

UNIVERSIDADE FEDERAL FLUMINENSE

LUIS PAULO FERREIRA HUMELINO

MYRIAM MARTINS DE OLIVEIRA

**DESENVOLVIMENTO DE SOFTWARE WEB PARA
GERENCIAMENTO DAS MÍDIAS DIGITAIS DOS ALUNOS
DE GRADUAÇÃO UFF**

Niterói

2019

**LUIS PAULO FERREIRA HUMELINO
MYRIAM MARTINS DE OLIVEIRA**

**DESENVOLVIMENTO DE SOFTWARE WEB PARA
GERENCIAMENTO DAS MÍDIAS DIGITAIS DOS ALUNOS
DE GRADUAÇÃO UFF**

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

**Orientador:
Flávio Luiz Seixas**

**NITERÓI
2019**

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

H921d Humelino, Luis Paulo Ferreira
DESENVOLVIMENTO DE SOFTWARE WEB PARA GERENCIAMENTO DAS
MÍDIAS DIGITAIS DOS ALUNOS DE GRADUAÇÃO UFF / Luis Paulo
Ferreira Humelino, Myriam Martins de Oliveira ; Flávio Luiz
Seixas, orientador. Niterói, 2019.
65 f. : il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia
de Sistemas de Computação)-Universidade Federal Fluminense,
Instituto de Computação, Niterói, 2019.

1. Base de dados. 2. Produção intelectual. I. Oliveira,
Myriam Martins de. II. Seixas, Flávio Luiz, orientador. III.
Universidade Federal Fluminense. Instituto de Computação.
IV. Título.

CDD -

**LUIS PAULO FERREIRA HUMELINO
MYRIAM MARTINS DE OLIVEIRA**

**DESENVOLVIMENTO DE SOFTWARE WEB PARA
GERENCIAMENTO DAS MÍDIAS DIGITAIS DOS ALUNOS
DE GRADUAÇÃO UFF**

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Niterói, ____ de _____ de 2019.

Banca Examinadora:

Prof. Flávio Luiz Seixas, DSc. – Orientador
UFF – Universidade Federal Fluminense

Prof. Leonardo Vasconcelos, Msc. – Avaliador
UFF - Universidade Federal Fluminense

Em memória do meu pai, Ozorio de Oliveira.
Certamente, meu maior exemplo de superação. Minha eterna gratidão.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ser o Senhor da minha vida e ter me conduzido durante toda essa jornada me dando forças para chegar até aqui. Agradeço a minha família, principalmente, minha mãe que sempre me deu a base para que eu pudesse ir em busca dos meus objetivos.

Agradeço a UFF e ao CEDERJ pelo comprometimento na entrega de um ensino de qualidade e pelas experiências vividas. Ao nosso orientador, Flávio Seixas, por ter topado ser nosso guia e ter instruído como deixar este trabalho apresentável.

Aos meus amigos que cultivei durante essa caminhada, agradeço demais ao Marcos Vinicius Calazans e Lais Betini o companheirismo, as trocas, os aprendizados. Um agradecimento especial ao Fabio Victorino, que além de um grande amigo, foi meu guru, meu mestre, meu incentivador, meu orientador, o cara que pegou na minha mão e me ajudou a chegar até aqui também.

E por fim, agradeço a minha dupla, Luis Paulo Humelino, que aceitou dividir essa carga comigo, que desde o início tivemos fácil entendimento e acordo sobre o que fazer e como fazer. Foi um prazer ter te conhecido e desenvolvido esse projeto com você. Muito obrigada por tudo!

Myriam Oliveira

“O sucesso é a soma de pequenos esforços
repetidos diariamente.”

Robert Collier

RESUMO

Este trabalho tem o objetivo de oferecer uma ferramenta para aprimorar o processo final da digitalização dos dossiês dos alunos de graduação da UFF realizado pelo LAboratório REprográfico (LARE). Todo documento digitalizado fica descentralizado em diversos computadores, sendo preciso que uma pessoa reúna esses dados e de forma manual transfira-os para máquinas localizadas na PRÓ-Reitoria de GRAduação (PROGRAD), onde serão objetos de pesquisa. Desse modo é necessário fazer *backups* semanais e acompanhamento por planilhas para ciência do que foi atualizado. Sendo assim, o objetivo é desenvolver um *software web* capaz de integrar a atividade realizada no LARE com a necessidade da PROGRAD, facilitando o arquivamento e consulta desses documentos. O banco MySQL será o responsável por gerenciar as informações e a Web será o meio de acesso, tornando a comunicação entre LARE e PROGRAD um processo automatizado, evitando a duplicidade de dados, diminuindo os possíveis riscos de perda de dados durante a transferência, além de tornar a consulta uma ação simples e rápida.

Palavras-chaves: banco de dados, gestão, sistemas de informação e software web.

ABSTRACT

The purpose of this paper is to improve the final process of digitalization of UFF undergraduate students' files carried out by the Reprographic Laboratory (LARE), because everything that is digitized is decentralized in several computers, requiring that a person gather this data and manually transfer them to the computers located at the Dean of Graduation (PROGRAD), where they will be research objects. This requires weekly backups and spreadsheet tracking to keep track of what has been updated. Thus, the goal is to develop web software capable of integrating the activity performed in LARE with the need of PROGRAD, facilitating the archiving and consultation of these documents. The MySQL database will be responsible for managing the information and the Web will be the means of access, making communication between LARE and PROGRAD an automated process avoiding duplication of data, reducing the potential risks of data loss during transfer, and making the query a simple and quick action.

Key words: database, information systems and software web.

LISTA DE ILUSTRAÇÕES

Figura 1: Diagrama de atividades PROGRAD	19
Figura 2: Estrutura do diretório de documentos digitalizados	20
Figura 3: Diagrama de atividades LARE	21
Figura 4: Diagrama de atividades LARE desejado	22
Figura 5: Diagrama de atividades PROGRAD desejado	23
Figura 6: Diagrama de Casos de Uso	27
Figura 7: Diagrama de Entidade-Relacionamento	38
Figura 8: Tela de <i>login</i>	42
Figura 9: Tela <i>index</i> do sistema	43
Figura 10: Consultando aluno cadastrado	44
Figura 11: Tela Cadastrar Aluno	45
Figura 12: Cadastrando novo aluno	46
Figura 13: Aluno cadastrado com sucesso.....	47
Figura 14: Inserindo alunos em massa	48
Figura 15: Código para criação do banco de dados e tabela login	53
Figura 16: Criação das tabelas Polo, Curso e Aluno	53
Figura 17: Cadastrando os polos existentes na base de dados	54
Figura 18: Populando a tabela Cursos – parte 1	54
Figura 19: Populando a tabela Cursos – parte 2	55
Figura 20: Populando a tabela Cursos – parte 3	55
Figura 21: Criação das <i>views</i>	56
Figura 22: Classe “user.php”	57
Figura 23: Página de <i>login</i> “login.php”	58
Figura 24: Funções “isInputNumber()” e “isMatricula()”	58
Figura 25: Tela inicial “index.php” - parte 1	59
Figura 26: Tela inicial “index.php” - parte 2	59
Figura 27: Página “listar.php” - parte 1	60
Figura 28: Página “listar.php” - parte 2	60
Figura 29: Página “abrir_arquivo.php”	61
Figura 30: Função “isInputName()”	61
Figura 31: Função “cadastrarAluno()” e validação de <i>user</i> logado	62

Figura 32: Página “cadastrar.php” - parte 1.....	63
Figura 33: Página “cadastrar.php” - parte 2.....	63
Figura 34: Página “search.php” - parte 1.....	64
Figura 35: Página “search.php” - parte 2.....	64
Figura 36: Archivo “conn.php”	65

LISTA DE TABELAS

Tabela 1: Tabela dos requisitos Funcionais	24
Tabela 2: Tabela dos requisitos Não Funcionais.....	25
Tabela 3: Tabela dos requisitos de Domínio do sistema	26
Tabela 4: Tabela Caso de Uso 1 – Fazer Login	28
Tabela 5: Tabela Caso de Uso 2 – Cadastrar Aluno	28
Tabela 6: Tabela Caso de Uso 3 – Inserir em Massa.....	30
Tabela 7: Tabela Caso de Uso 4 – Consultar Aluno.....	31
Tabela 8: Dicionário de Dados – Tabela login	38
Tabela 9: Dicionário de Dados – Tabela polo.....	39
Tabela 10: Dicionário de Dados – Tabela curso.....	39
Tabela 11: Dicionário de Dados – Tabela aluno.....	40

LISTA DE ABREVIATURAS E SIGLAS

PROGRAD – Pró-Reitoria de Graduação
LARE – Laboratório Reprográfico
UFF – Universidade Federal Fluminense
SDC – Superintendência de Documentação
AFD – Assentamento Funcional Digital
SGBD – Sistema Gerenciador de Banco de Dados
CSS – Cascading Style Sheets
PHP – HyperText Preprocessor
SQL – Structured Query Language
HTTP – HyperText Transfer Protocol
HTML – HyperText Markup Language
W3C – World Wide Web Consortium
SSL – Secure Socket Layer
FTP – File Transfer Protocol
HD – Hard Disk

SUMÁRIO

1	INTRODUÇÃO	16
2	DESCRIÇÃO DO CENÁRIO ATUAL	18
2.1	PROGRAD	18
2.2	LARE	20
2.3	Cenário desejado	22
3	ANÁLISE DE SISTEMAS	24
3.1	Requisitos Funcionais	24
3.2	Requisitos Não Funcionais	25
3.3	Requisitos de Domínio	26
3.4	Estudos dos casos de uso	27
3.4.1	Detalhamento dos Casos de Uso	28
4	ESTRUTURAS DE PROGRAMAÇÃO UTILIZADAS	32
4.1	HTML 5	32
4.2	CSS	33
4.3	Javascript	34
4.4	PHP	34
4.5	MySQL	35
4.6	Servidor HTTP Apache	35
5	DESENVOLVIMENTO DO SOFTWARE E INTERFACE	37
5.1	Diagrama de entidade-relacionamento	37
5.2	Criando o banco de dados	40
5.3	Descrição da implementação das telas do sistema	41
5.3.1	Tela de <i>Login</i>	41
5.3.2	Tela Inicial	42
5.3.3	Tela para cadastro de um novo aluno	45
5.3.4	Tela para inserção de dados em massa	47
5.3.5	Arquivo "conn.php"	49
6	CONCLUSÃO	50
6.1	Trabalhos Futuros	50
	REFERÊNCIAS	51
	APÊNDICE A – CRIAÇÃO DO BANCO DE DADOS	53

APÊNDICE B – CODIFICAÇÃO DO SOFTWARE	57
--	----

1 INTRODUÇÃO

A documentação do aluno de graduação é recebida pela universidade no ato da matrícula e encaminhada para o LARE para ser digitalizada. O processo de digitalização é dividido entre diversos funcionários do LARE. Após a digitalização, essas mídias digitais são unificadas e centralizadas em um único lugar para serem enviadas a PROGRAD, que em momento oportuno, fará as devidas consultas para tomada de decisão pertinente ao aluno.

Esse projeto, chamado de Assentamento Funcional Digital (AFD), visa trazer eficiência ao processo decisório, por meio de ferramentas que possibilitem disponibilizar a informação num curto espaço de tempo e com uma economia racional de espaço físico.

Observando de perto todo o processo, identificamos que seria possível desenvolver uma ferramenta simples e eficiente para encurtar algumas etapas, centralizar as informações em um banco de dados acessível aos dois setores responsáveis pelo manuseio desses dossiês e facilitar a busca dos dados quando necessário. Assim, visamos neste projeto a automação das etapas de arquivamento destas mídias digitais, facilitando o repasse e consulta desses dados, através de um sistema *web* gerenciador de banco de dados.

O primeiro passo é entender as regras do negócio e fazer o levantamento dos requisitos, identificando todas as necessidades e funcionalidades que o sistema terá. Depois estudar os casos de uso analisando todos os possíveis cenários dentro do sistema. O desenvolvimento utilizará o sistema gerenciador de banco de dados, MySQL, para administrar o banco que será responsável pelo armazenamento das mídias digitais, eliminando a necessidade do backup semanal para centralizar esses dados. A implementação será através das ferramentas HTML, CSS, PHP e JavaScript de modo a tornar o banco de dados disponível para que usuários autenticados possam fazer consultas *online*, completando a eficácia que buscamos com o desenvolvimento desse projeto.

Este trabalho está organizado da seguinte forma: o Capítulo 2 descreve o cenário atual detalhando as atividades e rotina do LARE e PROGRAD na organiza-

ção da documentação dos alunos; O capítulo 3 descreve o levantamento e análise de requisitos, estruturação dos casos de uso e seus diagramas; O capítulo 4 mostra a conceituação das metodologias aplicadas no desenvolvimento; O capítulo 5 documenta o desenvolvimento do presente estudo, que compreende o diagrama de entidade-relacionamento, detalhamento da codificação explicando seu funcionamento e a interface com os usuários; O capítulo 6 é a conclusão deste trabalho e sugestões para trabalhos futuros e os Apêndices A e B mostram a codificação do sistema.

2 DESCRIÇÃO DO CENÁRIO ATUAL

Este capítulo tem o objetivo de retratar como é feito o tratamento da documentação dos alunos de graduação pela PROGRAD e LARE. A Seção 2.1 explica o que a PROGRAD representa e a importância da digitalização dentro da administração desses dossiês. Na Seção 2.2 relata o papel do LARE no processo de digitalização. E na Seção 2.3 os benefícios que o sistema trará na tramitação dessa documentação.

