Ciclo de vida do desenvolvimento de software

Fábio Lucas

Mateus Machado

Leonardo Lins

Atividade de Requisitos

“SACAD.”

O que são requisitos?

- Requisitos são as **necessidades** do cliente!
- Os requisitos de um software são divididos em 2 categorias:
  - Requisitos Funcionais: são as tarefas, as funcionalidades de um sistema – eles são descritos na forma de verbos no infinitivo, pois são ações que os usuários executam em um software (ex: sacar dinheiro, buscar um livro, comprar um produto e etc)
  - Requisitos Não Funcionais: são as qualidades que um sistema precisa possuir para atender adequadamente as necessidades do usuário – eles são descritos na forma de adjetivos (ex. Usabilidade, segurança, desempenho e etc)

Requisitos Funcionais

“SACAD.”

Diagrama de Casos de Uso

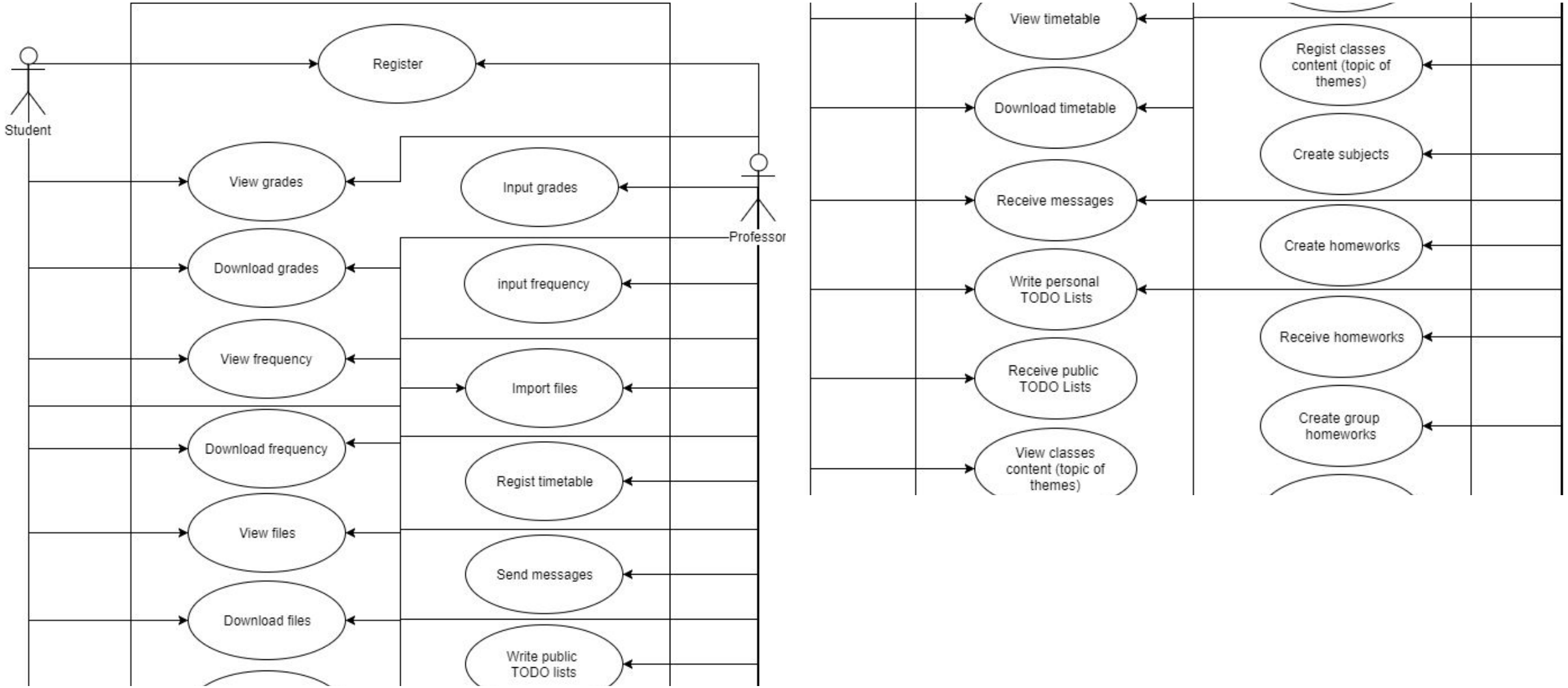
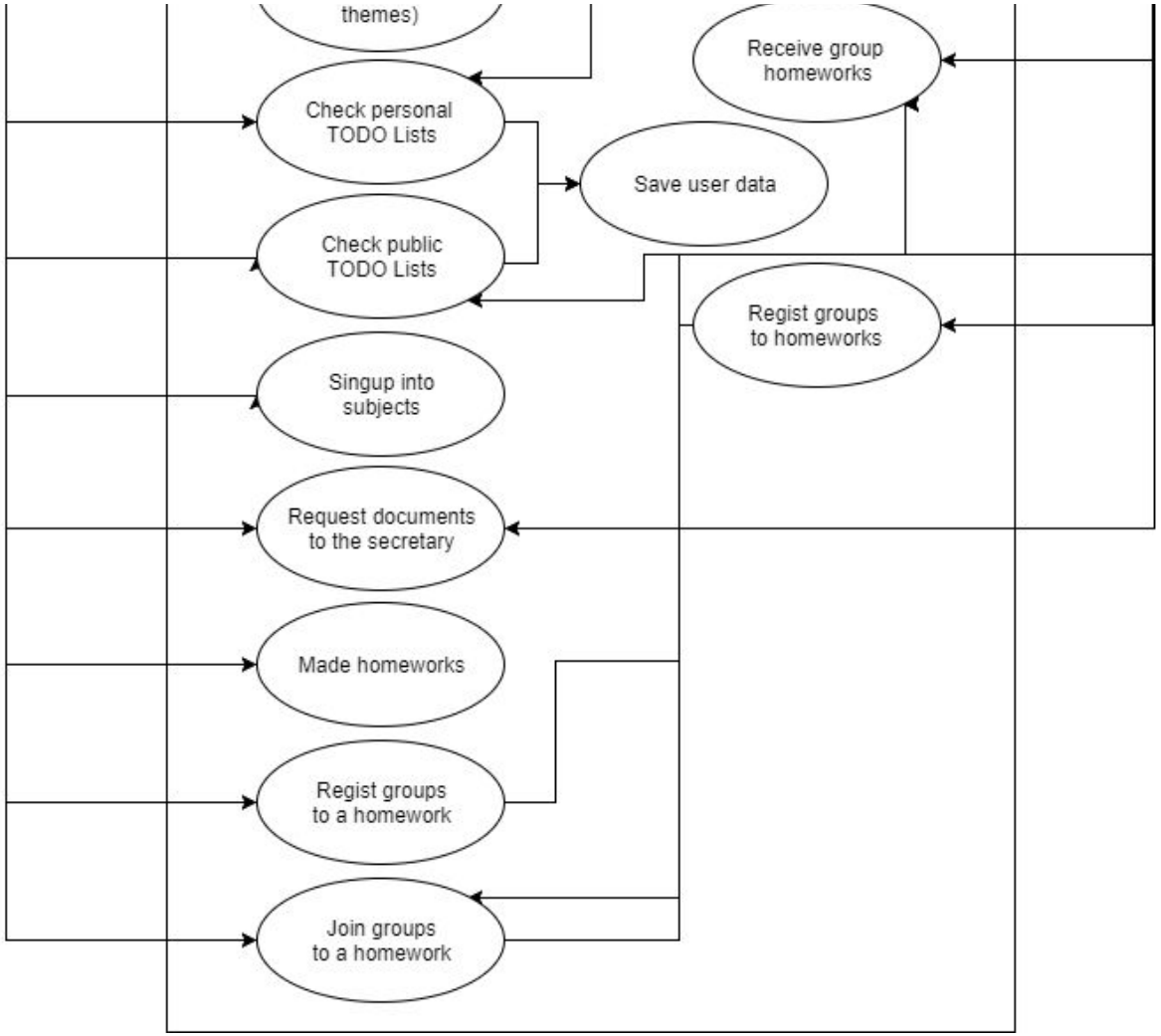


