetro *id* que irá receber o id da tabela onde se encontra aquela página. Primeiro é verificado se o usuário possui a devida permissão de *professor* para isso, logo em seguida, o objeto *my_curso* irá pegar qual curso está vinculado ao parâmetro *id* da *view*, assim a disciplina é cadastrada dentro daquele curso específico.

O passo seguinte é criar a *tarefa*, onde o *professor* deve passar o título da tarefa, a descrição e por fim enviar seu artigo (ou artigos) que ficará disponível para o *aluno* fazer download do mesmo e usar como base para criação de seu mapa conceitual.

Ainda na parte de *tarefas*, mas como um usuário do tipo *aluno*, a tarefa aparece disponível dentro da disciplina, após abri-la, é mostrado o título e descrição da mesma, seguido dos artigos disponíveis para download e por fim a opção de *Enviar Mapa*. O aplicativo utilizado para criação deste tipo de mapa é o *CmapTools (https://cmap.ihmc.us/cmaptools/)* nele, após o mapa ser finalizado, o discente deve exportá-lo no formato *outiline do cmap.* A **figura 4** ilustra um mapa conceitual no formato *outline* do *CmapTools*.

Figura 4: Saída do Mapa Conceitual no formato Outline

```
PLN
    Baseado em
        Linguagem Natural
            Utiliza
                Processamento Computacional
                    Aplica
                        Técnicas de Machine Learning
                            Descobre
                                Padrões
                                    Utilizam
                                        Algorítmos Específicos
                    Através
                        De abordagems específicas
                            Categorizadas
                                Híbrida
                                    que emprega
                                        Algorítmos Específicos
                    Normaliza
                        Textos
                            Pontuando
                                Palavras Frequentes
                                    Que Reflete
                                        Conceitos Importantes
                                            Validado por
                                                Técnicas
                            Reduzindo
                                Ambiguidades
                            Removendo
                                Stopwords
    Utiliza
        Aprendizagem Significativa
            Baseado em
                Mapas Conceituais
                    Formado por
                        Proposições
                            Refletem
                                Nível Cognitivo
                                    Aprimorando
                                        Metodologias em Sala de Aula
                                            Utilizando
                                                Técnicas
```

Fonte: (NETO, 2019)

Após o *aluno* enviar seu mapa, o mesmo passa por um algoritmo de remontagem que foi desenvolvido por Neto (2019) e atualmente com pequenas modificações minhas para o sistema atual. A **figura 5** apresenta como parte do processo de remontagem é feito.

Figura 5: Função para criar o mapa remontado

```
def main(arq_url):

    with open(arq_url, 'r') as f:
        linhas = f.readlines()

    try:
        string=''
        afirm = afirmacoes(linhas)
        for niv, c1, v, c2 in sorted(afirm, key=lambda a: a[0]): # ordena pelo nivel
            string+='N{} | {} | {} | {}\n'.format(niv, c1, v, c2)

    except Exception as e:
        print(str(e))

    finally:
        return string


def mapa_remontado(caminho):
    variav=main(caminho)
    return variav
```

Fonte: Autoria Própria

A função *main* recebe o mapa e o reescreve, armazenando todo seu conteúdo em uma variável chamada *string*, que posteriormente, o conteúdo desta variável será salvo no banco de dados. Esta função, através da quantidade de *TABS* presente no arquivo, identifica o nível de profundidade do mapa, além de separar cada ligação de uma forma padrão: *NX | 1º Conceito | verbo de ligação | 2º Conceito*. Onde *NX* é o nível de profundidade daquela linha. Para seguir uma sequência, foi utilizado o mapa da **figura 4** para o processo de remontagem, veja a saída desse processo na **figura 6**.

Figura 6: Arquivo de Saída do formato Outline Remontado

```
N0 | PLN | Baseado em | Linguagem Natural
N0 | PLN | Utiliza | Aprendizagem Significativa
N1 | Linguagem Natural | Utiliza | Processamento Computacional
N1 | Aprendizagem Significativa | Baseado em | Mapas Conceituais
N2 | Processamento Computacional | Aplica | Técnicas de Machine Learning
N2 | Processamento Computacional | Através | De abordagems específicas
N2 | Processamento Computacional | Normaliza | Textos
N2 | Mapas Conceituais | Formado por | Proposições
N3 | Técnicas de Machine Learning | Descobre | Padrões
N3 | De abordagems específicas | Categorizadas | Híbrida
N3 | Textos | Pontuando | Palavras Frequentes
N3 | Textos | Reduzindo | Ambiguidades
N3 | Textos | Removendo | Stopwords
N3 | Proposições | Refletem | Nível Cognitivo
N4 | Padrões | Utilizam | Algorítmos Específicos
N4 | Híbrida | que emprega | Algorítmos Específicos
N4 | Palavras Frequentes | Que Reflete | Conceitos Importantes
N4 | Nível Cognitivo | Aprimorando | Metodologias em Sala de Aula
N5 | Conceitos Importantes | Validado por | Técnicas
N5 | Metodologias em Sala de Aula | Utilizando | Técnicas
```