2.1 PROGRAD

A Pró-Reitoria de Graduação (PROGRAD) é responsável pela implantação e pelo acompanhamento das políticas de ensino de graduação nas modalidades presencial e à distância. Em suma, a PROGRAD tem como foco o estudante e sua formação.

Em função disso, a geração de documentos dessa Pró-Reitoria é elevada e o acesso para a tomada de decisões se torna mais premente, visto que o produto final da PROGRAD faz parte da atividade final da Universidade Federal Fluminense. Nesse sentido, a digitalização entra como mecanismo subsidiador garantindo agilidade no processo decisório, revisão e padronização de armazenamento de dados. Permitindo maior qualidade e produtividade ao serviço público; preservação de documentos que integram o patrimônio documental; redução da massa documental, otimizando os espaços físicos e promovendo economia de recursos públicos.

Através da Figura 1, que mostra o processo atualmente, podemos observar no cenário atual que existe uma rotina maior e duplicidade de dados, pois implica que tanto o LARE quanto a PROGRAD mantenham os mesmos arquivos em ambientes distintos.

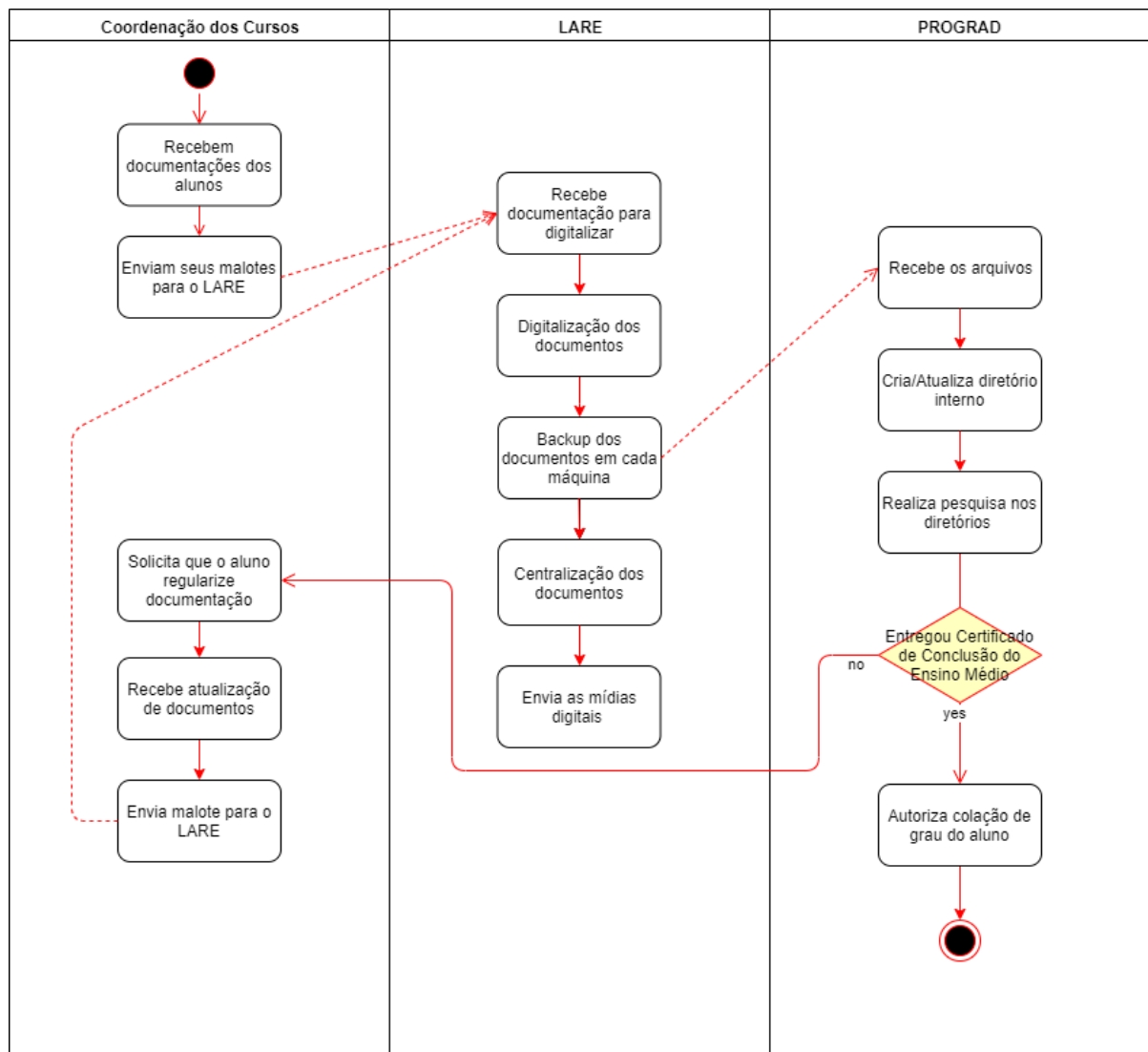


Figura 1: Diagrama de atividades PROGRAD

O que vemos é um processo que foi criado para trazer agilidade à consulta dos dossiês dos alunos, porém, devido à falta de um sistema que integre as atividades realizadas no LARE com a PROGRAD, ainda temos procedimentos sendo realizados de forma manual, como a cópia, *backup* e repasse, que retardam a execução do processo completo.

2.2 LARE

A Coordenação de ARquivos (CAR), subordinada à Superintendência de Documentação (SDC), é o órgão centralizador da documentação da Universidade. O Laboratório REprográfico (LARE) é responsável pela digitalização de documentos e está diretamente vinculado à SDC.

Ou seja, o LARE será o principal agente do sistema; inserindo, mantendo e excluindo os dados quando necessário. Hoje, esse trabalho ocorre da seguinte maneira: criam-se os diretórios (conforme Figura 2), realizam-se as digitalizações dos documentos direcionando os arquivos para as pastas correspondentes. Depois é necessário que uma pessoa vá em cada computador que realizou as digitalizações e faça um *backup* reunindo os dados em um único dispositivo e, periodicamente, as atualizações são enviadas para a PROGRAD via HD externo e/ou pen drives.

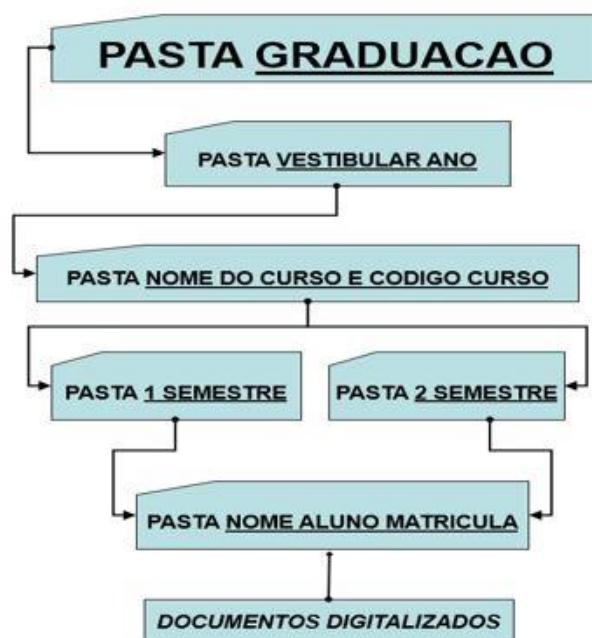


Figura 2: Estrutura do diretório de documentos digitalizados

Existem ainda, situações em que o LARE recebe uma requisição de pesquisa a um determinado dossiê, e como somente um computador detém o *backup* completo de todo o acervo digitalizado, fica restrito a somente uma pessoa a realização dessa tarefa. Se a requisição vem completa de detalhes chega-se mais facilmente na pasta do aluno, mas caso venha faltando uma informação como curso, semestre, ano, o retorno a essa pesquisa será mais demorada, dependendo da velocidade do processador em questão. A Figura 3 mostra o processo descrito acima realizado pelo LARE.

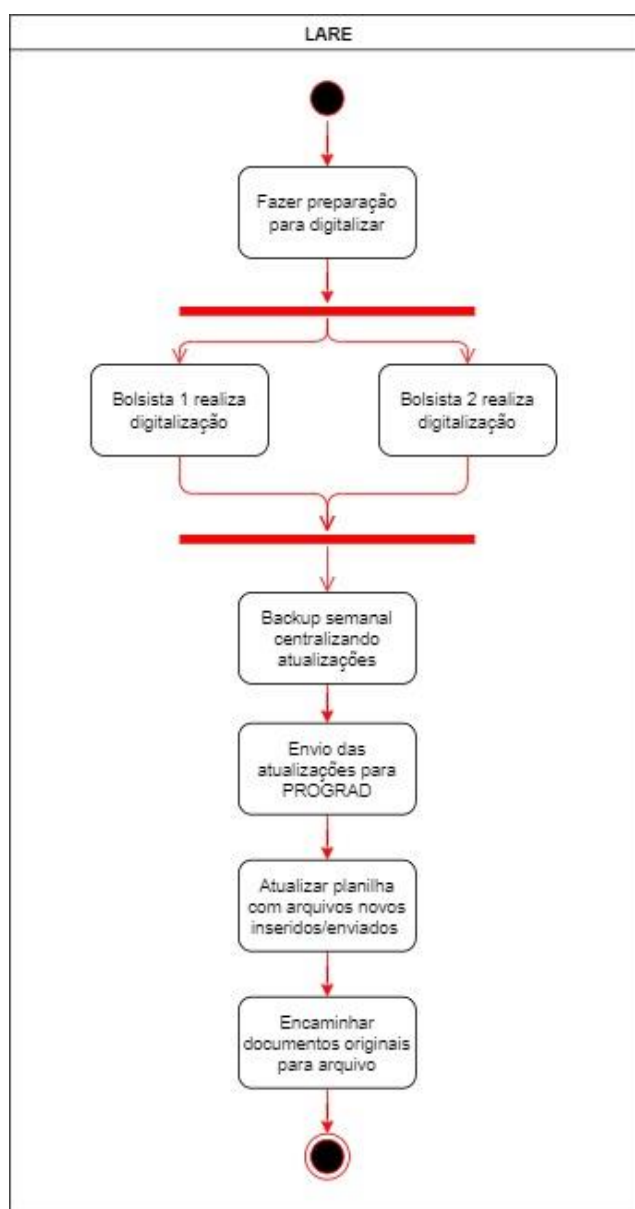


Figura 3: Diagrama de atividades LARE

2.3 CENÁRIO DESEJADO

No cenário desejado as informações são centralizadas em uma única base de dados visível e acessível aos dois setores (vide Figura 4), evitando assim uma replicação e compartilhamento manual suscetível a falhas, além de encurtar o caminho na comunicação entre LARE e PROGRAD, tornando o processo muito mais ágil.

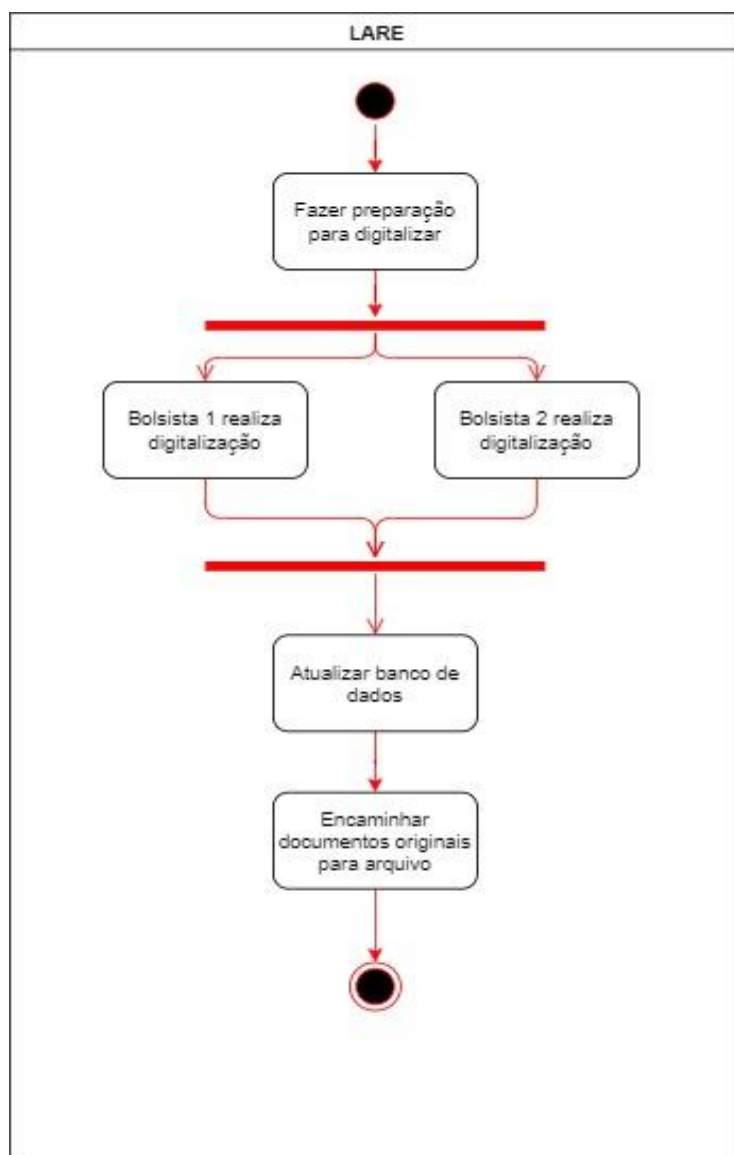


Figura 4: Diagrama de atividades LARE desejado

Uma das grandes facilidades da aplicação que não está muito visível nos diagramas é a consulta a um dado documento. Antes a pesquisa era feita seguindo toda a estrutura de pastas (vide Figura 2) até chegar no aluno desejado. Agora, basta digitar a matrícula do aluno na página de consulta para essa varredura ser feita automaticamente e nos mostrar em segundos o arquivo procurado e todos os dados referente a este aluno. O processo de consulta pode ser visto na figura 5.