Diagrama de Casos de Uso



User case	Registrar usuário
Ator Principal	Usuário
Resumo	Um usuário pode se registrar
Pré-Condição	Não estar registrado
- Ações do Ator	-Ações do sistema
1 - Clicar em se registrar	
	2 - Abrir formulário de registro
3 - Preencher o formulário (Nome, e-mail, user, senha, cód)	
	4 - Nenhum
5 - Enviar formulário	
	6 - Validar formulário
	7 - Registrar Usuário
Restrições/Validações	Usuário precisa ser único no database
	Código precisa ser válido (Fornecido pela Fatec)
Fluxo de Exceção	
5 - Envar formulário	
	6 - Validar formulário
	7 - Não registrar formulário

User Case	Visualizar notas
Ator principal	Usuário
Resumo	O usuário pode visualizar notas
Pré-Condição	Nenhuma
- Ações do ator	- Ações do sistema
1 - Clicar em visualizar notas	
	2 - Abrir página com as notas
Restrições/Validações	Nenhuma
Fluxo de Exceção	Nenhum

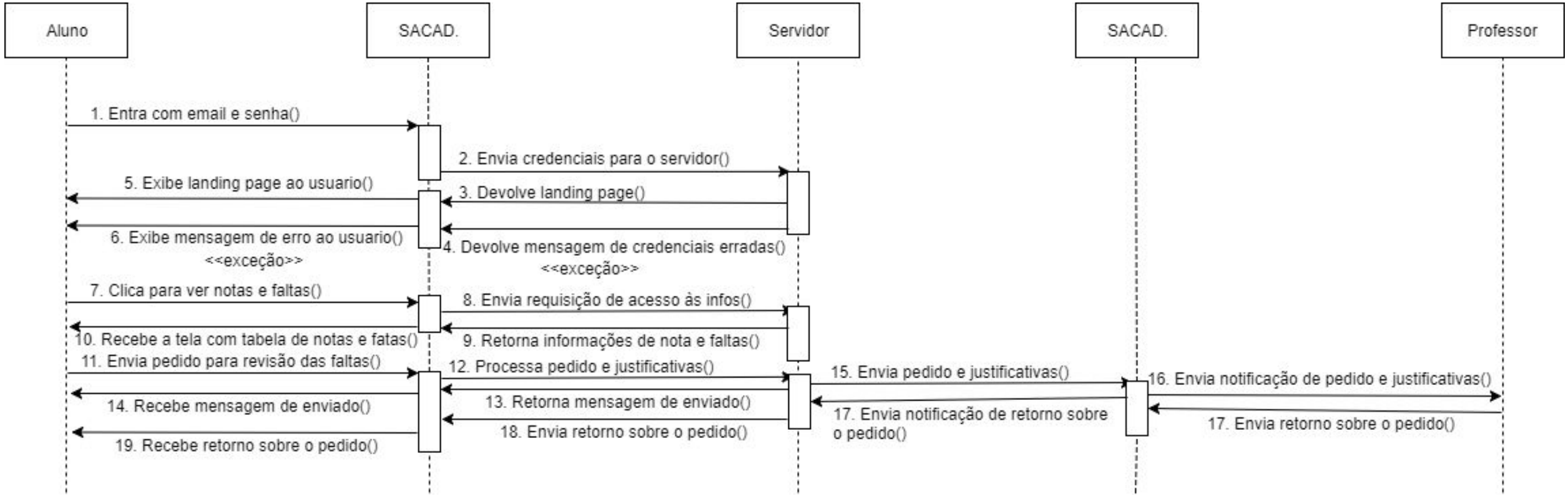
User Case	Baixar Notas
Ator Principal	Usuário
Resumo	O usuário pode baixar as notas
Pré-Condição	Existirem notas
- Ações do ator	- Ações do sistema
1 - Clicar em visualizar notas	
	2 - Abrir página com as notas
	3 - Caso existam notas cadastradas, exibir opção de download
4 - Clicar em download	
Restrições/Validações	Nenhuma

User case	Registrar Notas
Ator Principal	Usuário
Resumo	Um usuário (professor) pode inserir notas
Pré-Condição	Estar registrado
- Ações do Ator	- Ações do Sistema
1 - Abrir o formulário de preenchimento de notas	
	2 - Fornecer o formulário
3 - Preencher o formulário	
4 - Enviar formulário	
	5 - Validar se 1 <= notas <= 10
	6 - Registrar as notas
Fluxo de exceção	
3 - Preencher o formulário	
4 - Enviar formulário	
	5 - Validar se 1 <= notas <= 10
	6 - Não registra as notas
	7 - Exibe mensagem para ator
Restrições	Estar logado
Fluxo exceção	Nenhuma

User case	Inserir frequência
Ator principal	usuário (professor) pode inserir frequência
Pré-condição	Estar em período letivo
- Ações do ator	- Ações do sistema
1 - Abrir o formulário de chamada	
	2 - Fornecer o formulário
3 - Preencher o formulário	
4 - Submeter o formulário	
	5 - Gravar no banco
Restrições/Validações	Todos os alunos devem estar com presença ou falta
Fluxo de exceção	
4 - submeter o formulário	
	5 - não é gravado no banco
	6 - o formulário é devolvido indicando onde está errado
Restrições	Estar logado
Exceção	Nenhuma

Use case	Importar arquivos
Ator principal	usuário (professor) pode disponibilizar arquivos para alunos
Pré-condição	Arquivo deve ser menor do que xxxxxx MB (a decidir)
- Ações do ator	- Ações do sistema
1 - Abrir a tela de upload de arquivos	
2 - Escolher o arquivo	
	3 - Validar se o tamanho do arquivo < XXXX MB
	4 - Disponibilizar para os alunos
Restrições/Validações	O arquivo deve ser menor do que XXX MB
Fluxo de exceção	
2 - Escolher o arquivo	
	3 - Validar se o tamanho do arquivo < XXXX MB
	4 - Exibe mensagem de arquivo muito grande
Restrições	Estar logado
Exceção	Nenhuma

Professor
SACAD.
Servidor
1. Entra com email e senha()
2. Envia credenciais para o servidor()
3. Devolve landing page()
4. Devolve mensagem de credenciais erradas()
<<exceção>>
5. Exibe landing page ao usuario()
6. Exibe mensagem de erro ao usuario()
<<exceção>>
7. Seleciona a turma desejada()
8. Envia requisição de acesso à turma()
9. Retorna informações da turma()
10. Recebe a tela com informações da turma()
11. Abre modo de edição de notas e faltas()
12. Busca todas as notas e faltas dos alunos()
13. Retorna notas e faltas()
14. Recebe notas e faltas editaveis()
15. Faz alterações ou entra com informações()
16. Faz validação dos dados()
17. Envia erro de validação()
<<exceção>>
18. Exibe erro de validação()
<<exceção>>
19. Envia feedback de sucesso e novas infos()
20. Exibe mensagem de sucesso e
exibe novas informacoes()


Aluno
SACAD.
Servidor
SACAD.
Professor
1. Entra com email e senha()
2. Envia credenciais para o servidor()
5. Exibe landing page ao usuario()
3. Devolve landing page()
6. Exibe mensagem de erro ao usuario()
<<exceção>>
4. Devolve mensagem de credenciais erradas()
<<exceção>>
7. Clica para ver notas e faltas()
8. Envia requisição de acesso às infos()
10. Recebe a tela com tabela de notas e fatas()
9. Retorna informações de nota e faltas()
11. Envia pedido para revisão das faltas()
12. Processa pedido e justificativas()
15. Envia pedido e justificativas()
16. Envia notificação de pedido e justificativas()
14. Recebe mensagem de enviado()
13. Retorna mensagem de enviado()
17. Envia notificação de retorno sobre o pedido()
17. Envia retorno sobre o pedido()
18. Envia retorno sobre o pedido()
19. Recebe retorno sobre o pedido()

Requisitos Não Funcionais “Usabilidade”

“SACAD.”