Fonte: (NETO, 2019)

Com a possibilidade de utilizarmos códigos em ***python,*** o arquivo enviado pelo aluno passa por duas *views,* onde a primeira - *view tarefa* - exerce o trabalho de renomear esse arquivo e o salva com todos os dados necessários - disciplina, tarefa e nome do professor. Já a segunda - *view gerando_mapa* - não existe uma página web para a mesma, é tudo feito de forma interna, onde são criados novos dados em duas tabelas - *MapaFinal* e *AnaliseMapa* - a **figura 7** a seguir apresenta esta *view*.

Figura 7: View gerando_mapa

```
def gerando_mapa(request, chave):
    if perm_aluno(request):
        mapa = MapaConceitual.objects.get(chave_mapa=chave)

        string = str(mapa)
        caminho_mapa = caminho_raiz+'map/{}'.format(string)
        abrir = open(caminho_mapa)
        mapa_original = abrir.read()
        abrir.close()
        saida = mapa_remontado(caminho_mapa)
        saida=saida[:-1]

        form2 = MapaFinalForm(request.POST or None)
        f=form2.save(commit=False)
        f.mapa_aluno = mapa
        f.saida=saida
        f.recebido=mapa_original
        f.valida=''     ## ESTÃO COMO STRINGS VAZIAS APENAS PARA
        f.invalida=''   ## O CAMPO DEIXAR DE SER NONE.
        f.save()

        remontado = [x for x in saida.split('\n')]

        for i in remontado:
            form = analise_mapaForm (request.POST or None)
            gerando_mapa = form.save(commit=False)
            gerando_mapa.mapa_do_aluno = mapa
            gerando_mapa.linha = i
            gerando_mapa.save()
```

Fonte: Autoria Própria

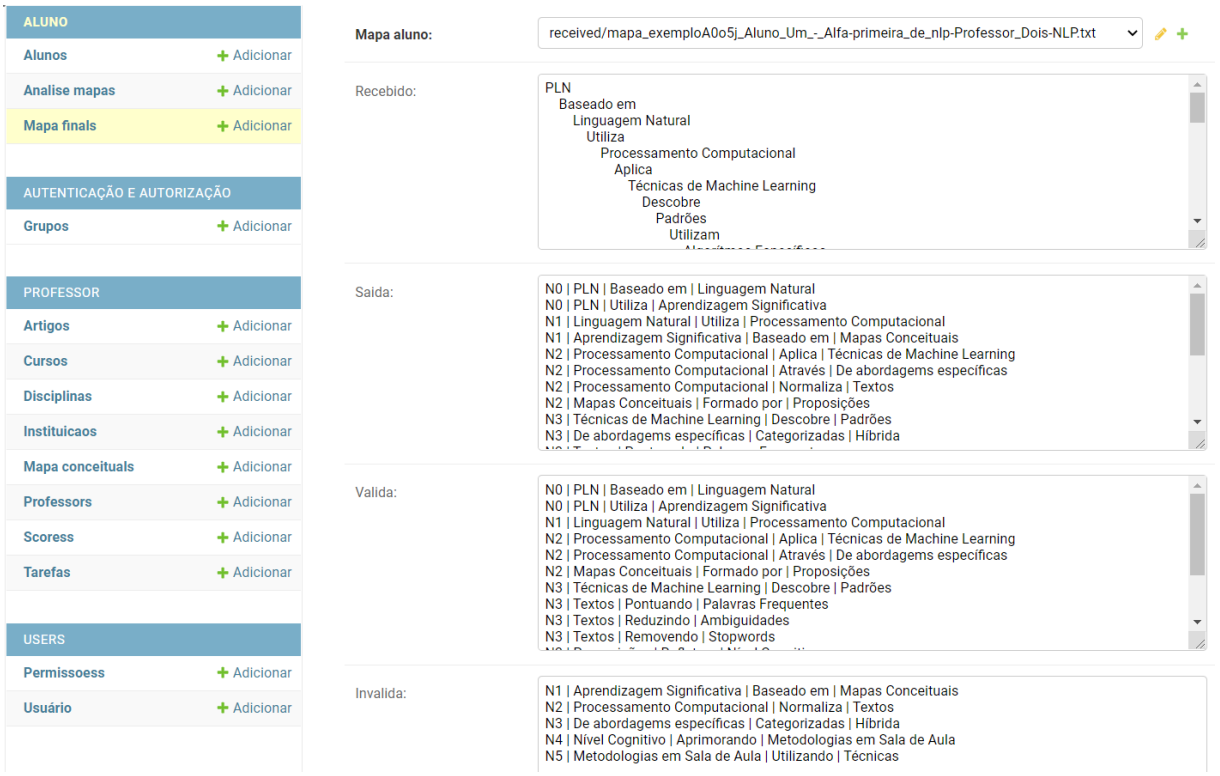
Nela, são gerados dois formulários, onde o primeiro - *form2* - é do tipo *MapaFinalForm*, este objeto funciona criando uma nova tupla em uma tabela chamada de *mapa final* onde os atributos da mesma são:

- o mapa do aluno - O nome do mapa que foi criado na *view tarefa* se torna uma chave estrangeira nesse atributo;
- arquivo recebido - Esse atributo é preenchido com o conteúdo do mapa que é enviado pelo aluno;

- arquivo de saida - Preenchido com todo o conteúdo do arquivo de saida remontado;
- proposições válidas - As proposições que o professor considerar válidas
- proposições inválidas - As proposições que o professor considerar inválidas

A **figura 8** apresenta o formato dessa tabela mais perceptível.

Figura 8: Django-admin - Tabela MapaFinal



Fonte: Autoria própria

O segundo formulário - *form* - aparece mais abaixo dentro de um laço de repetição, onde, sua função é criar várias tuplas na tabela *AnaliseMapa*, cada tupla criada, é formada por cada linha do campo "saída" da **figura 8**. O objetivo dessa tabela é apresentar para o professor essas proposições de forma separada e assim o mesmo poder avaliá-las. Veja a seguir na **figura 9** um modelo da tabela *AnaliseMapa*.

Figura 9: Django-admin - Tabela AnaliseMapa

LINHA	MAPA DO ALUNO	AVALIADA	ID
N0 | PLN | Baseado em | Linguagem Natural	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	757
N0 | PLN | Utiliza | Aprendizagem Significativa	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	758
N1 | Aprendizagem Significativa | Baseado em | Mapas Conceituais	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	760
N1 | Linguagem Natural | Utiliza | Processamento Computacional	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	759
N2 | Mapas Conceituais | Formado por | Proposições	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	764
N2 | Processamento Computacional | Aplica | Técnicas de Machine Learning	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	761
N2 | Processamento Computacional | Através | De abordagems específicas	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	762
N2 | Processamento Computacional | Normaliza | Textos	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	763
N3 | De abordagems específicas | Categorizadas | Híbrida	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	766
N3 | Proposições | Refletem | Nível Cognitivo	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	770
N3 | Textos | Pontuando | Palavras Frequentes	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	767
N3 | Textos | Reduzindo | Ambiguidades	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	768
N3 | Textos | Removendo | Stopwords	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	769
N3 | Técnicas de Machine Learning | Descobre | Padrões	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	765
N4 | Híbrida | que emprega | Algorítmos Específicos	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	772
N4 | Nível Cognitivo | Aprimorando | Metodologias em Sala de Aula	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	774
N4 | Padrões | Utilizam | Algorítmos Específicos	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	771
N4 | Palavras Frequentes | Que Reflete | Conceitos Importantes	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	773
N5 | Conceitos Importantes | Validado por | Técnicas	received/mapa_exemploA0o5j_Aluno_Um_-_Alfa-primeira_de_nlp-Professor_Dois-NLP.txt	✓	775

Fonte: Autoria Própria

Em resumo, o arquivo remontado completo é armazenado na *tabela mapa_final* e suas linhas separadamente na tabela *AnaliseMapa,* que será mostrado para o professor na página da aplicação com os dados do aluno que o enviou. O *professor* deverá avaliar cada proposição (linha) como uma proposição **válida** ou **inválida,** que será armazenada em suas respectivas listas. Segundo Neto (2019) *"as proposições contidas na "Lista Inválida" referem-se às orações que não condizem com o contexto exposto no conteúdo programático."*. Este processo de avaliação é necessário para gerar o *Score* final do mapa.