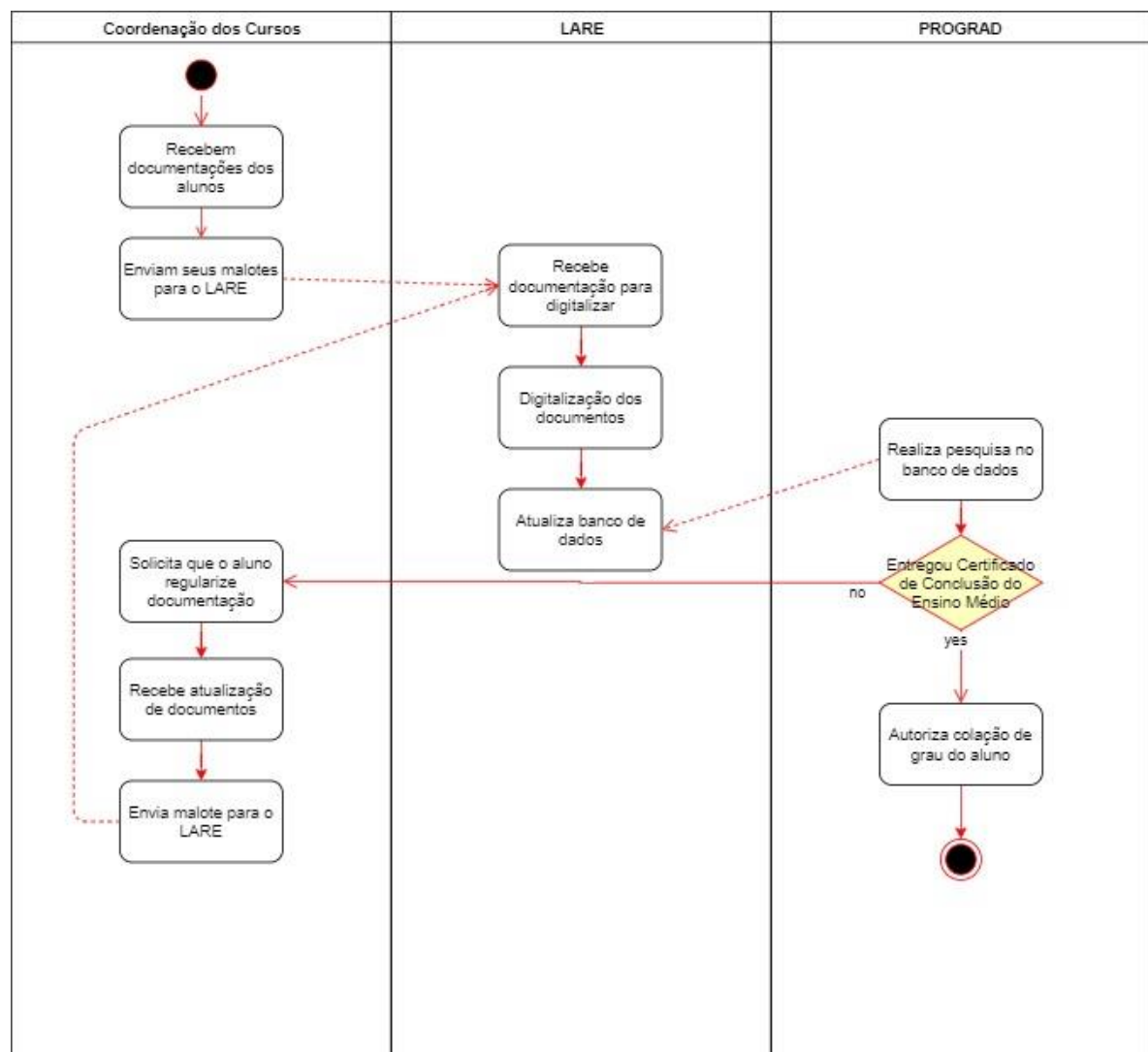


Figura 5: Diagrama de atividades PROGRAD desejado

3 ANÁLISE DE SISTEMAS

Neste capítulo serão descritos os requisitos funcionais, não funcionais, de domínio e casos de uso, etapas da análise de sistemas. Está organizando nas seguintes subseções: a Seção 3.1 é descrito os requisitos funcionais, as funcionalidades que o sistema deve ter, já na Seção 3.2 é a vez dos requisitos não funcionais, ou seja, relacionados a exigência técnica de ambiente, na Seção 3.3 são mostrados os requisitos de domínio, derivados do domínio da aplicação e na Seção 3.4 são feitos os estudos de casos de uso.

3.1 REQUISITOS FUNCIONAIS

São requisitos diretamente ligados a funcionalidade do *software*, descreve as ações que o sistema deve executar. Na Tabela 1, observada abaixo, vemos os requisitos funcionais do sistema.

Tabela 1: Tabela dos requisitos Funcionais

ID	DESCRIÇÃO
RF1	O sistema deve permitir ao usuário cadastrar novos alunos.
RF2	O sistema deve permitir ao usuário a consulta de alunos cadastrados.
RF3	O sistema deve identificar através da matrícula do aluno qual ano/semestre de ingresso e o curso.
RF4	O sistema deve permitir ao usuário fazer o <i>upload</i> de mídias digitais.
RF5	O sistema deve realizar inserções em massa de alunos, de forma automática, através da leitura a pastas específicas com a documentação dos alunos.
RF6	O sistema deve permitir o <i>download</i> da documentação do aluno ao realizar uma consulta.
RF7	O sistema deverá validar o campo da matrícula.

3.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais expressam condições que o software deve atender ou qualidades específicas que ele deve ter, são restrições impostas ao sistema. Na Tabela 2, podemos observar os requisitos não funcionais pertencentes ao sistema.

Tabela 2: Tabela dos requisitos Não Funcionais

ID	DESCRIÇÃO
RNF1	O sistema deverá ser uma aplicação web.
RNF2	O sistema deve ser desenvolvido na linguagem PHP.
RNF3	O sistema deve utilizar um banco de dados relacional MySQL para gerenciar de forma consistente os dados.
RNF4	Somente usuários autorizados deverão ter acessos as informações.
RNF5	O sistema deverá aceitar para a documentação do aluno, apenas arquivos no formato “.tif” ou “.tiff” pois são os formatos gerados nas digitalizações.
RNF6	Deverá ser dada permissão de acesso aos funcionários lotados nos LARE e PROGRAD, pois são os setores responsáveis pela documentação armazenada.
RNF7	O sistema deverá ser implantado na intranet da UFF para melhor acessibilidade e segurança.
RNF8	O tempo de resposta da funcionalidade de inserção em massa não deverá ultrapassar 60 segundos.
RNF9	O sistema deve utilizar o XAMPP como servidor independente na plataforma Windows.

RNF10	A interface do software deve ser amigável e responsiva, ou seja, deixando o usuário confortável ao utilizar o sistema, de forma que sua experiência seja fácil.
RNF11	A probabilidade de o sistema estar operacional num instante de tempo determinado deve ser alta.

3.3 REQUISITOS DE DOMÍNIO

São requisitos derivados do domínio da aplicação e descrevem características do sistema e qualidades que refletem o domínio. Podem ser requisitos funcionais novos, restrições sobre requisitos existentes ou computações específicas. Na Tabela a seguir mostramos os requisitos de domínio do sistema em questão.

Tabela 3: Tabela dos requisitos de domínio do sistema

ID	DESCRIÇÃO
RD1	Um aluno poderá ser da categoria, Graduação (alunos da modalidade presencial) ou CEDERJ (alunos da modalidade à distância).
RD2	Para a função de inserção em massa, deverá ser respeitado a estrutura de diretório conforme Figura 2.
RD3	Um aluno é qualquer pessoa que ingressou no vestibular da UFF, presencial ou à distância, já terminou ou não o curso.
RD4	Um aluno no decorrer do curso pode trancar a matrícula, cancelar, mudar de curso, reingressar, e toda essa movimentação do aluno é identificada pelo primeiro dígito da matrícula, referente ao semestre, que terá seu código alterado de acordo com o status do aluno.
RD5	Uma matrícula está atribuída a somente um aluno e um aluno só está atribuído a uma matrícula vigente.
RD6	A matrícula do aluno não deve conter caracteres além de números.

RD7	A matrícula do aluno deverá ter entre 8 e 11 caracteres.
-----	--

3.4 ESTUDOS DOS CASOS DE USO

Após diagnosticado os requisitos funcionais, é possível fazer a representação da interação entre um usuário e o sistema, assim um caso de uso tem a descrição da funcionalidade que será construída no sistema proposto. Desenhar o processo de execução e visualizar a responsabilidade de cada ator é entender o comportamento do sistema quando estiver pronto.

Dessa forma, detalharemos o que cada parte do *software* desenvolvido deve oferecer aos seus usuários.

O sistema será resumido em dois tipos de usuários, PROGRAD e LARE. O ator PROGRAD, terá dois casos de uso, ele terá acesso a fazer login no sistema e realizar uma consulta a um aluno previamente cadastrado. O ator LARE, terá quatro casos de uso, acesso a login no sistema; pode realizar o cadastro de um novo aluno no sistema; inserir em massa, ou seja, cadastrar mais de um aluno de uma vez no sistema; e por último, também poderá fazer consulta aos alunos cadastrados. A Figura 6 apresenta o diagrama de casos de uso.

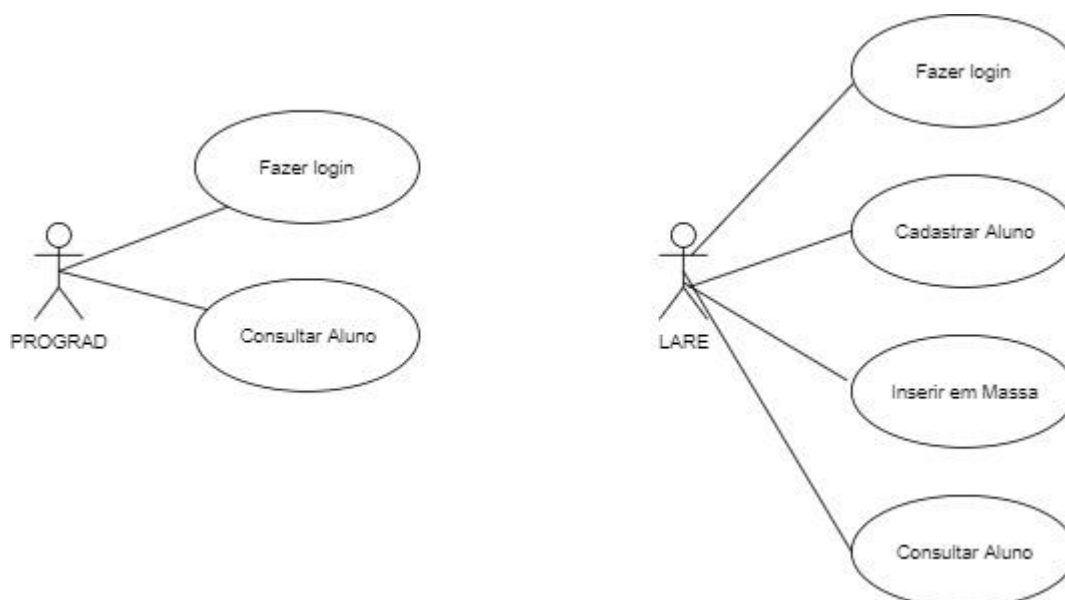


Figura 6: Diagrama de Casos de Uso

3.4.1 Detalhamento dos Casos de Uso

Tabela 4: Tabela Caso de Uso 1 – Fazer Login

Nome	Cenário	Ator(es)
Fazer Login	<p>Pré-condição:</p> <p>Usuário deve ser funcionário LARE ou PROGRAD;</p> <p>Fluxo normal:</p> <ol style="list-style-type: none">1. O caso de uso começa quando o usuário faz o seu primeiro <i>login</i> e realiza o cadastro designando seu perfil;2. O sistema no ambiente PROGRAD apresenta a função “Consultar Aluno” e “Logout”;3. O sistema no ambiente LARE apresenta as opções “Consultar Aluno”, “Inserir em Massa”, “Cadastrar Aluno”, “Excluir Aluno” e “Logout”;4. O caso de uso termina. <p>Fluxo de exceção:</p> <ol style="list-style-type: none">1. Se o usuário fechar o site o caso de uso termina;2. Se não for encontrado o usuário e senha compatíveis na base de dados, uma mensagem de <i>login</i> incorreto é informada. <p>Pós-condição:</p> <ol style="list-style-type: none">1. Perfil do usuário fica armazenado no banco de dados	Funcionário LARE; Funcionário PROGRAD