1 – VISIBILIDADE DE STATUS DO SISTEMA

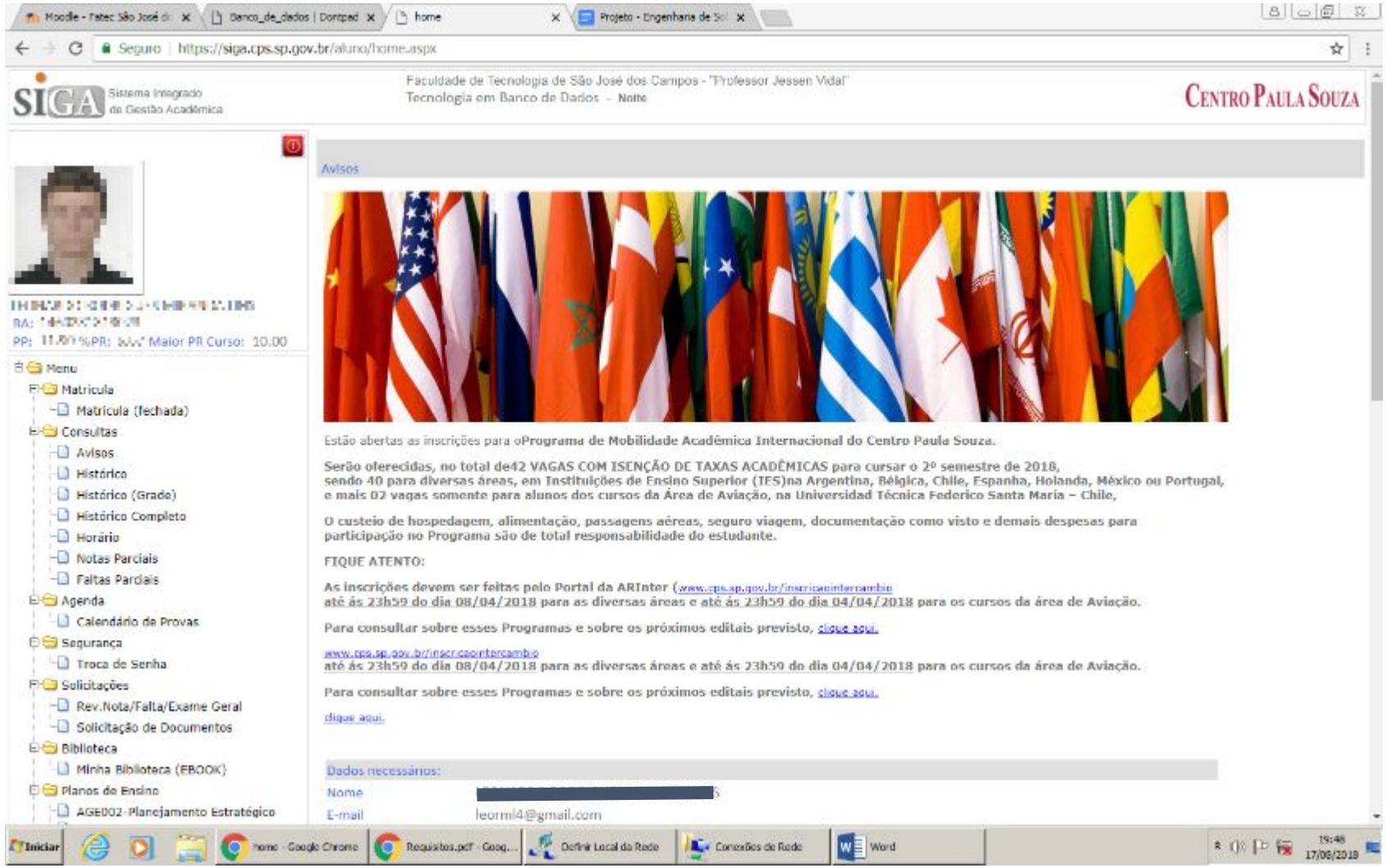


Figura 1 - Página do SIGA não apresenta visibilidade de conexão do site

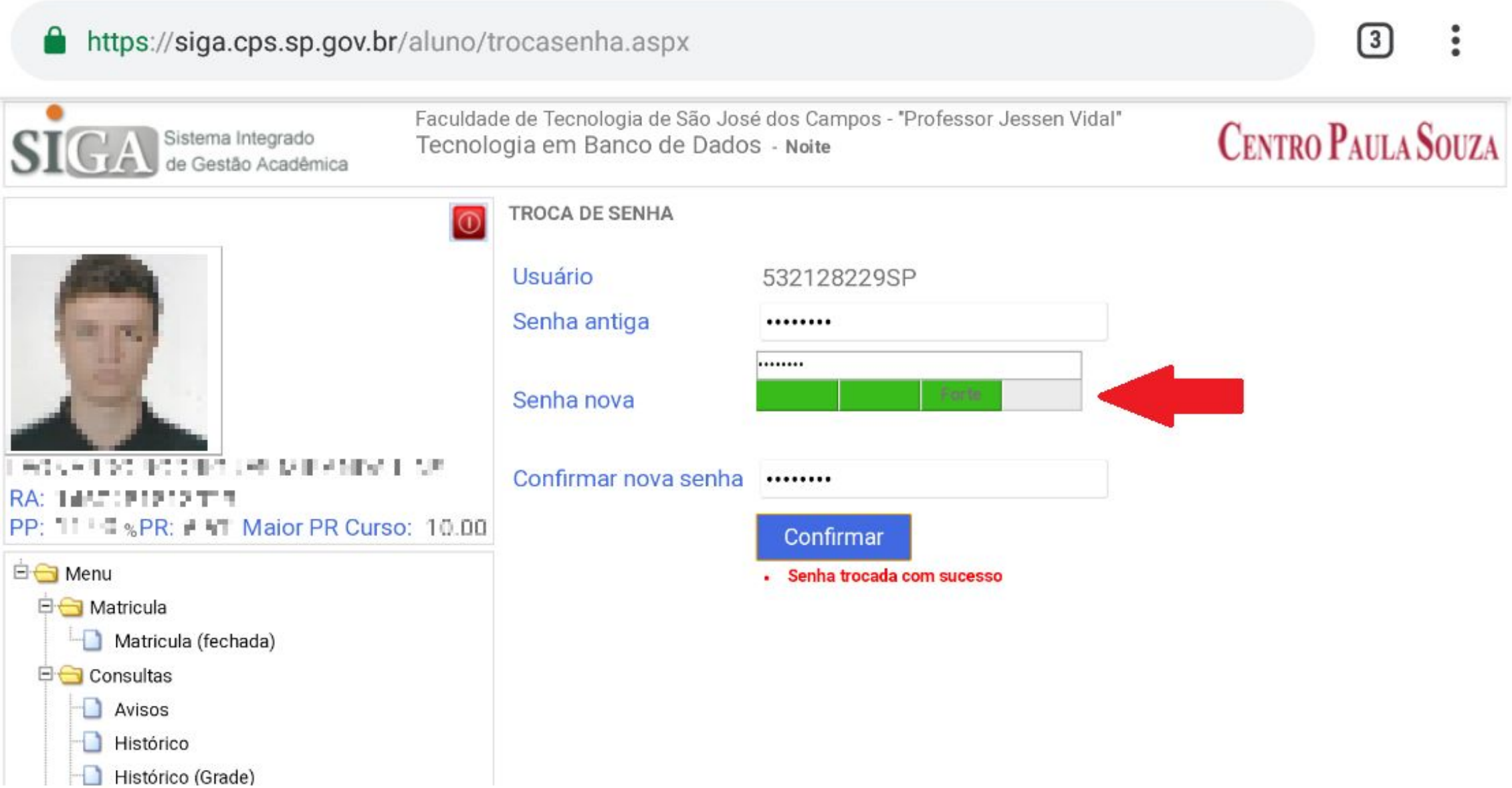


Figura 2 - Feedback enquanto usuário insere senha

2 – FALAR A LINGUAGEM DO USUÁRIO

Figura 3 - Siglas pouco conhecidas, sem detalhe do significado.

3 - CONTROLE E LIBERDADE DE USUÁRIO

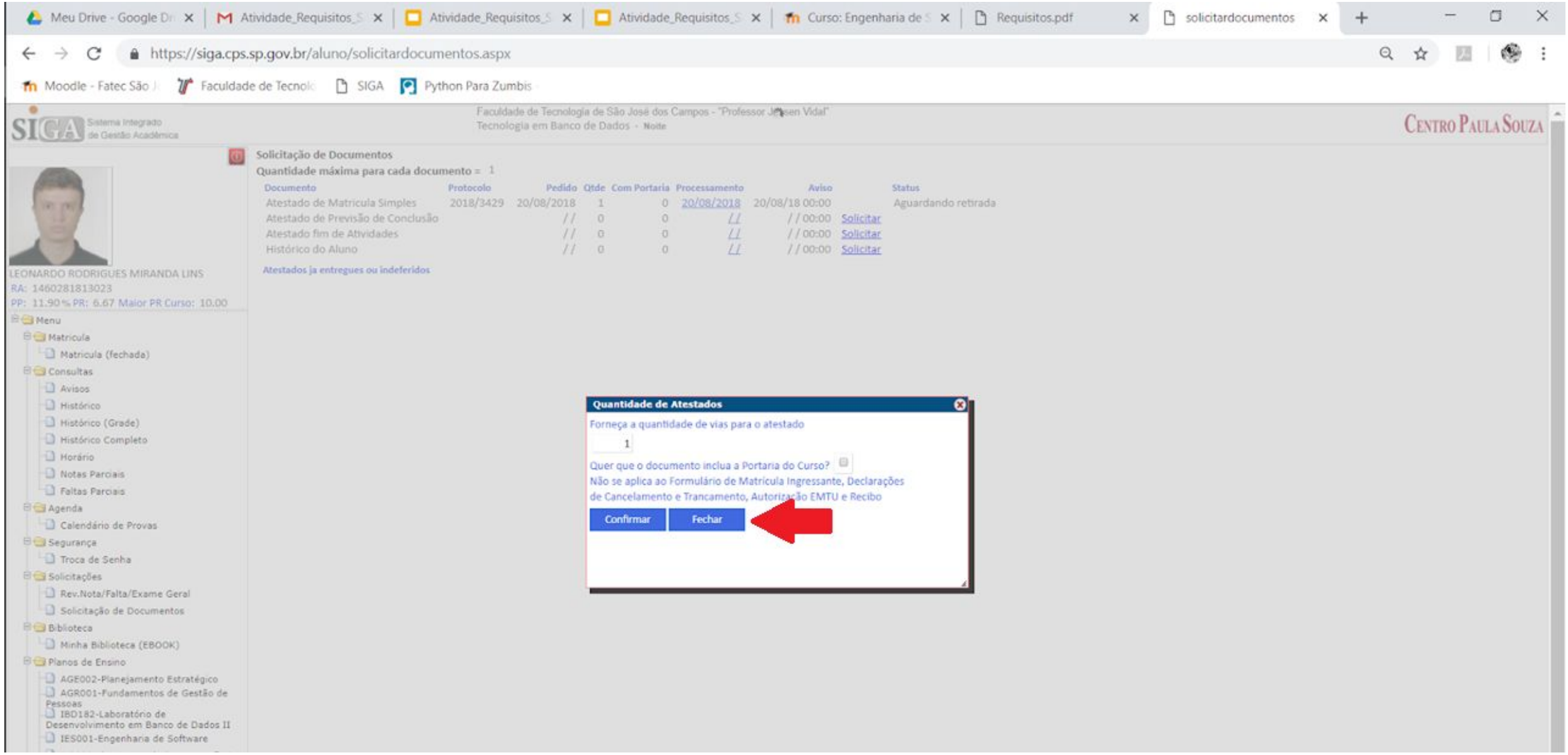