O *Score* é gerado a partir da avaliação das proposições do aluno. Após todas as proposições avaliadas, as mesmas são passadas como parâmetro de uma função nomeada **re_final**, que por sua vez chama outras funções para avaliar os seguintes critérios:

- Relacionamento - Se a estrutura está correta, ou seja, se realmente o primeiro elemento é um conceito, o segundo elemento um verbo de ligação, e o terceiro elemento um conceito novamente. São atribuídos 3 pontos para cada relacionamento;

- Ramificações: A pontuação da ramificação é atribuída de forma específica dependendo do nível onde ocorra a ramificação, sendo:
    - Nível 1: 5 pontos
    - Nível 2: 4 pontos
    - Nível 3: 3 pontos
    - Nível 4: 2 pontos
    - Nível 5 em diante: 1 ponto;
- Hierarquia - Através do "*NX*" composto antes da proposição em si, o algoritmo identifica a profundidade daquele mapa, e se todas as proposições até ele também foram válidas. Para cada nível de hierarquia são atribuídos 5 pontos.

A função **re_final** (que seria Resultado Final), é uma função python que possui três parâmetros, onde o primeiro (a_out) recebe o *mapa remontado* do aluno, o segundo (p_val) são todas as *proposições válidas* e por fim, todas as *proposições inválidas (p_inval)*. Todos esses dados são retirados de seus respectivos campos da **figura 8**. Veja a seguir na **figura 10** como foi pensada a função *re_final*.

Figura 10: Função re_final

```
def re_final(a_out, p_val, p_inval):
    global resultado_final
    props = [x for x in a_out.split('\n')]
    props_validas = [x for x in p_val.split('\n')]
    props_invalidas = [x for x in p_inval.split('\n')]

    sp = scores(props, props_validas)
    sn = scores(props, props_invalidas)

    resultado_final+='Score Positivo = {}\n'.format(sp)
    resultado_final+='Score Negativo = {}\n'.format(sn)
    resultado_final+='Score Total = {}\n'.format(sp-sn)

    lista=[x for x in resultado_final.split('\n')]

    lista.insert(0, '------- ..:: PROPOSICOES VALIDAS ::.. -------')
    lista.insert(4,'------- ..:: PROPOSICOES INVALIDAS ::.. -------')
    lista.insert(8,'------- ..:: Score Final ::.. -------')

    resultado_final = ''
    return lista
```

Fonte: Autoria própria

Nela, existem duas variáveis importantes que são ***sp*** e ***sn*** - Score positivo e Score negativo, respectivamente - ambas chamam a função ***scores*** (NETO, 2019), com a diferença que na primeira é passado as proposições válidas, e na segunda as inválidas. Com isso, esta função consegue verificar todos os requisitos necessários para obter a avaliação do mapa. A **figura 11** apresenta a função *scores* criada por Neto (2019).

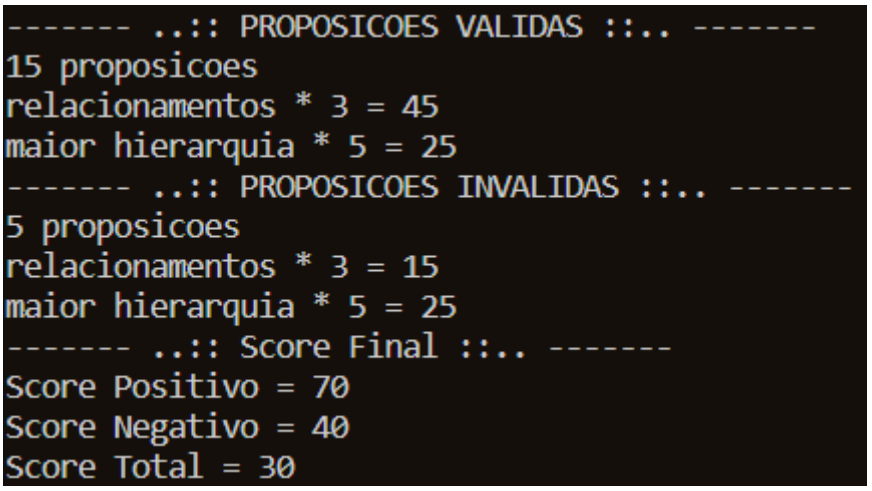
Figura 11: Função score

```
def scores(all_props, props):
    relacionamentos = len(props) * 3
    hierarquia = maior_hierarquia(props) * 5
    ramificacoes = ramificacoes_scores(all_props)

    ramificacoes_unicas = set()
    for prop in props:
        nv, c1, v, c2 = prop.split('|')
        ramificacoes_unicas.add(f'{nv} | {c1}'.lower())

    score = 0
    for rami in ramificacoes_unicas:
        if rami in ramificacoes:
            nv = ramificacoes[rami]
            if nv == 1:
                score += 5
            elif nv == 2:
                score += 4
            elif nv == 3:
                score += 3
            elif nv == 4:
                score += 2
            elif nv >= 5:
                score += 1

    return relacionamentos + hierarquia + score
```