Tabela 5: Tabela Caso de Uso 2 – Cadastrar Aluno

Nome	Cenário	Ator(es)
------	---------	----------

<p>Cadastrar Aluno</p>	<p>Pré-condição:</p> <ol style="list-style-type: none"> 1. Usuário deve estar logado no sistema; 2. Usuário deve ter perfil do LARE; 3. O modulo APACHE do XAMPP deve estar iniciado; 4. O modulo MySQL do XAMPP deve estar iniciado. <p>Fluxo normal:</p> <ol style="list-style-type: none"> 1. O caso de uso começa quando o usuário seleciona a opção “Cadastrar Aluno”; 2. O usuário deve digitar a matrícula do aluno no campo indicado; 3. O usuário deve digitar o nome do aluno no campo indicado; 4. O Usuário deve selecionar a opção “Anexar Documento”, para selecionar o documento através da caixa de seleção do navegador; 5. O Usuário deve pressionar o botão “Cadastrar” para inserir esse novo aluno no banco de dados; 6. O caso de uso termina. <p>Fluxo de exceção:</p> <ol style="list-style-type: none"> 1. Se o usuário fechar o site o caso de uso termina; 2. Se o usuário digitar a matrícula fora dos padrões, será informada um erro como número de matrícula inválida; 3. Se o usuário anexar um arquivo de imagem diferente do tipo “.tif” ou “.tiff”, uma mensagem de erro será retornada; 4. Caso exista um número de matrícula já cadastrado, igual ao inserido no campo de matrícula, será retornado um erro. <p>Pós-condição:</p> <ol style="list-style-type: none"> 1. Banco de dados é incrementado com a(s) nova(s) inserção(s). 	<p>Funcionário LARE</p>
------------------------	---	-------------------------

Tabela 6: Tabela Caso de Uso 3 – Inserir em Massa

Nome	Cenário	Ator(es)
Inserir em Massa	<p>Pré-condição:</p> <ol style="list-style-type: none"> 1. O Usuário deve estar logado no sistema; 2. O Usuário deve ter perfil do LARE; 3. A pasta <i>Scan</i>, raiz do diretório contendo o documento de todos alunos no atual formato (formato usado antes da implementação do sistema), deve se encontrar dentro da pasta “tcc”, diretório raiz onde se encontra todo o sistema no servidor APACHE; 4. O módulo APACHE do XAMPP deve estar iniciado; 5. O modulo MySQL do XAMPP deve estar iniciado. <p>Fluxo normal:</p> <ol style="list-style-type: none"> 1. O caso de uso começa quando o usuário seleciona a opção “Inserir em Massa”; 2. O sistema faz uma varredura na pasta <i>Scan</i> e todas suas subpastas e insere todos os alunos ainda não cadastrados, na base de dados; 3. O caso de uso termina. <p>Fluxo de exceção:</p> <ol style="list-style-type: none"> 1. Se o usuário fechar o site o caso de uso termina; 2. Para todo diretório fora dos padrões até então atual, será retornado um erro de aluno não inserido; 3. Caso número de matrícula já for existente na base, será retornado um erro. <p>Pós-condição:</p> <ol style="list-style-type: none"> 1.O banco de dados é incrementado com as novas inserções. 	Funcionário LARE

Tabela 7: Tabela Caso de Uso 4 – Consultar Aluno

Nome	Cenário	Ator(es)
Consultar Aluno	<p>Pré-condição:</p> <ol style="list-style-type: none"> 1. Usuário deve estar logado no sistema; 2. Usuário deve ter perfil do LARE ou PROGRAD; 3. O modulo APACHE do XAMPP deve estar iniciado; 4. O modulo MySQL do XAMPP deve estar iniciado. <p>Fluxo normal:</p> <ol style="list-style-type: none"> 1. O caso de uso começa quando o usuário seleciona a opção “Consultar Aluno” ou faz o <i>login</i> no sistema; 2. O usuário deve digitar a matrícula ou o nome desejado no campo indicado e pressionar “Procurar”; 3. Encontrado o aluno, o sistema retorna: matrícula, nome, curso, polo, ingresso e documento referente ao aluno encontrado; 4. O campo “Documento” traz a opção “abrir” que ao ser selecionada fará <i>download</i> da documentação que foi digitalizada e anexada referente ao aluno; 5. O caso de uso termina. <p>Fluxo de exceção:</p> <ol style="list-style-type: none"> 1. Se o usuário fechar o site o caso de uso termina; 2. Se o número de matrícula não for encontrado na base de dados, uma mensagem contendo “aluno não encontrado” será retornada. <p>Pós-condição:</p> <ol style="list-style-type: none"> 1. Documento do Aluno fica salvo na pasta <i>downloads</i>. 	Funcionário PROGRAD; Funcionário LARE.

4 ESTRUTURAS DE PROGRAMAÇÃO UTILIZADAS

Neste capítulo serão abordadas as metodologias escolhidas para implementação do sistema, entendendo a importância de cada uma na execução do projeto. E teremos a seguinte ordenação: a Seção 4.1 refere-se ao HTML5, linguagem primordial para um *site web*, a Seção 4.2, o CSS é responsável pela apresentação visual da página, já na Seção 4.3 falamos do JavaScript, linguagem de programação interpretada estruturada. Na subseção 4.4 apresentamos o PHP, linguagem para aplicações presentes no lado do servidor, na subseção 4.5 conceituamos o SGBD MySQL e na subseção 4.6 fica explícito a necessidade do servidor HTTP Apache.

4.1 HTML 5

De acordo com seu desenvolvedor, W3C, *HyperText Markup Language* (HTML) é usada para descrever documentos de texto estáticos, e um navegador *web* é essencialmente um interpretador de HTML. A tecnologia surgiu em meados dos anos 90 como um documento curto que detalhava uma gama de elementos utilizados para construção de páginas *web*; e veio da junção entre os padrões *HyTime* e SGML. O *HyTime* é um padrão para representação estruturada de hipermídia e conteúdo baseado em tempo (W3, 2010).

Conforme a evolução da linguagem suas versões foram sendo incrementadas, e adquirindo outros elementos e ajustes nas regras. Atualmente na versão HTML 5, herda grande parte das características de seus predecessores, tornando, portanto, uma linguagem compatível tanto com navegadores antigos quanto os mais recentes. Essa compatibilidade é um princípio-chave do HTML 5. Algumas das muitas funcionalidades que foram adicionadas, são para dar semântica aos elementos; outras já são mais complexas e ajudam a construir aplicações *web* mais poderosas (W3, 2010).

4.2 CSS

CSS foi desenvolvido pelo W3C (*World Wide Web Consortium*) em 1996, por uma razão bem simples, o HTML não foi projetado para ter *tags* que ajudariam a formatar a página. Com as atualizações da linguagem HTML, houve um acréscimo de *tags* (elementos da linguagem HTML) que passaram a descrever também detalhes de fontes e cores. Então, a linguagem de folhas de estilos, *Cascading Style Sheets* (CSS), foi desenvolvida para acondicionar a formatação de estilo de página de um arquivo HTML, definindo, portanto, a apresentação desses documentos (Gonçalves, 2019).

O CSS separa o conteúdo da representação visual do site. Utilizando o CSS é possível alterar a cor do texto e do fundo, fonte e espaçamento entre parágrafos. Também é possível a criação de tabelas, usar variações de *layouts*, ajustar imagens para suas respectivas telas e assim por diante. E o que antes era escrito num único arquivo (conteúdo, códigos para descrever conteúdo e códigos para descrever aparência), passou a ser desmembrado (Vieira, 2010).

Sua atual versão, CSS3, introduziu uma grande variedade de efeitos visuais, tais como sombras, cantos arredondados e gradientes, deixando-o bem mais poderoso que as versões anteriores. A melhor novidade é em relação a flexibilidade na criação de layouts, trazendo mais autonomia para os *webdesigners* e também desenvolvedores, que de certa forma estão ligados ao visual do site (Gonçalves, 2019).

4.3 JAVASCRIPT

JavaScript é uma linguagem de programação que permite a você implementar itens complexos em páginas *web*, inventada por Brendan Eich em 1995, é uma linguagem de *script* dinâmica que oferece suporte à construção de objetos baseados em protótipo (W3, 2017). Sua sintaxe é intencionalmente similar a ambos Java e C++ para reduzir o número de conceitos novos necessários para aprender a linguagem. O JavaScript atua como um complemento às linguagens HTML, CSS e PHP no momento em que o desenvolvedor vai construir uma página na internet.

4.4 PHP

O PHP (um acrônimo recursivo para PHP: *Hypertext Preprocessor*) é uma linguagem de *script open source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento *web* e que pode ser embutida dentro do HTML (*PHP Documentation Group, 2019*).

Essa linguagem é mundialmente conhecida e uma das mais utilizadas pela facilidade em aprendê-la, manuseá-la, além de ser compatível com quase todos os sistemas operacionais que existem. Em vez de muitos comandos para mostrar HTML (como acontece com C ou Perl), as páginas PHP contêm HTML em código mesclado. O código PHP é delimitado pelas instruções de processamento (*tags*) de início e fim `<?php` e `?>` que permitem que você entre e saia do "modo PHP".

O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador (C. Felipe, 2019). O navegador recebe os resultados da execução desse *script*, mas não sabe qual era o código fonte. O servidor *web* pode ser configurado para processar todos os arquivos HTML com o PHP, e então não há como o usuário saber o código-fonte.

Em 1997, dois programadores reformularam os códigos do PHP e lançaram o PHP 3, mudando o nome de *Personal Home Page*, para *PHP Hypertext Pre-*

processor, essa versão contou com a primeira forma de recursos de orientações a objetos (C. Felipe, 2019). Com essa atualização, os programadores podiam implementar métodos e códigos. Atualmente, o PHP encontra-se na versão 7, a mesma que será usada neste projeto.

4.5 MYSQL

De acordo com Juliano Niederauer e Rubens Prates (2006), MySQL é um SGBD relacional que utiliza a linguagem padrão SQL (*Structured Query Language*) e é largamente utilizado em aplicações para a Internet. É o mais popular entre os bancos de dados com código-fonte aberto. Há mais de cinco milhões de instalações do MySQL no mundo todo, inclusive em *sítes* com alto volume de dados e de tráfego, como *Associated Press*, Google, NASA e Suzuki.

O MySQL é uma alternativa atrativa porque, mesmo possuindo uma tecnologia complexa de banco de dados, seu custo é bastante baixo. Tem como destaque suas características de velocidade, escalabilidade e confiabilidade, o que vem fazendo com que ele seja adotado por departamentos de TI (Tecnologia da Informação), desenvolvedores *Web* e vendedores de pacotes de *softwares* (Niederauer, 2006).

4.6 SERVIDOR HTTP APACHE

Um servidor *Web* é um computador que processa solicitações HTTP (*Hyper-Text Transfer Protocol*), o protocolo padrão da *Web*. Ao usar um navegador de internet para acessar um *site*, este faz as solicitações devidas ao servidor *Web* do *site* através de HTTP e então recebe o conteúdo correspondente. No caso do Apache, ele não só executa o HTTP, como outros protocolos, tais como o HTTPS (O HTTP combinado com a camada de segurança SSL - *Secure Socket Layer*), o FTP (*File Transfer Protocol*), entre outros (Alecrim, 2006).

Como servidor *Web*, o Apache é o mais conhecido e usado. Os motivos incluem sua excelente performance, segurança, compatibilidade com diversas plataformas e todos os seus recursos. É um *software* livre, o que significa que qualquer um pode estudar ou alterar seu código-fonte, além de poder utilizá-lo gratuitamente (Alecrim, 2006).

O servidor Apache é capaz de executar código em PHP, Perl, *Shell Script* e até em ASP e pode atuar como servidor FTP, HTTP, entre outros (Alecrim, 2006). Sua utilização mais conhecida é a que combina o Apache com a linguagem PHP e o banco de dados MySQL (combinação usada este projeto).

5 DESENVOLVIMENTO DO SOFTWARE E INTERFACE

A partir de agora detalharemos como o sistema foi desenvolvido, como as tabelas e objetos interagem entre si. O funcionamento das funções que darão a automação que buscamos para manutenção do banco. E o resultado final, a interface com o usuário. Ficando, portanto, este capítulo organizado na seguinte estrutura: na Seção 5.1 é apresentado o diagrama de entidade-relacionamento e as definições de seus objetos, na Seção 5.2 explicamos a criação do banco de dados, na Seção 5.3 comentamos sobre cada codificação do sistema, e por fim, na Seção 5.4 a apresentação dos projetos de telas

5.1 DIAGRAMA DE ENTIDADE-RELACIONAMENTO

Nos próximos parágrafos será descrito cada entidade do sistema e seus atributos, conforme visto no diagrama da Figura 7. O sistema é composto pelas seguintes entidades: curso, polo, login, aluno, consulta_aluno, curso_polo, assim como seus atributos e relacionamentos entre elas.

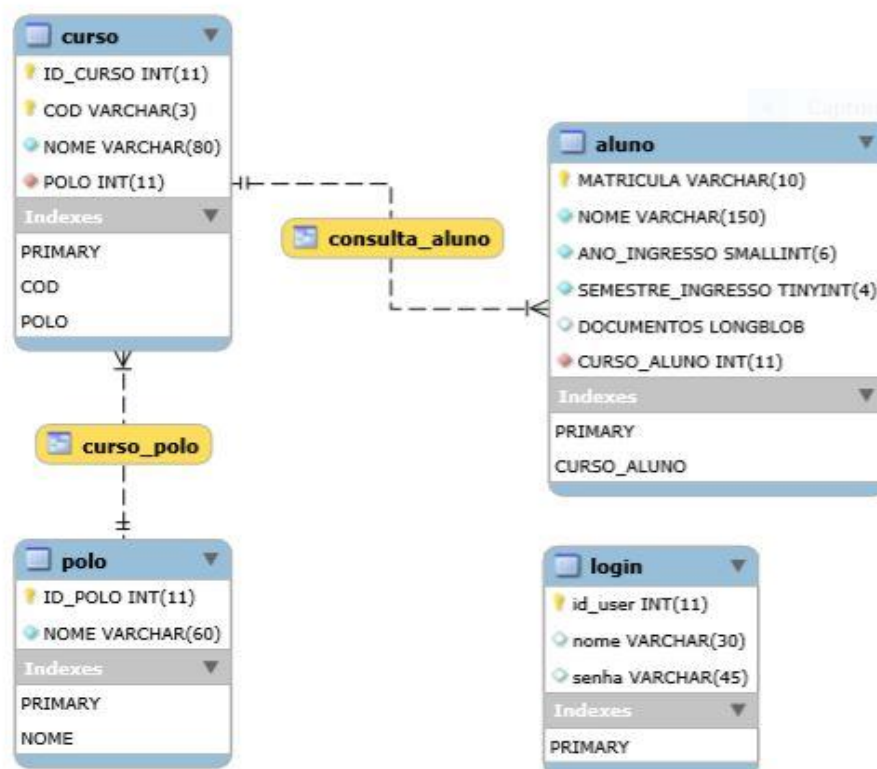


Figura 7: Diagrama de Entidade-Relacionamento

A Tabela *login* tem como função o armazenamento de cada usuário que acessará o sistema, contendo três colunas de autenticação, estão descritas estas na Tabela abaixo. E como parte do requisito apenas funcionários do LARE e PRO-GRAD devem ser cadastrados como usuários.