Figura 4 - Capacidade de retroceder caso escolha a opção errada.

4. CONSISTÊNCIA E PADRÕES

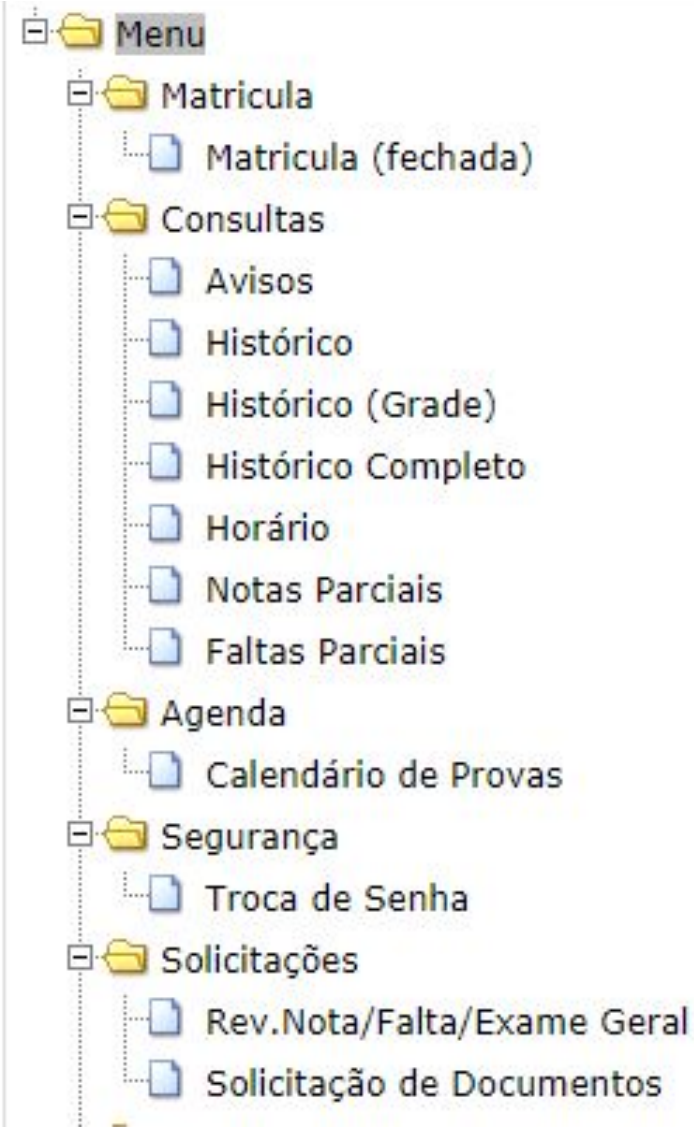


Figura 5 - Árvore de diretórios apresentada pela aplicação

https://siga.cps.sp.gov.br/aluno/home.aspx#

SIGA Sistema Integrado de Gestão Acadêmica

Faculdade de Tecnologia de São José dos Campos - "Professor Jessen Vidal"
Tecnologia em Banco de Dados - Noite