Fonte: (NETO, 2019)

Por fim, é retornado para o *professor* os dados do *Score* final, apenas apresentando os dados mais relevantes, a **figura 12** ilustra a forma da pontuação final, gerada pela avaliação do professor a partir do arquivo remontado da **figura 6.**

Figura 12: Score Final



Fonte: Autoria própria

5 RESULTADOS

WEB-SAM foi finalizado e desenvolvido visando a necessidade de construção de meios tecnológicos que consigam auxiliar o professor nas tarefas inerentes ao processo de ensino e aprendizagem, onde tais meios permitam o acesso detalhado ao nível de conhecimento do discente acerca de determinado tema. Assim, proporcionando um ambiente virtual de aprendizagem prático e intuitivo.

Entretanto, ao longo do desenvolvimento foi observado que existem algumas funcionalidades que poderiam ser empregadas posteriormente, como por exemplo a implementação de um NLP para a avaliação de proposições, e/ou uma interface mais intuitiva para melhor experiência do usuário.

Contudo, o sistema WEB-SAM pode ser considerado uma ferramenta que facilita o trabalho do professor, onde viabiliza o mesmo a possibilidade de identificar o nível de cognição do discente em relação ao assunto específico. A identificação do conhecimento consegue ser prática, utilizando processos e as ligações textuais/contextuais criadas pelos alunos por intermédio de ferramentas de desenvolvimentos de Mapas Conceituais, no qual, se dá uma forma simples de avaliação.

REFERÊNCIAS

NETO, R. N. Sistema Classificador de Mapas Conceituais: Uma Arquitetura Computacional Baseada em Processamento de Linguagem Natural. 2019. Dissertação (Mestrado) - Universidade do Estado do Rio Grande do Norte, Mossoró-RN, 2019.

AGUIAR, J. G. de; CORREIA, P. R. M. Como fazer bons mapas conceituais? estabelecendo parâmetros de referências e propondo atividades de treinamento. Revista Brasileira de pesquisa em Educação em Ciências, v. 13, n. 2, p. 141–157, 2013.

BARBOSA, J. L. N. et al. Introdução ao processamento de linguagem natural usando python. III ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ, v. 1, p. 336–360, 2017.

PYTHON: The Python Tutorial. 3.4.9. [*S. l.*], [S. D.]. Disponível em: https://docs.python.org/3/tutorial/. Acesso em: 22 abr. 2021.

DJANGO: Why Django?. 3.2. [*S. l.*], [S. D.]. Disponível em: https://www.djangoproject.com/start/overview/. Acesso em: 22 abr. 2021.

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512021000805-7**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 19/04/2021, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Desenvolvimento de um sistema web para classificação do nível cognitivo do discente com base em um Mapa Conceitual

Data de publicação: 19/04/2021

Data de criação: 19/04/2021

Titular(es): UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - UERN

Autor(es): JOYCE CLAINE PEREIRA FREITAS; ROMMEL WLADIMIR DE LIMA; RAIMUNDO NONATO BEZERRA NETO; RENATO MONTEIRO NICACIO

Linguagem: PYTHON

Campo de aplicação: ED-01; ED-04

Tipo de programa: FA-01

Algoritmo hash: OUTROS

Resumo digital hash: fd26f86c2f5fb536c0fff6086894fd84

Expedido em: 04/05/2021

Aprovado por:
Carlos Alexandre Fernandes Silva
Chefe da DIPTO