Tabela 8: Dicionário de Dados – Tabela login

Atributo	Tipo	Descrição
id_user	int	Chave primária, identifica o usuário como único dentro do banco.
nome	varchar	Utilizado para iniciar conexão com o sistema.
senha	varchar	Utilizada para autenticar o login no sistema.

A entidade *polo*, tabela do banco de dados, armazena todos os polos existentes da universidade. Essa tabela é populada previamente da utilização da aplicação pelo usuário, para perfeito funcionamento do sistema. Abaixo, é descrito seus atributos através da Tabela 9.

Tabela 9: Dicionário de Dados – Tabela polo

Atributo	Tipo	Descrição
id_polo	int	Chave primária, id único para cada polo.
nome	varchar	<i>String</i> do nome propriamente dito de cada polo

A entidade *curso* armazena todos os cursos existentes da universidade de diferentes polos, incluindo os cursos de ensino à distância. Todas estas características, são atributos descritos abaixo, na Tabela 10.

Tabela 10: Dicionário de Dados – Tabela curso

Atributo	Tipo	Descrição
id_curso	int	Chave primária, auto incremento, única para cada curso cadastrado no banco de dados
cod	varchar	Junto com id_curso completa a chave primária da tabela, identifica um código único de cada curso.
nome	varchar	Nome do curso referente ao código único do mesmo.
polo	int	Chave estrangeira, referencia id_polo design qual polo cada curso pertence.

A entidade *aluno* representa o aluno que cursou ou está cursando uma graduação na Universidade Federal Fluminense. Contendo seis colunas, todas representando atributos do aluno, como matrícula, nome, ano de ingresso e semestre, curso e seus documentos, são descritas abaixo na Tabela 11.

Tabela 11: Dicionário de Dados – Tabela aluno

Atributo	Tipo	Descrição
matricula	varchar	Chave primária, única para cada aluno.
nome	varchar	Nome do aluno.
ano_ingresso	smallint	Ano de ingresso no curso de graduação.
semestre_ingresso	tinyint	Semestre de ingresso no curso de graduação.
documentos	longblob	Documento digitalizado com extensão .tif ou .tiff.
curso_aluno	int	Chave estrangeira referencia id_curso, utilizado para obter informação do curso que determinada matricula está inscrita.

Os objetos **curso_polo** e **consulta_aluno** são apenas *views* do banco de dados, que fazem uma integração entre as tabelas da base para uma melhor busca e inserção dos dados no funcionamento da aplicação, sem necessidade de poluição de linhas no código da aplicação com *queries* demasiadamente grandes.

5.2 CRIANDO O BANCO DE DADOS

É de extrema importância e vital para o funcionamento do sistema a pré-criação de objetos e a população deles na base MySQL que foi chamada de “tcc”. São, portanto, definidas as seguintes tabelas: “LOGIN”, “POLO”, “CURSO” e “ALUNO”; as *views* “CURSO_POLO” e “CONSULTA_ALUNO”. O código SQL para criação da base de dados, as tabelas onde serão armazenados os dados dos alunos e a criação de um usuário chamado “admin” se encontram nas Figuras 15 e 16 do Apêndice A.

Após a criação das tabelas, foi preciso inserir os dados já existentes que são os polos da universidade e os cursos oferecidos por cada um, juntos eles irão no

código designar o local do curso quando um aluno novo for adicionado. Matrículas de 2011 para trás possuem 8 dígitos, sendo os algarismos 4 e 5 da matrícula designado para identificação do curso. E as matrículas de 2012 em diante possuem 9 dígitos, sendo os algarismos 4, 5 e 6 para identificação do curso. Mais adiante entenderemos como o sistema fará a leitura dessas matrículas para alocar adequadamente cada aluno nos seus respectivos cursos. Nas Figuras 17,18,19 e 20 do Apêndice A se encontram os códigos dessa inserção.

As *views* criadas fazem uma consulta relacionando as tabelas curso, polo e aluno, para uma melhor visualização dos dados e também para uma melhor performance através das consultas pela aplicação. Código localizado na Figura 21 do Apêndice A.

5.3 DESCRIÇÃO DA IMPLEMENTAÇÃO DAS TELAS DO SISTEMA

Dando continuidade no desenvolvimento do software, a seguir explicaremos as telas de navegação e suas funcionalidades. Dentre elas, temos a tela de login; tela inicial, que será a mesma para consulta do aluno; tela de cadastro de um novo aluno; e a tela para inserção em massa de alunos. Todas descritas em subseções abaixo.

5.3.1 Tela de *Login*

A tela de *login* (figura 8) é constituída pela página “*login.php*”, nela temos o *layout* da página e a instanciação da classe “*user*” encontrada dentro do arquivo “*user.php*”. Esses dois arquivos em conjunto têm o objetivo de instanciar um novo usuário, validar os dados de entrada de *login*, fazer a conexão na base de dados e a validação se o usuário existe no banco, sendo os dois últimos através das funções *connect* e *login* da classe “*user*”. Somente usuários cadastrados previamente na base de dados terão acesso ao sistema. As figuras 22 e 23 no Apêndice B contemplam a codificação descrita acima.

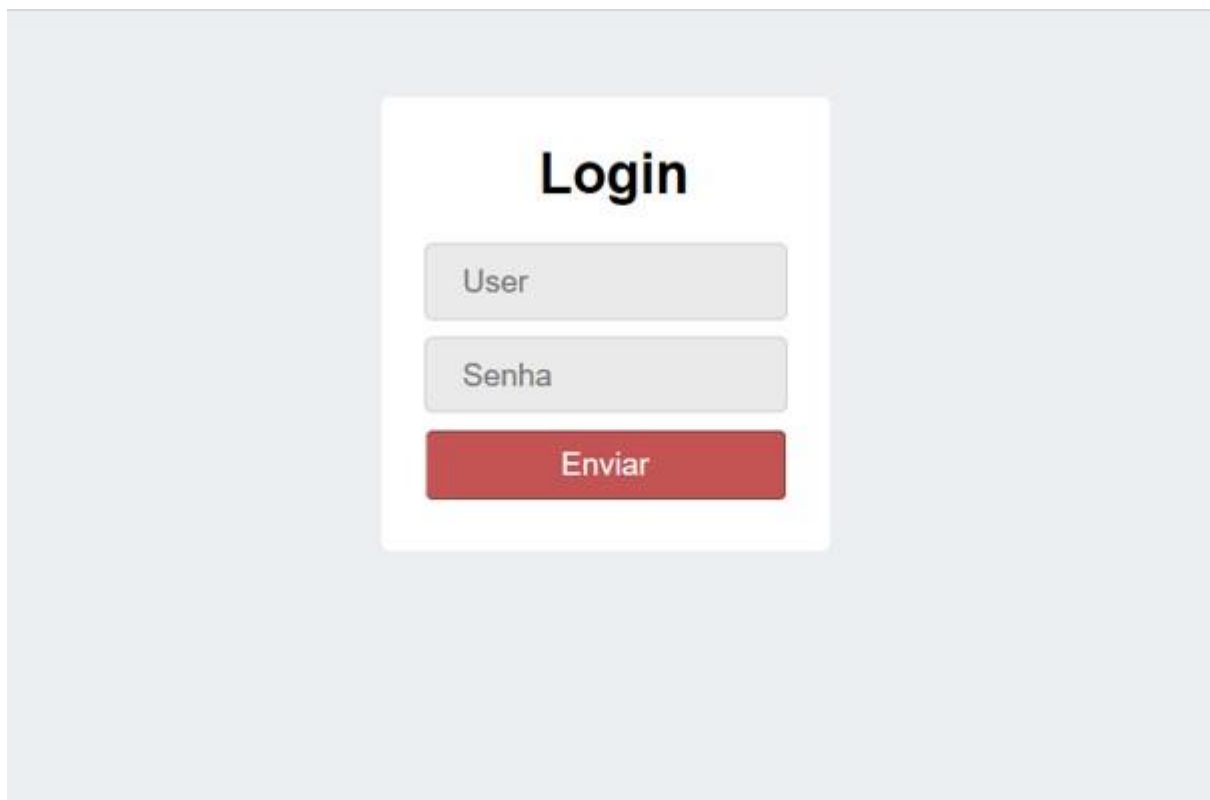


Figura 8: Tela de *login*

5.3.2 Tela Inicial

A tela inicial após *login* no sistema é justamente o campo para fazer a busca de um aluno no banco de dados, motivo principal para desenvolvimento do sistema. Após verificado o *login* e autenticado, o usuário é redirecionado para a página de novo “*index.php*” (Figura 9).

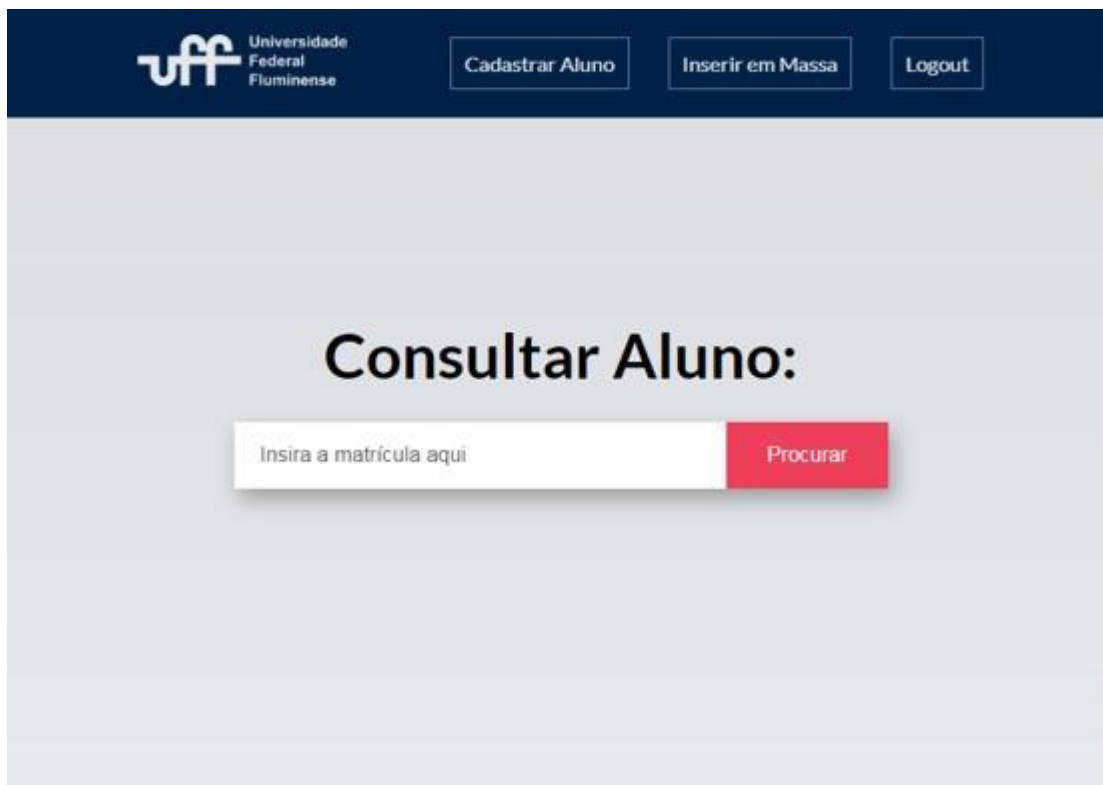


Figura 9: Tela *index* do sistema

O usuário poderá fazer a consulta ao aluno através de seu número de matrícula (Figura 10), número único para cada aluno. Toda vez que a página “*index.php*” é carregada, será feita uma autenticação se o usuário está logado, caso contrário, será redirecionado para a tela de *login*.

A página “*index.php*” exibe um formulário para entrada da matrícula onde será validado pelas funções JavaScript “*isInputNumber(evt)*” e “*isMatricula()*”. A função “*isInputNumber(evt)*” verifica em tempo real e só aceita que todo valor digitado no *input* da matrícula seja um número. Já a função “*isMatricula()*” verifica após o acionamento do botão enviar, via *regex*, se a matrícula corresponde aos padrões de matrícula UFF para alunos acima de 2011, onde a matrícula é composta por nove números. A Figura 24 no Apêndice B demonstra as funções de validação JavaScript.

Após a validação via JavaScript, o número da matrícula é passado como variável para a função “*consultarAluno(\$matricula)*”, que se encontra no arquivo “*listar.php*” com a dita função e o *layout* de exibição da tabela com as informações do aluno. A função *consultarAluno(\$matricula)* verifica se a matrícula passada existe na

base de dados e se realmente há um aluno com esse número cadastrado. Em caso de sucesso, será exibido na própria tela de consulta as informações do aluno contendo nome, matrícula, polo, curso, ano e semestre de ingresso e sua documentação. As Figuras 25, 26, 27 e 28 no Apêndice B ilustram toda a codificação descrita.