CENTRO PAULA SOUZA

LEONARDO RODRIGUES MIRANDA LINS
RA: 1460281813023
PP: 11.90 %PR: 6.67 Maior PR Curso: 10.00

- Menu
  - Matricula
    - Matricula (fechada)
  - Consultas
    - Avisos
    - Histórico
    - Histórico (Grade)
    - Histórico Completo
    - Horário
    - Notas Parciais
    - Faltas Parciais
  - Agenda
    - Calendário de Provas

Avisos

Estão abertas as inscrições para o Programa de Mobilidade Acadêmica Internacional do Centro Paula Souza, para maiores informações, acesse: http://www.cps.sp.gov.br/internacional/

Dados necessários:

Nome	LEONARDO RODRIGUES MIRANDA LINS
E-mail	leorml4@gmail.com
CPF	45079767820
Data de nascimento	20/05/1998
Índices	
PP: 11.90	PR: 6.67

Figura 6 - Botão sair do lado esquerdo

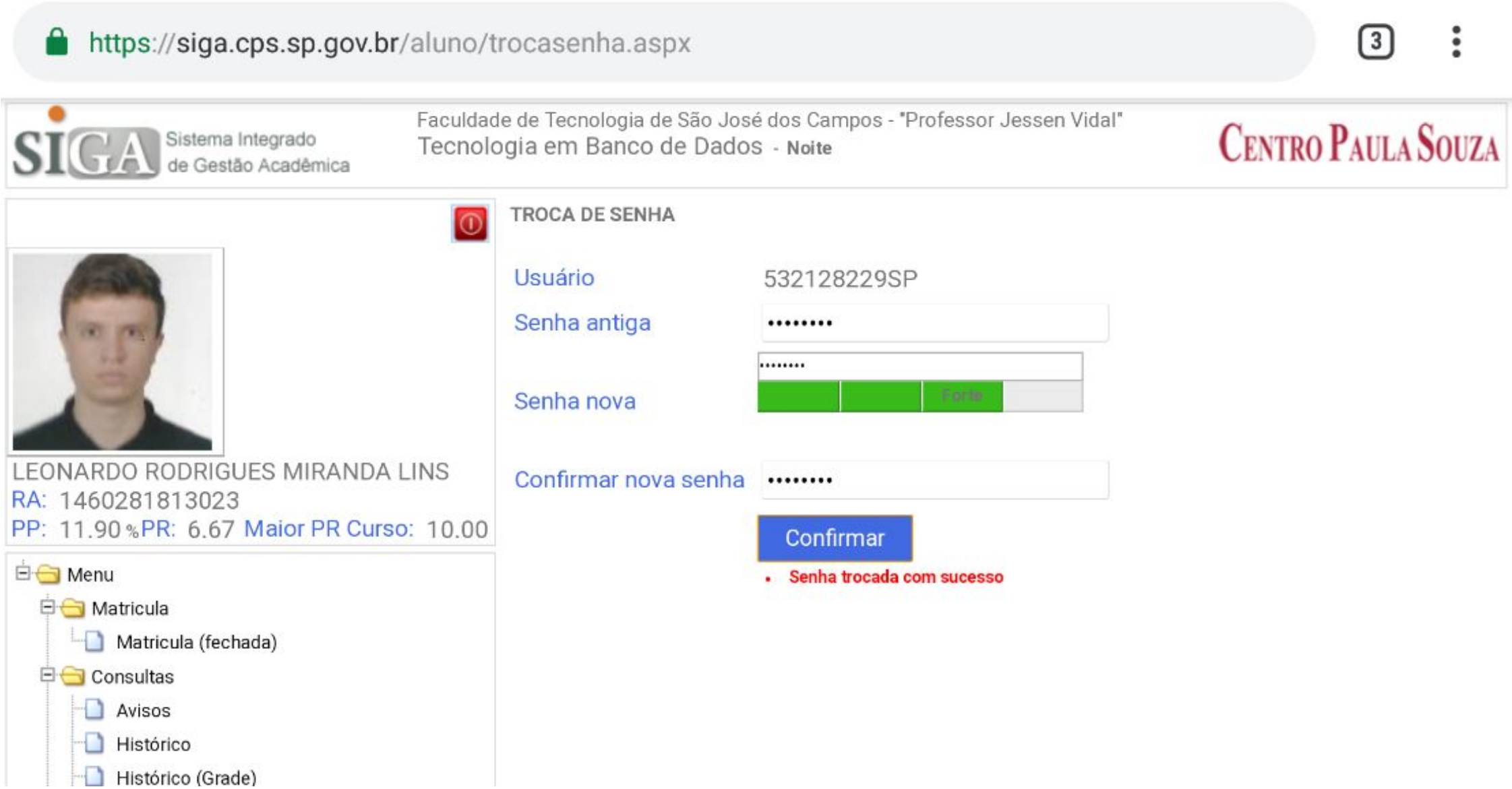


Figura 7 - Feedback positivo com cor em vermelho

5. PREVENÇÃO DE ERROS

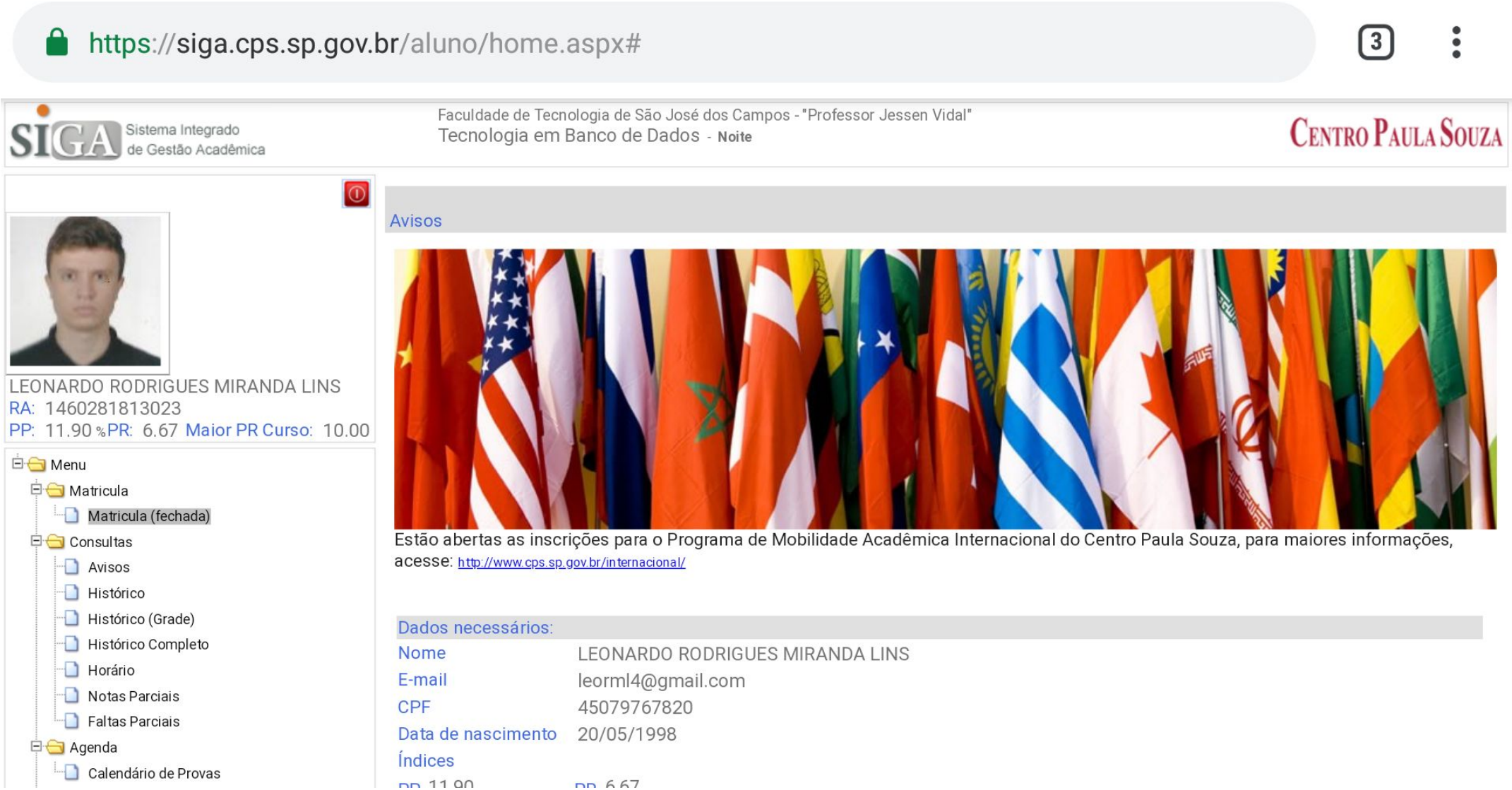


Figura 8 - Botão “Matrícula” não apresenta resultado nenhum

6. RECONHECIMENTO AO INVÉS DE LEMBRANÇA

Faculdade de Tecnologia de São José dos Campos - "Professor Jessen Vidal"
Tecnologia em Banco de Dados - Noite

CENTRO PAULA SOUZA

Sigla	Disciplina	Turma	Professor
AGE002	Planejamento Estratégico - 2hs/aula	A	VALTER JOÃO DE SOUSA
AGR001	Fundamentos de Gestão de Pessoas - 2hs/aula	A	GERALDO JOSE LOMBARDI DE SOUZA
IBD182	Laboratório de Desenvolvimento em Banco de Dados II - 4hs/aula	A	ADRIANA DA SILVA JACINTO
IES001	Engenharia de Software - 4hs/aula	A	GIULIANO ARAUJO BERTOTI
ILP008	Linguagem de Programação I - 4hs/aula	A	ADRIANA DA SILVA JACINTO
IMB003	Arquitetura e Modelagem de Banco de Dados - 4hs/aula	A	EMANUEL MINEDA CARNEIRO
MCA001	Fundamentos de Cálculo - 2hs/aula	A	DERCY FELIX DA SILVA
TAA100	Atividades Acadêmico-científico-culturais - 2hs/aula	A	FABIANO SABHA WALCZAK

Segunda-Feira

	Horário	Disciplina	Turma
	20:25-21:15	AGE002	A
»	19:35-20:25	IES001	A
»	18:45-19:35	IES001	A
	21:25-22:15	MCA001	A
	22:15-23:05	MCA001	A

Quarta-Feira

Horário	Disciplina	Turma
19:35-20:25	IBD182	A
20:25-21:15	IBD182	A
18:45-19:35	IBD182	A
21:25-22:15	ILP008	A
22:15-23:05	ILP008	A

Sexta-Feira

	Horário	Disciplina	Turma
»	19:35-20:25	IES001	A
»	18:45-19:35	IES001	A
	21:25-22:15	IMB003	A
	22:15-23:05	IMB003	A

Terça-Feira

Horário	Disciplina	Turma
20:25-21:15	AGE002	A
19:35-20:25	AGR001	A
18:45-19:35	AGR001	A
21:25-22:15	ILP008	A
22:15-23:05	ILP008	A

Quinta-Feira

Horario	Disciplina	Turma
20:25-21:15	IBD182	A
21:25-22:15	IMB003	A
22:15-23:05	IMB003	A

Sábado

Horario	Disciplina	Turma
08:00-08:50	TAA100	A
08:50-09:40	TAA100	A

Imprimir

Figura 9 - Marcar para usuário a disciplina, ao invés de lembrar.

7. FLEXIBILIDADE E EFICIÊNCIA DE USO

Sigla	Disciplina	Turma	Professor
AGE002	Planejamento Estratégico - 2hs/aula	A	VALTER JOÃO DE SOUSA
AGR001	Fundamentos de Gestão de Pessoas - 2hs/aula	A	GERALDO JOSE LOMBARDI DE SOUZA
IBD182	Laboratório de Desenvolvimento em Banco de Dados II - 4hs/aula	A	ADRIANA DA SILVA JACINTO
IES001	Engenharia de Software - 4hs/aula	A	GIULIANO ARAUJO BERTOTI
ILP008	Linguagem de Programação I - 4hs/aula	A	ADRIANA DA SILVA JACINTO
IMB003	Arquitetura e Modelagem de Banco de Dados - 4hs/aula	A	EMANUEL MINEDA CARNEIRO
MCA001	Fundamentos de Cálculo - 2hs/aula	A	DERCY FELIX DA SILVA
TAA100	Atividades Acadêmico-científico-culturais - 2hs/aula	A	FABIANO SABHA WALCZAK

Segunda-Feira

Horário	Disciplina	Turma
20:25-21:15	AGE002	A
19:35-20:25	IES001	A
18:45-19:35	IES001	A
21:25-22:15	MCA001	A
22:15-23:05	MCA001	A

Quarta-Feira

Horário	Disciplina	Turma
19:35-20:25	IBD182	A
20:25-21:15	IBD182	A
18:45-19:35	IBD182	A
21:25-22:15	ILP008	A
22:15-23:05	ILP008	A

Sexta-Feira

Horário	Disciplina	Turma
19:35-20:25	IES001	A
18:45-19:35	IES001	A
21:25-22:15	IMB003	A
22:15-23:05	IMB003	A

Terça-Feira

Horário	Disciplina	Turma
20:25-21:15	AGE002	A
19:35-20:25	AGR001	A
18:45-19:35	AGR001	A
21:25-22:15	ILP008	A
22:15-23:05	ILP008	A

Quinta-Feira

Horario	Disciplina	Turma
20:25-21:15	IBD182	A
21:25-22:15	IMB003	A
22:15-23:05	IMB003	A

Sábado

Horario	Disciplina	Turma
08:00-08:50	TAA100	A
08:50-09:40	TAA100	A

Imprimir

Figura 10 - Praticidade do botão imprimir.