Na página “listar.php”, ao exibir a coluna “documentos” do aluno, será feita uma chamada à página “abrir_arquivo.php”. Essa página será carregada exibindo o documento anexado do aluno e disponibilizando para *download* através do *link* “abrir” no campo da coluna documentos. A Figura 29 do Apêndice B ilustra o processo.

Matrícula	Nome	Curso	Pólo	Ingresso	Documento
21104202	Fernanda Gomes Motta	Engenharia De Produção	Niterói	2011/2	abrir

Figura 10: Consultando aluno cadastrado

5.3.3 Tela para cadastro de um novo aluno

Essa tela permite o cadastro de um novo aluno no sistema de forma individual. A página “cadastrar.php” contém o formulário para cadastro dos alunos, a função “cadastrarAluno(\$matricula, \$nome, \$doc)”, além de validações de *login* e dados de entrada.

Ao entrar na página será feita a mesma validação se o usuário está logado, caso não esteja, será redirecionado à página inicial. Só então será exibido um formulário para cadastro do aluno com os campos de nome, matrícula e documento a ser anexado (Figura 11). O formulário conta com as validações via JavaScript “isInputNumber()” e “isMatricula()”, já demonstradas na Figura 24. E também a função “isInputName()” que só permite caracteres do alfabeto no campo destinado ao nome do aluno. A Figura 30 no Apêndice B apresenta o código dessa função.



Figura 11: Tela Cadastrar Aluno

Para obter êxito na inserção dos dados não pode haver campos em branco e o formato da imagem deve ser “.tif” ou “.tiff”. Após toda validação, a função “cadastrarAluno(\$matricula, \$nome, \$doc)”, é chamada e sendo passado os parâmetros enviados pelo usuário através do formulário, fará o tratamento de todos os dados a serem inseridos corretamente na base de dados (Figuras 12 e 13).

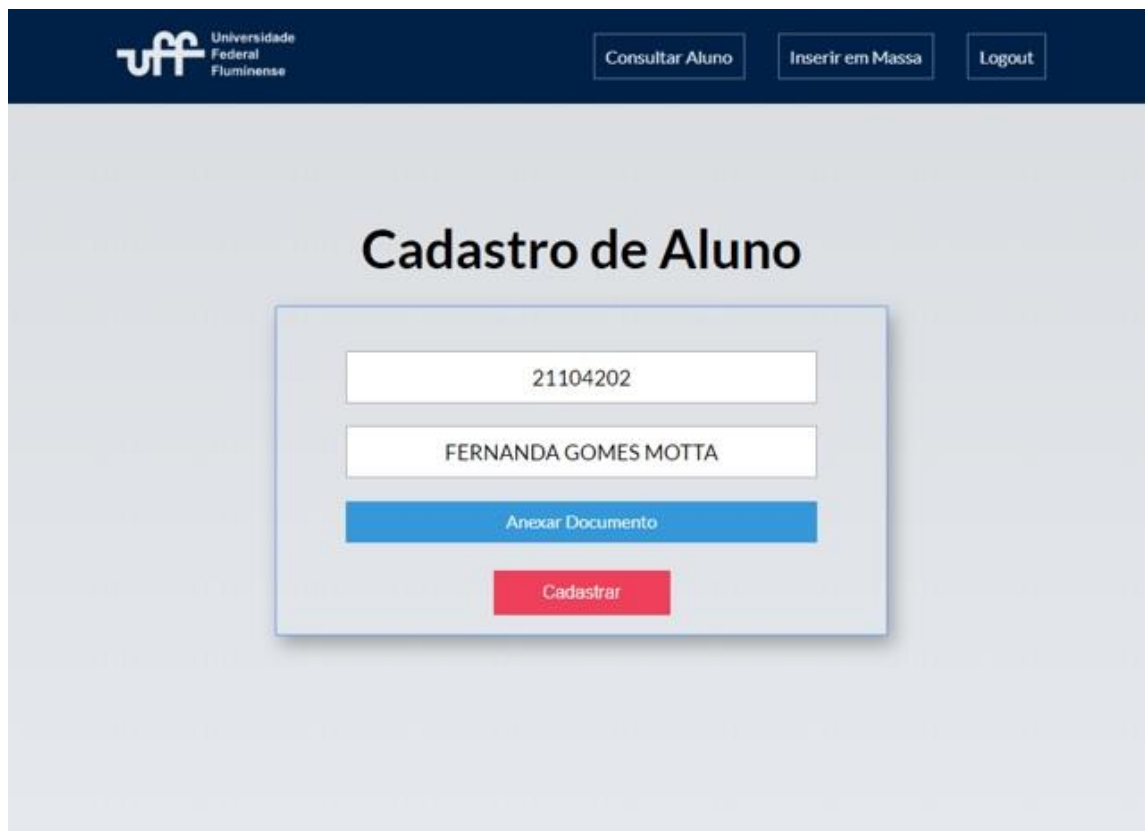


Figura 12: Cadastrando novo aluno

Fazendo a leitura da matrícula conseguimos as informações de semestre e ano de ingresso, curso e polo. Portanto, a inserção dos dados nas suas respectivas colunas do banco é feita através de uma função que desmembra a matrícula em partes, onde cada parte determina essa correspondência. A matrícula é lida da seguinte maneira:

- 1º dígito = semestre de ingresso (1º ou 2º)
- 2º e 3º dígitos = ano de ingresso.
- 4º, 5º e 6º dígitos = curso (obs. matrículas de 2011 pra trás são considerados apenas os dígitos 4º e 5º)

- 7º, 8º e 9º dígitos = matrícula do aluno (obs. matrículas de 2011 pra trás não possui o 9º dígito)

As Figuras 31, 32 e 33 no Apêndice B ilustram a codificação do descrito acima.

UFF Universidade Federal Fluminense

Consultar Aluno Inserir em Massa Logout

Cadastro de Aluno

Matrícula do Aluno

Nome do Aluno

Anexar Documento

Cadastrar

Inserido com sucesso!

Figura 13: Aluno cadastrado com sucesso

5.3.4 Tela para inserção de dados em massa

A página "search.php" foi projetada para solucionar o grande problema encontrado quando decidimos desenvolver esse sistema. Essa funcionalidade é o que de fato torna o processo de arquivamento automatizado, pois seria um trabalho demorado e cansativo fazer a inserção de todo acervo digital já existente através da página "cadastrar.php". Assim, com esse desenvolvimento permite-se fazer um *scan* em todo o diretório Graduação e inserir de forma mais prática todos os alunos e documentos previamente digitalizados no formato de banco de dados relacional do sistema. O usuário só terá acesso a esta página se estiver logado.

Todo o conteúdo dos diretórios deve estar no formato padrão já explicitado na Figura 3. Para a realização do *script* da página “search.php” é necessário que todo o diretório dos alunos já existente seja movido ou copiado para dentro do caminho de *localhost* do servidor, a partir disso, deverá ser configurada uma variável dentro do *script* com o caminho absoluto desse diretório movido para o *localhost* até a pasta “GRADUACAO”.

Assim que a página “search.php” for carregada o *script* entra em ação, sendo feito então um *scan* pasta por pasta de cada diretório até localizar a pasta com o nome e matrícula do aluno, ambos serão armazenados em variáveis. Já o *scan* realizado no documento digitalizado do aluno terá o seu caminho armazenado em uma variável. Após todo o tratamento referente as regras da matrícula, o aluno será inserido automaticamente na base de dados, ocorrendo um *loop* até que todo o diretório/aluno seja lido. As Figuras 34 e 35, no Apêndice B, mostram o código da página “search.php”.

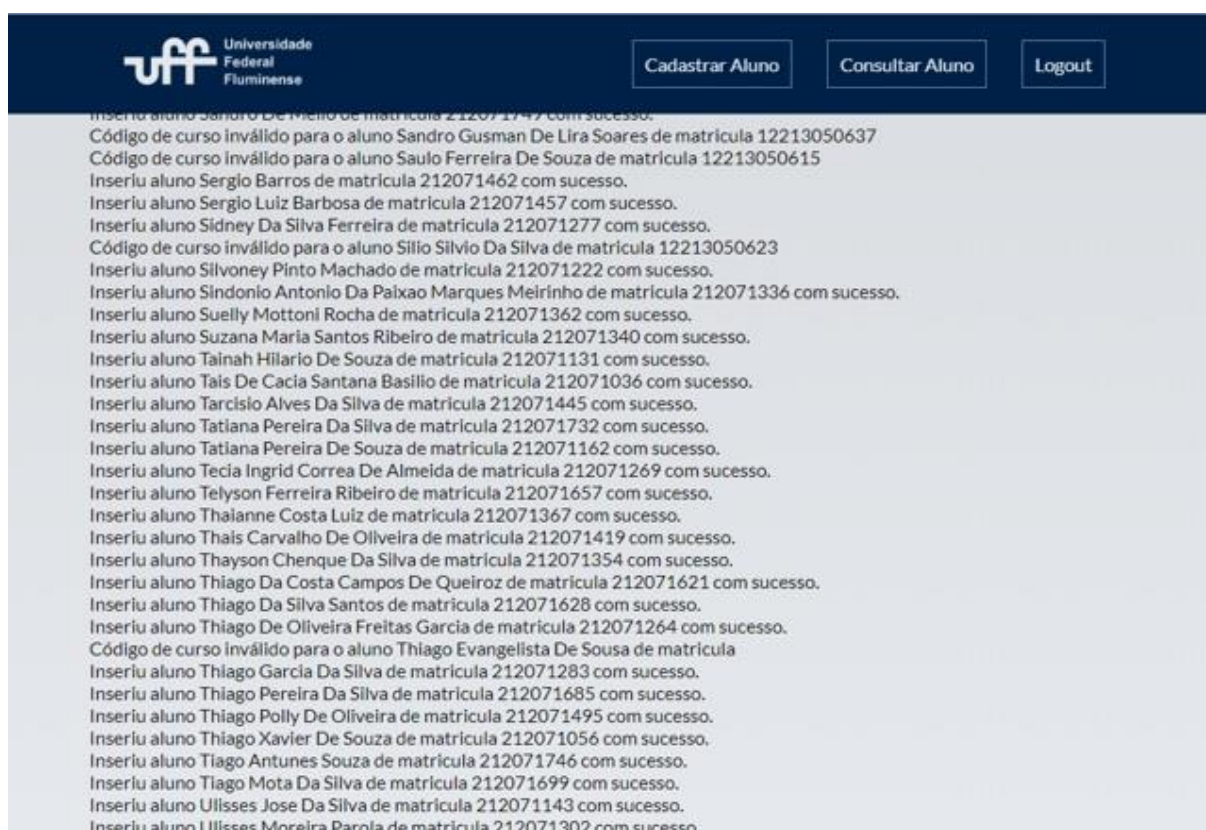


Figura 14: Inserindo alunos em massa

5.3.5 Arquivo "conn.php"

O arquivo "conn.php" é apenas um arquivo com as configurações referentes a conexão com o *database*, com ele toda a conexão do sistema é realizada e em caso de alteração, só é necessário a mudança no mesmo. Nele estão presentes as variáveis de ambiente referente ao *localhost*, nome do *database*, usuário e senha para conexão. Além de tratamentos de erros, como exceções em caso de conexão inválida. Código explícito na figura 36 do Apêndice B.

6 CONCLUSÃO

Este trabalho teve como objetivo desenvolver um sistema *web* simples, porém eficaz, no intuito de trazer mais praticidade, agilidade e modernidade no trabalho de digitalização dos documentos estudantis administrados pela PROGRAD junto ao LARE.

Propomos uma solução para o problema criando um banco de dados MySQL, estruturado seguindo a padronização dos diretórios já existentes de modo a receber os arquivos relativos aos estudantes de graduação da UFF. Assim, fornecer um mecanismo de manuseio dos dados mais eficiente que o praticado.

Durante os testes realizados no ambiente do LARE foi possível observar que o sistema atende aos requisitos, funciona como o esperado, e de fato trouxe melhorias nos processos de arquivamento e consulta, tornando-os mais ágeis. A aceitação foi unânime e todos se beneficiaram com a implantação do projeto.

6.1 TRABALHOS FUTUROS

Devido às limitações encontradas na infraestrutura do LARE desde a elaboração do projeto, o sistema foi projetado para rodar dentro do servidor HTTP Apache. Portanto, um ponto a ser trabalho futuramente é a hospedagem dele num servidor de interesse da UFF, tornando-o mais abrangente.

Como o foco principal do trabalho foi a automação do arquivamento e consulta dos dados, não foi implementado itens complementares como, por exemplo, relatórios oriundos do sistema, onde poderiam exibir toda movimentação sistema, entrada e exclusão de dados.

E, assim como os dossiês dos alunos são digitalizados, o mesmo será feito com a documentação dos servidores públicos, funcionários da UFF, então poderia ser estudado um novo módulo do sistema para gestão desses arquivos também.