8. ESTÉTICA E DESIGN MINIMALISTA

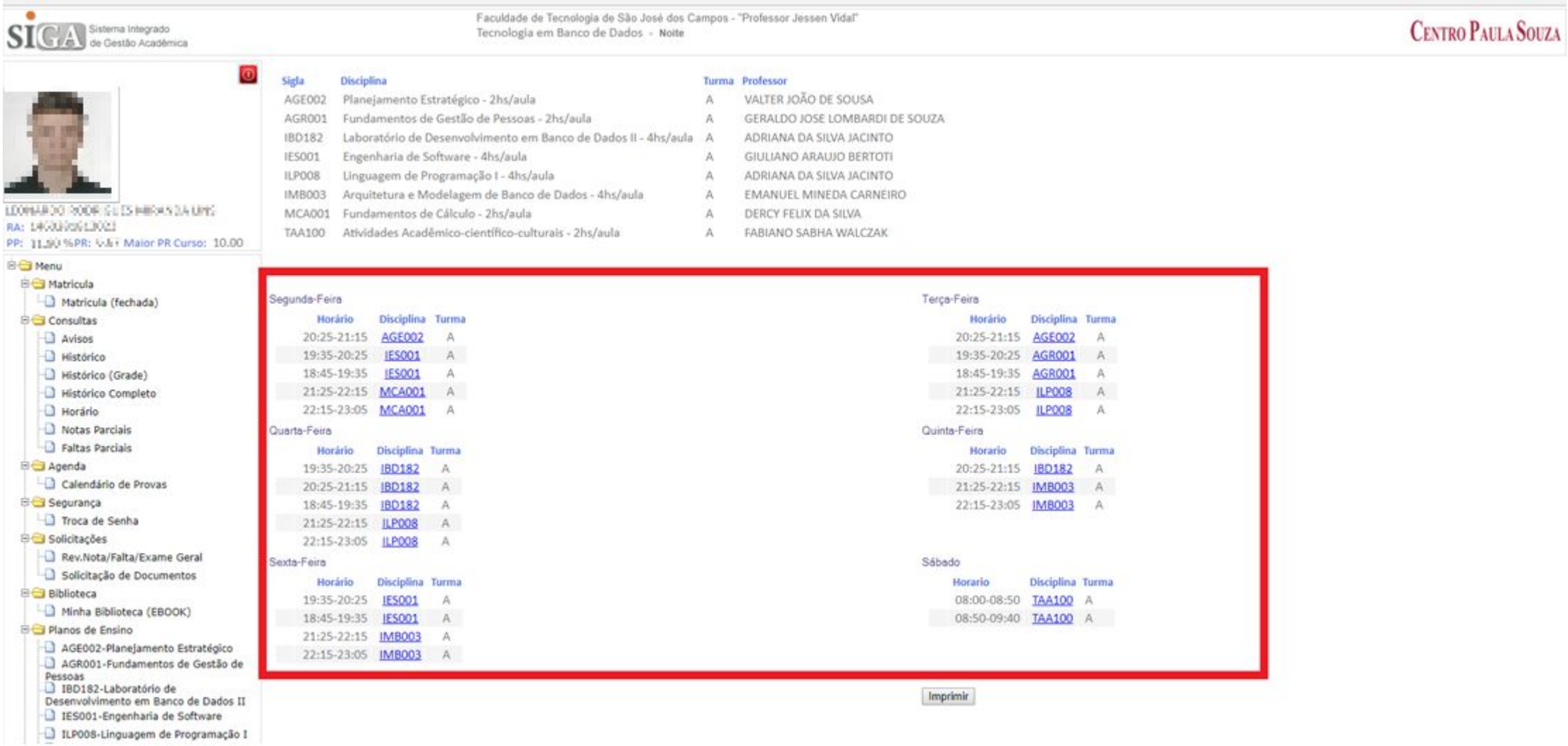


Figura 11 – Horário com visibilidade ruim.

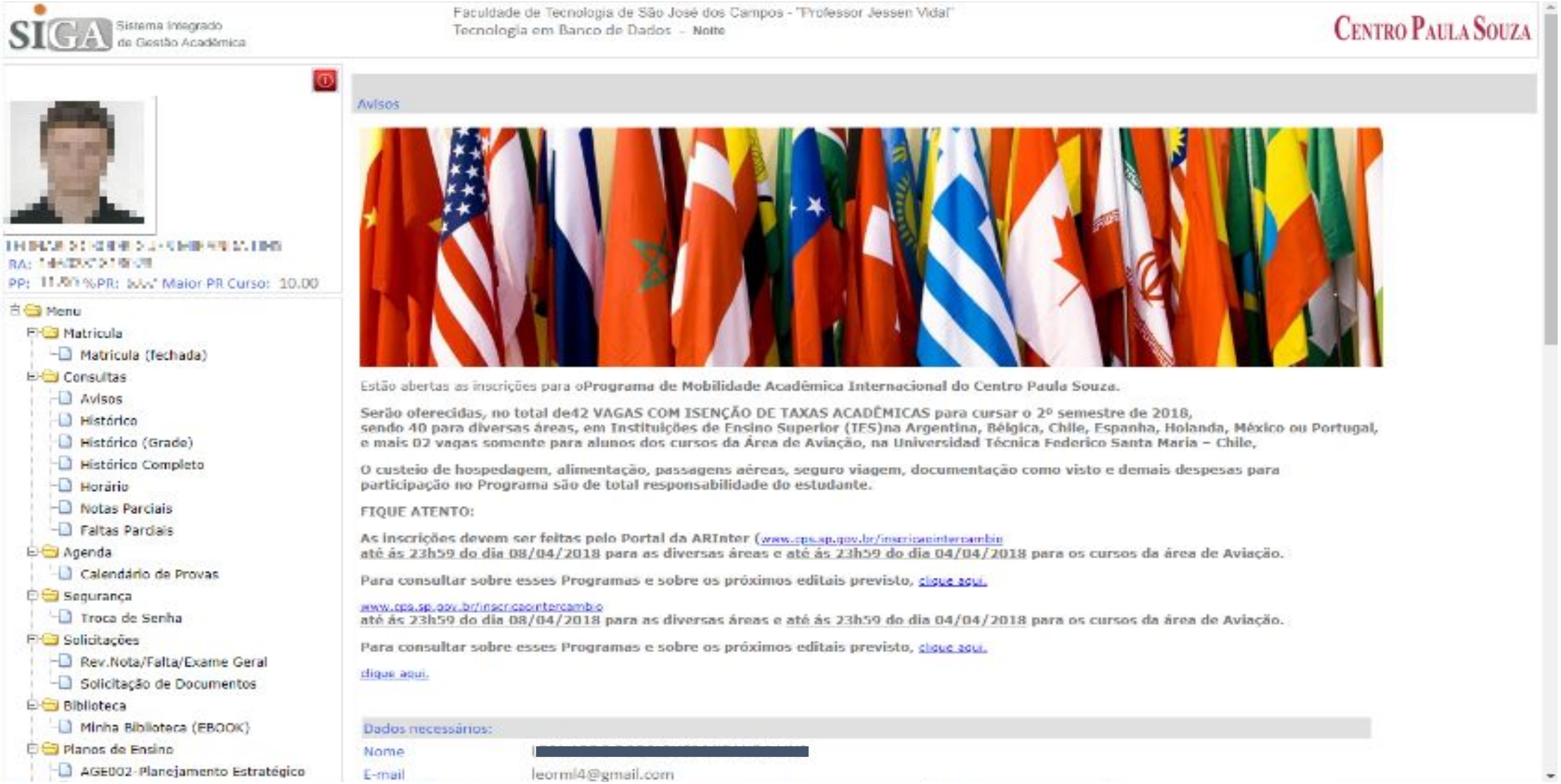


Figura 12 – Tela com muitas opções, muitos específicas e pouco categorizadas.