REFERÊNCIAS

- [1] Vieira, K. (2010). Hostigator. Obtido em 05 de 09 de 2019, de [hostigator.com: https://www.hostigator.com.br/blog/o-que-e-css/](https://www.hostigator.com.br/blog/o-que-e-css/)
- [2] Gonçalves, A. (2019). Hostinger. Obtido em 05 de 09 de 2019, de <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>
- [3] What is PHP? (2001-2019). PHP Documentation. Obtido em 06 de 09 de 2019, de <https://www.php.net/manual/en/intro-what-is.php>
- [4] Felipe, C. (2019). Hostigator. Obtido em 06 de 09 de 2019, de <https://www.hostinger.com.br/tutoriais/o-que-e-php-guia-basico/>
- [5] Niederauer, J & Prates, R (2006). MySQL 5 - Guia de Consulta Rápida. São Paulo: Novatec.
- [6] Alecrim, E. (2006). Infowester. Obtido em 06 de 09 de 2019, de <https://www.infowester.com/servapach.php>
- [7] W3School HTML/CSS Tutorials, References and Examples (2019), Obtido em 01 de 09 de 2019, de <http://www.w3schools.com/>. (W3School is not related to W3C).
- [8] Santos, J. N. & Silva, J. A. S. (2015). SGBD MySQL. FACCAT. Taquara.
- [9] Duckett, J. (2011). HTML & CSS Design and build Websites. John Wiley & Sons, Inc.
- [10] TutorialRepublic. PHP MySQL Login System, Obtido em 15 de 09 de 2019, de <https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php>
- [11] W3. HTML5. (2010). Documentação do HTML5. Obtido em 01 de 09 de 2019, de <https://dev.w3.org/html5/html-author>
- [12] W3. Javascript. (2017). Documentação do Javascript. Obtido em 01 de 09 de 2019 de <https://www.w3.org/standards/webdesign/script>

APÊNDICE A – CRIAÇÃO DO BANCO DE DADOS

```
CREATE DATABASE IF NOT EXISTS `tcc` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;  
USE `tcc`;  
  
DROP TABLE IF EXISTS `login`;  
  
CREATE TABLE `login` (  
  `id_user` int(11) NOT NULL AUTO_INCREMENT,  
  `nome` varchar(30) NOT NULL,  
  `senha` varchar(45) NOT NULL,  
  `tipo` varchar(2) NOT NULL,  
  PRIMARY KEY (`id_user`)  
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT COLLATE=utf8_general_ci;  
  
INSERT INTO `login` (`id_user`, `nome`, `senha`, `tipo`) VALUES  
(1, 'admin', '202cb962ac59075b964b07152d234b70', 1);
```

Figura 15: Código para criação do banco de dados e tabela login

```
CREATE TABLE POLO (  
  ID_POLO INT PRIMARY KEY AUTO_INCREMENT,  
  NOME VARCHAR(60) UNIQUE NOT NULL  
) DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;  
  
CREATE TABLE CURSO (  
  ID_CURSO INT AUTO_INCREMENT,  
  COD VARCHAR(3) UNIQUE,  
  NOME VARCHAR(80) NOT NULL,  
  POLO INT NOT NULL,  
  PRIMARY KEY (ID_CURSO, COD),  
  FOREIGN KEY (POLO) REFERENCES POLO(ID_POLO)  
) DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;  
  
CREATE TABLE ALUNO (  
  MATRICULA VARCHAR(10) PRIMARY KEY UNIQUE NOT NULL,  
  NOME VARCHAR(150) NOT NULL,  
  ANO_INGRESSO smallint NOT NULL,  
  SEMESTRE_INGRESSO TINYINT NOT NULL,  
  DOCUMENTOS LONGBLOB,  
  CURSO_ALUNO INT NOT NULL,  
  FOREIGN KEY (CURSO_ALUNO) REFERENCES CURSO(ID_CURSO)  
) DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Figura 16: Criação das tabelas Polo, Curso e Aluno

```

INSERT INTO POLO (NOME) VALUES ('Angra dos Reis');
INSERT INTO POLO (NOME) VALUES ('Campos dos Goytacazes');
INSERT INTO POLO (NOME) VALUES ('Curso à Distância (CEDERJ)');
INSERT INTO POLO (NOME) VALUES ('Macaé');
INSERT INTO POLO (NOME) VALUES ('Niterói');
INSERT INTO POLO (NOME) VALUES ('Nova Friburgo');
INSERT INTO POLO (NOME) VALUES ('Rio das Ostras');
INSERT INTO POLO (NOME) VALUES ('Santo Antônio de Pádua');
INSERT INTO POLO (NOME) VALUES ('Volta Redonda');

```

Figura 17: Cadastrando os polos existentes na base de dados

```

INSERT INTO CURSO (COD, NOME, POLO) VALUES ('001', 'BIBLIOTECONOMIA E DOCUMENTAÇÃO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('002', 'HISTÓRIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('003', 'GEOGRAFIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('004', 'CIÊNCIAS ECONÔMICAS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('005', 'CIÊNCIAS SOCIAIS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('006', 'SERVIÇO SOCIAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('007', 'DIREITO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('008', 'ENFERMAGEM E OBSTETRÍCIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('009', 'NUTRIÇÃO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('010', 'PEDAGOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('011', 'JORNALISMO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('012', 'PUBLICIDADE E PROPAGANDA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('013', 'CINEMA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('014', 'ARQUIVOLOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('015', 'FARMÁCIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('016', 'MEDICINA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('017', 'ODONTOLOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('018', 'MEDICINA VETERINÁRIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('019', 'ENGENHARIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('020', 'MATEMÁTICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('021', 'LETRAS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('022', 'CIÊNCIAS CONTÁBEIS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('023', 'ADMINISTRAÇÃO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('024', 'PSICOLOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('025', 'FÍSICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('026', 'ARQUITETURA E URBANISMO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('027', 'ENGENHARIA QUÍMICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('028', 'QUÍMICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('029', 'QUÍMICA INDUSTRIAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('030', 'COMUNICAÇÃO SOCIAL JORNALISMO / PUBLICIDADE E PROPAGANDA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('031', 'CIÊNCIA DA COMPUTAÇÃO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('032', 'PEDAGOGIA', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('033', 'PRODUÇÃO CULTURAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('034', 'ENFERMAGEM', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('035', 'MATEMÁTICA', 8);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('036', 'SERVIÇO SOCIAL', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('037', 'ENGENHARIA CIVIL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('038', 'ENGENHARIA ELÉTRICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('039', 'ENGENHARIA METALÚRGICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('040', 'ENGENHARIA MECÂNICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('041', 'ENGENHARIA DE TELECOMUNICAÇÕES', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('042', 'ENGENHARIA DE PRODUÇÃO', 5);

```

Figura 18: Populando a tabela Cursos – parte 1


```

INSERT INTO CURSO (COD, NOME, POLO) VALUES ('043', 'ENGENHARIA AGRÍCOLA E AMBIENTAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('044', 'CIÊNCIAS BIOLÓGICAS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('045', 'ENGENHARIA DE PRODUÇÃO', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('046', 'ENGENHARIA MECÂNICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('047', 'TURISMO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('048', 'BIOMEDICINA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('049', 'ESTUDOS DE MÍDIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('050', 'GEOFÍSICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('051', 'ENGENHARIA DE PETRÓLEO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('052', 'ENGENHARIA DE AGRONEGÓCIOS', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('053', 'ADMINISTRAÇÃO', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('054', 'ESTATÍSTICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('055', 'EDUCAÇÃO FÍSICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('056', 'ENGENHARIA DE RECURSOS HÍDRICOS E DO MEIO AMBIENTE', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('057', 'CINEMA E AUDIOVISUAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('058', 'FILOSOFIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('059', 'RELAÇÕES INTERNACIONAIS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('060', 'CIÊNCIA DA COMPUTAÇÃO', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('061', 'ODONTOLOGIA', 6);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('062', 'PRODUÇÃO CULTURAL', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('063', 'ENGENHARIA DE PRODUÇÃO', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('064', 'SERVIÇO SOCIAL', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('065', 'PEDAGOGIA', 8);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('066', 'CIÊNCIAS ECONÔMICAS', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('067', 'GEOGRAFIA', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('068', 'CIÊNCIAS SOCIAIS', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('069', 'PSICOLOGIA', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('070', 'MATEMÁTICA', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('071', 'TECNOLOGIA COMPUTAÇÃO', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('072', 'EMPREENDEDORISMO E INOVAÇÃO', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('073', 'ENFERMAGEM', 7);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('074', 'ADMINISTRAÇÃO PÚBLICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('075', 'CIÊNCIAS CONTÁBEIS', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('076', 'FÍSICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('077', 'MATEMÁTICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('078', 'QUÍMICA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('079', 'BIOMEDICINA', 6);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('080', 'FONOAUDIOLOGIA', 6);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('081', 'PSICOLOGIA', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('082', 'FÍSICA', 8);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('083', 'SISTEMAS DE INFORMAÇÃO', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('084', 'DIREITO', 4);

```

Figura 19: Populando a tabela Cursos – parte 2

```

INSERT INTO CURSO (COD, NOME, POLO) VALUES ('085', 'CIÊNCIAS ATUARIAIS', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('086', 'ADMINISTRAÇÃO PÚBLICA', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('087', 'LETRAS', 3);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('088', 'DIREITO', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('089', 'HISTÓRIA', 2);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('090', 'DIREITO', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('091', 'PSICOLOGIA', 9);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('092', 'COMPUTAÇÃO', 6);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('093', 'CIÊNCIAS NATURAIS', 8);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('094', 'HOTELARIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('095', 'CIÊNCIA AMBIENTAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('096', 'DESENHO INDUSTRIAL', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('097', 'DISCIPLINA ISOLADA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('098', 'SOCIOLOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('099', 'ANTROPOLOGIA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('100', 'ARTES', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('101', 'GEOGRAFIA', 1);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('102', 'SEGURANÇA PÚBLICA', 5);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('103', 'POLÍTICAS PÚBLICAS', 1);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('104', 'ADMINISTRAÇÃO', 4);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('105', 'CIÊNCIAS CONTÁBEIS', 4);
INSERT INTO CURSO (COD, NOME, POLO) VALUES ('107', 'PROCESSOS GERENCIAIS', 5);

```

Figura 20: Populando a tabela Cursos – parte 3

```

-- CRIAÇÃO DE VIEW CURSO_POLO

CREATE VIEW CURSO_POLO AS
SELECT C.COD, C.NOME AS NOME_CURSO, P.NOME AS NOME_POLO
FROM CURSO C, POLO P
WHERE P.ID_POLO = C.POLO
ORDER BY COD;

-- CRIAÇÃO DE VIEW CONSULTA_ALUNO

CREATE VIEW CONSULTA_ALUNO AS
SELECT A.MATRICULA, A.NOME, C.NOME AS CURSO,
P.NOME AS POLO, A.ANO_INGRESSO, A.SEMESTRE_INGRESSO, A.DOCUMENTOS
FROM ALUNO A, CURSO C, POLO P
WHERE A.CURSO_ALUNO=C.ID_CURSO
AND C.POLO=P.ID_POLO;

```

Figura 21: Criação das *views*

APÊNDICE B – CODIFICAÇÃO DO SOFTWARE

```
<?php
Class User{
    private $pdo;
    public $error = "";

    public function connect($db, $host, $user, $password){
        global $pdo;
        global $error;
        try {
            $pdo = new PDO("mysql:dbname=".$db.";host=".$host,$user,$password);
        } catch (PDOException $e) {
            $error = $e->getMessage();
        }
    }

    public function login($login, $senha){
        global $pdo;
        $sql = $pdo->prepare("SELECT id_user FROM
        |login WHERE nome = :e AND senha = :s");
        $sql->bindValue(":e",$login);
        $sql->bindValue(":s",md5($senha));
        $sql->execute();
        if ($sql->rowCount() > 0){
            $dado = $sql->fetch();
            session_start();
            $_SESSION['id_user']=$dado['id_user'];
            return true;
        } else{
            return false; //nao logou
        }
    }
}
?>
```

Figura 22: Classe “user.php”

```

<?php
    require_once 'class/user.php';
    $u = new User;
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/style.css">
    <title>TCC</title>
    <script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
<div class="container">
    <form action="" class="form-contact" method="post" tabindex="1">
    <h1>login</h1>
    <input type="text" class="form-contact-input" name="user" maxlength="30" placeholder="User" />
    <input type="password" class="form-contact-input" name="pwd" maxlength="45" placeholder="Senha" />
    <button type="submit" value="Entrar" name="enviar" class="form-contact-button">Enviar</button>
    </form>
</div>
<?php
    if (isset($_POST['enviar'])){
        $user = addslashes($_POST['user']);
        $senha = addslashes($_POST['pwd']);

        if (!empty($user) && !empty($senha)){
            $u->connect("tcc", "localhost", "root", "");
            if($u->error == ""){
                if($u->login($user,$senha)){
                    echo "logou";
                    header("location: index.php");
                } else {
                    echo "Email ou senha inválidos.";
                }
            } else {
                echo "Erro: ".$u->error;
            }
        } else {
            echo "Preencha todos os campos!";
        }
    }
?>
</body>
</html>

```

Figura 23: Página de login “login.php”

```

function isInputNumber(evt){
    var ch = String.fromCharCode(evt.which);
    if(!(/[0-9]/.test(ch))){
        evt.preventDefault();
    }
}

function isMatricula(){
    var matricula = document.getElementById('matricula');
    var regex = /^[0-9]{3}\.[0-9]{3}\.[0-9]{3}|[0-9]{8}|[0-9]{9}$/;
    if (!regex.test(matricula.value)){
        alert('Número de matrícula inválido.');
```

Figura 24: Funções “isInputNumber()” e “isMatricula()”

```

<?php
    session_start();
    if(!isset($_SESSION['id_user'])){
        header("location: login.php");
        exit;
    }
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/main_style.css">
    <title>TCC</title>
    <script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
<div class="container">
    <header>
        <h2><a href="http://www.uff.br/"></a></h2>
        <nav>
            <ul>
                <li>
                    <a href="cadastrar.php" class="btn" title="Cadastrar">Cadastrar Aluno</a>
                </li>
                <li>
                    <a href="search.php" class="btn" title="Inserir em Massa">Inserir em Massa</a>
                </li>
                <li>
                    <a class="btn" href="sair.php" title="Logout">Logout</a>
                </li>
            </ul>
        </nav>
    </header>

```

Figura 25: Tela inicial "index.php" - parte 1

```

<div class="cover">
    <h1>Consultar Aluno:</h1>
    <form class="flex-form" name="matriculaform" method="post" tabindex="1" onsubmit="return isMatricula();">
        <input type="search" id="matricula" name="matricula"
        placeholder="Insira a matricula aqui" required onkeypress="isInputNumber(event)" />
        <input type="submit" name="enviar" value="Procurar">
    </form>
    <?php
        if (isset($_POST['enviar'])){
            $matricula = $_POST['matricula'];
            include("listar.php");
            consultarAluno($matricula);
        }
    ?>
</div>
</body>
</html>

```

Figura 26: Tela inicial "index.php" - parte 2

```

<!DOCTYPE html>
<html lang="pt-br">

<head>
    <meta charset="UTF-8">
    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" href="">
    <style>
        table{border-spacing: 1; border-collapse: collapse; background: white; border-radius: 10px; overflow: hidden; width: 100%; margin: 0 auto;
            position: relative;}
        div {display: block; margin-top: 10px;}
        table100-head tr {height: 60px; background: #363636;}
        table100-head th {font-family: OpenSans-Regular;font-size: 19px;color: #fff;line-height: 1.2;font-weight: unset;}
        table td, table th {text-align: left;padding-left: 8px;}
        .column1 {width: 120px; text-align: left;padding-left: 20px;}
        .column2 {width: 100px;}
        .column3 {width: 220px;}
        .column4 {width: 190px;}
        .column5 {width: 80px;text-align: center;padding-right: 20px;}
        .column6 {width: 50px;text-align: right; padding-right: 20px;}
        s {color: blue;}
        table tbody tr td.column6{text-align: center;}
        tbody {display: table-row-group;vertical-align: middle;border-color: inherit;}
        table tbody tr {height: 50px;}
        tbody tr {font-family: OpenSans-Regular;font-size: 17px;color: #555555;line-height: 1.2;font-weight: unset; border: 0;background-color: #f5f5f5;}
        tbody tr: hover {color: #555555; background-color: #f0f0f0; cursor: pointer;}

        .nofind {background-color: #d73851; width: 250px; height: 30px; border-radius: 2px; display: block; align-items: center; }
        div.nofind p {font-size: 18px; font-family: 'Lato', sans-serif; font-weight: 500; color: white; display: inline-block; text-align: center;
            margin: 5px 20px;}
    </style>
</head>

```

Figura 27: Página "listar.php" - parte 1

```

<body>
    <div class="container">
        <?php
            function consultarAluno($matricula){
                require_once 'conn.php';
                $consultaSQL = "SELECT * FROM CONSULTA_ALUNO where matricula='$matricula'";
                $resultado = mysqli_query($conn, $consultaSQL);
                $nrow = mysqli_num_rows($resultado);
                if ($nrow>0){
                    $row = mysqli_fetch_assoc($resultado);
                    $nome = ucwords(mb_strtolower($row['nome'], 'UTF-8'));
                    $ano_ingresso = $row['ano_ingresso'];
                    $semestre_ingresso = $row['semestre_ingresso'];
                    $documentos = $row['documentos'];
                    $curso_aluno = ucwords(mb_strtolower($row['curso'], 'UTF-8'));
                    $polo = ucwords(mb_strtolower($row['polo'], 'UTF-8'));
                    echo("
                        <div class='table100'>
                            <table>
                                <thead>
                                    <tr class='table100-head'>
                                        <th class='column1'>Matricula</th>
                                        <th class='column2'>Nome</th>
                                        <th class='column3'>Curso</th>
                                        <th class='column4'>Pólo</th>
                                        <th class='column5'>Ingresso</th>
                                        <th class='column6'>Documento</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    <tr>
                                        <td class='column1'>$matricula</td>
                                        <td class='column2'>$nome</td>
                                        <td class='column3'>$curso_aluno</td>
                                        <td class='column4'>$polo</td>
                                        <td class='column5'>$ano_ingresso/$semestre_ingresso</td>";
                                        if (strlen($documentos)>600){
                                            echo "<td class='column6'><a href='abrir_arquivo.php?id=$matricula'>abrir</a></td>";
                                        } else echo "<td class='column6'><a target='_blank' href='$documentos'>abrir</a></td>";
                                        echo "</tr></tbody>";
                                    } else echo "<div class='nofind'><p>Matrícula não encontrada.</p></div>";
                                }
                            </table></div>";
                }
            }
        </div></body></html>

```

Figura 28: Página "listar.php" - parte 2


```

<?php
try
{
    $conn = mysqli_connect("localhost", "root", "", "tcc");
    $consultaSQL = "SELECT documentos FROM aluno WHERE matricula=$_GET[id]";
    $resultado = mysqli_query($conn, $consultaSQL);

    $row=mysqli_fetch_assoc($resultado);
    $conteudo=$row['documentos'];
    $tipo = 'image/tiff';
    header("Content-Type: $tipo");
    echo $conteudo;
} catch(PDOException $erro)
{
    echo("Errrooooo! foi esse: " . $erro->getMessage());
}
?>

```

Figura 29: Página “abrir_arquivo.php”

```

function isInputName(evt){
    var ch = String.fromCharCode(evt.which);
    if(!(/[A-Za-zâãäåêëîíïóôõöúçñÀÁÂÃÄÅËÊËÌÍÎÏÓÔÕÖÙÇÑ ]/.test(ch))){
        evt.preventDefault();
    }
}

```

Figura 30: Função “isInputName()”

```

<?php
session_start();
if(!isset($_SESSION['id_user'])){
    header("location: login.php");
    exit;
}
function cadastrarAluno($matricula, $nome, $doc){
    require_once 'conn.php';
    $matricula = trim($matricula);
    $matricula = str_replace(".", "", $matricula);
    $semestre = $matricula[0];
    $ano = '20' . $matricula[1] . $matricula[2];
    $curso = $matricula[3] . $matricula[4] . $matricula[5];
    if($ano > date('Y')+3) {
        $ano = $ano - 100;
    }

    $find_curso_query = "select id_curso from curso where cod='$curso'";
    $resultado = mysqli_query($conn, $find_curso_query);
    $nrow = mysqli_num_rows($resultado);

    if ($nrow>0){
        $row = mysqli_fetch_assoc($resultado);
        $curso_id=$row['id_curso'];
        $insere_aluno_sql = "INSERT INTO ALUNO (MATRICULA, NOME, ANO_INGRESSO,
        SEMESTRE_INGRESSO, DOCUMENTOS, CURSO_ALUNO)
        VALUES ('$matricula', '$nome', '$ano', '$semestre', '$doc', '$curso_id')";
        $resultado_insere_aluno = mysqli_query($conn, $insere_aluno_sql);
        if ($resultado_insere_aluno){
            echo "<div class='queryok'>Inserido com sucesso!</div>";
        } else {
            $e=mysqli_error($conn);
            echo "<div class='queryfb'>Erro ao inserir. <br> '$e'.</div>";
        }
    } else echo "<div class='queryfb'>
    <span>Número de matrícula inválido. Curso inexistente.</span>
    </div>";
}
?>

```

Figura 31: Função “cadastrarAluno()” e validação de *user* logado

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="css/estilo_cadastrar.css">
<style>
.queryfb {width: 25%; background-color: red; margin-top: 7px; color: white; border-radius: 2px; padding-left: 4px; text-align: center; }
.queryok {width: 25%; background-color: green; margin-top: 7px; color: white; border-radius: 2px; padding-left: 4px; text-align: center; }
</style>
<title>TCC</title>
<script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
<div class="container">
<header>
<div><a href="http://www.uff.br/"></a></div>
<nav>
<ul>
<li>
<a href="index.php" class="btn" title="Consultar Aluno">Consultar Aluno</a>
</li>
<li>
<a href="search.php" class="btn" title="Inserir em Massa">Inserir em Massa</a>
</li>
<li>
<a class="btn" href="sair.php" title="Logout">Logout</a>
</li>
</ul>
</nav>
</header>
<div class="cover">
<h1>Cadastro de Aluno</h1>
<form class="form" method="post" tabindex="1" enctype="multipart/form-data" onsubmit="return isMatricula();">
<input type="text" id="matricula" name="matricula" placeholder="Matricula do Aluno" onkeypress="isInputNumber(event)" required />
<input type="text" name="nome" onkeypress="isInputName(event)" placeholder="Nome do Aluno" required />
<input type="file" id="arquivo" name="arquivo" accept="image/tiff" />
<label for="arquivo">Anexar Documento</label>
<input type="submit" name="enviar" value="Cadastrar">
</form>

```

Figura 32: Página “cadastrar.php” - parte 1

```

<?php
if (isset($_POST['enviar'])){
    $matricula = $_POST['matricula'];
    $nome = $_POST['nome'];
    $arquivo = $_FILES["arquivo"]["tmp_name"];
    $tamanho = $_FILES["arquivo"]["size"];
    $extensao = strtolower(substr($_FILES["arquivo"]["name"], -4));
    if (($nome!="" and $matricula!="" and $arquivo!="") and ($extensao==" .tif" or $extensao==" .tiff")){
        $fp = fopen($arquivo, "rb");
        $conteudo = fread($fp, $tamanho);
        $conteudo = addslashes($conteudo);
        fclose($fp);
        // include("cadastra_aluno.php");
        cadastrarAluno($matricula, $nome, $conteudo);
    } else echo "<div class='queryfb'> <p>Por favor, insira um documento do tipo tif.</p></div>";
}
?>
</div>
</body>
</html>

```

Figura 33: Página “cadastrar.php” - parte 2

```
<?php
// Copiar as pastas GRADUACAO E CEDEHU para dentro de C:\xampp\htdocs\tcc\Scan\ATUALIZACAO\
$dir = 'C:\xampp\htdocs\tcc\Scan\ATUALIZACAO\GRADUACAO\';
$graduacao_ano = scandir($dir);
require_once 'conn.php';
for ($i=2; $i < count($graduacao_ano); $i++){
    $vestibular_ano = $graduacao_ano[$i];
    $vestibular_dir = scandir($dir.$vestibular_ano);
    for ($x=2; $x < count($vestibular_dir); $x++){
        $nome_curso = $vestibular_dir[$x];
        $curso_dir = scandir($dir.$vestibular_ano.'\'\''. $nome_curso);
        for ($y=2; $y < count($curso_dir); $y++){
            $semestre = $curso_dir[$y];
            $semestre_dir = scandir($dir.$vestibular_ano.'\'\''. $nome_curso.'\'\''. $semestre);
            for ($z=2; $z < count($semestre_dir); $z++){
                $aluno = $semestre_dir[$z];
                $matricula = strpbrk($aluno, '0123456789');
                $len_nome = strlen($aluno) - strlen($matricula);
                $nome = trim(ucwords(mb_strtolower(substr($aluno, 0, $len_nome))));
                $dir_final = $dir.$vestibular_ano.'\'\''. $nome_curso.'\'\''. $semestre.'\'\''. $aluno;
                $doc = scandir($dir_final);
                $iano = '20'. $matricula[1]. $matricula[2];
                $isem = $matricula[0];
                $curso = $curso = $matricula[3]. $matricula[4]. $matricula[5];
                $find_curso_query = "select id_curso from curso where cod='$curso'";
                $find_matricula_query = "select matricula from consulta_aluno where matricula='$matricula'";
                $resultado_curso = mysqli_query($conn, $find_curso_query);
                $resultado_matricula = mysqli_query($conn, $find_matricula_query);
                $nrow_mat = mysqli_num_rows($resultado_matricula);
                $nrow = mysqli_num_rows($resultado_curso);
```

Figura 34: Página “search.php” - parte 1

```
if ($row=0){  
    if ($row_mat==0) {  
        for ($w=2; $w < count($doc); $w++){  
            $documento=$dir_final.'\\'.$doc[$w];  
            $doc_final.substr($documento, 20); //substituir o numero 20 pelo numero de letras até a pasta o início da  
palavra Scan (contando com as barras)  
            $doc_final = mysql_real_escape_string($conn, $doc_final);  
            $resultado = mysqli_query($conn,"INSERT INTO ALUNO (matricula, nome, ano_ingresso, semestre_ingresso,  
documentos, curso_aluno) VALUES ('$matricula', '$nome', '$ano', '$sem', '$doc_final', '$curso')");  
            if ($resultado) echo "Inseriu aluno $nome de matricula $matricula com sucesso, <br>";  
            else {echo "$nome de matricula $matricula não foi inserido. erro: "; echo mysqli_error($conn) "<br>;"  
                }  
        }  
    } else echo "Código de curso inválido para o aluno $nome de matricula $matricula <br>";  
}  
}  
}  
}</div></div>  
</body>  
</html>
```

Figura 35: Página “search.php” - parte 2


```
<?php
$user= 'root';
$pwd= '';
$db='tcc';
$host = 'localhost';

//define("MYSQL_CONN_ERROR", "Unable to connect to database.");
mysqli_report(MYSQLI_REPORT_STRICT);

try {
    $conn = mysqli_connect($host, $user, $pwd, $db);
    mysqli_query($conn,"SET NAMES 'utf8'");
} catch (Exception $e) {
    echo "Não foi possível conectar a base: ".$e;
    die();
}

?>
```

Figura 36: Arquivo “conn.php”