9. AJUDAR O USUÁRIO A RECONHECER, DIAGNOSTICAR E CORRIGIR PROBLEMAS

Figura 13 – Advertência a respeito solicitações fora de prazo.

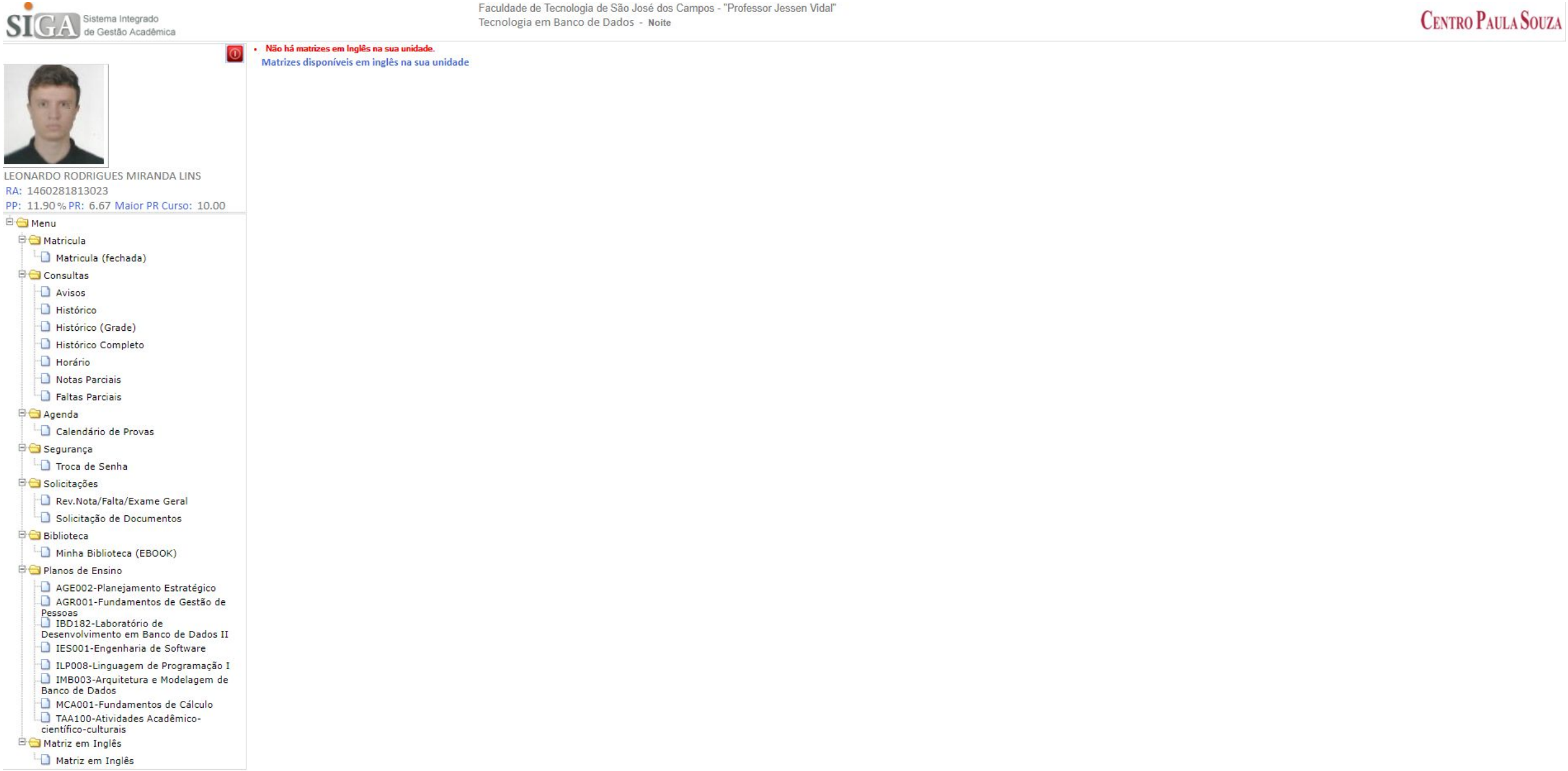


Figura 14 – Advertência sobre indisponibilidade da opção para unidade.

10. AJUDA E DOCUMENTAÇÃO

SOLICITAÇÃO DE REVISÃO DE NOTAS/FALTAS OU EXAME GERAL

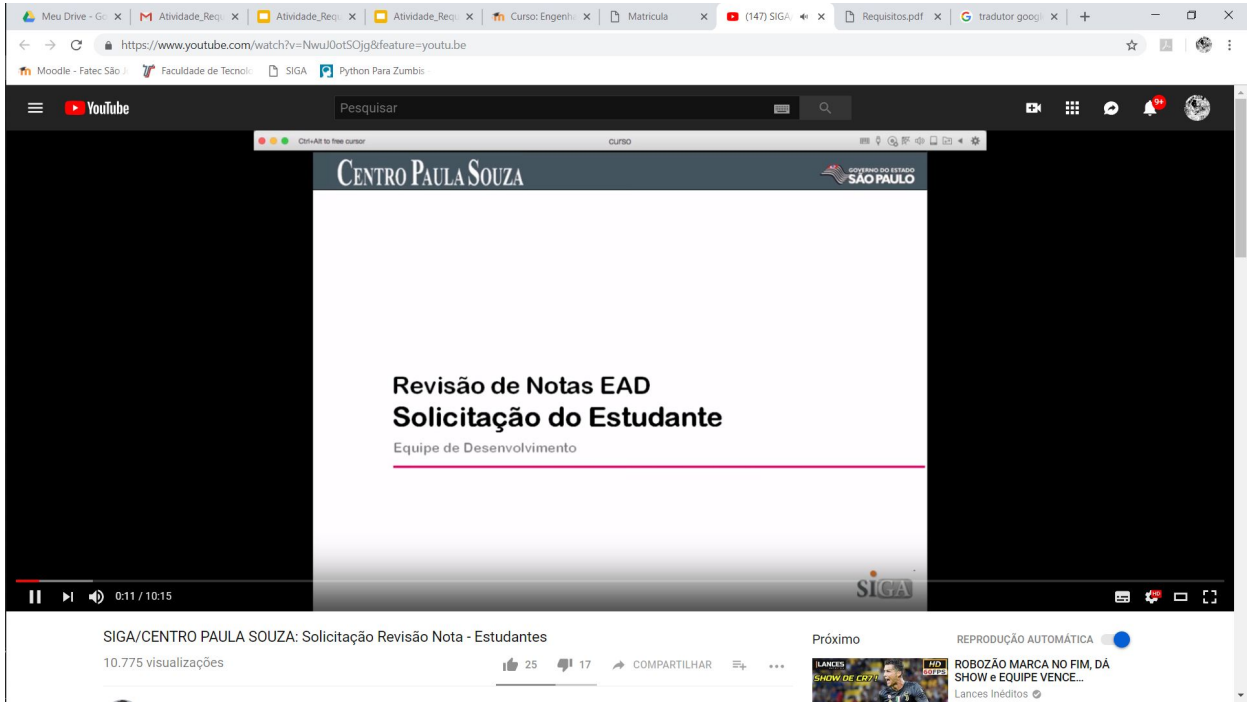
Status	
Aberto	Solicitação de revisão que ainda não foi avaliada pelo Coordenador de Curso
Pendente	Solicitação que se encontra com o docente para avaliação
Deferido	A solicitação foi deferida pelo docente da disciplina e a Nota/Falta foi lançada na Errata
Indeferido	A solicitação foi indeferida pelo docente da disciplina ou pelo Coordenador de Curso

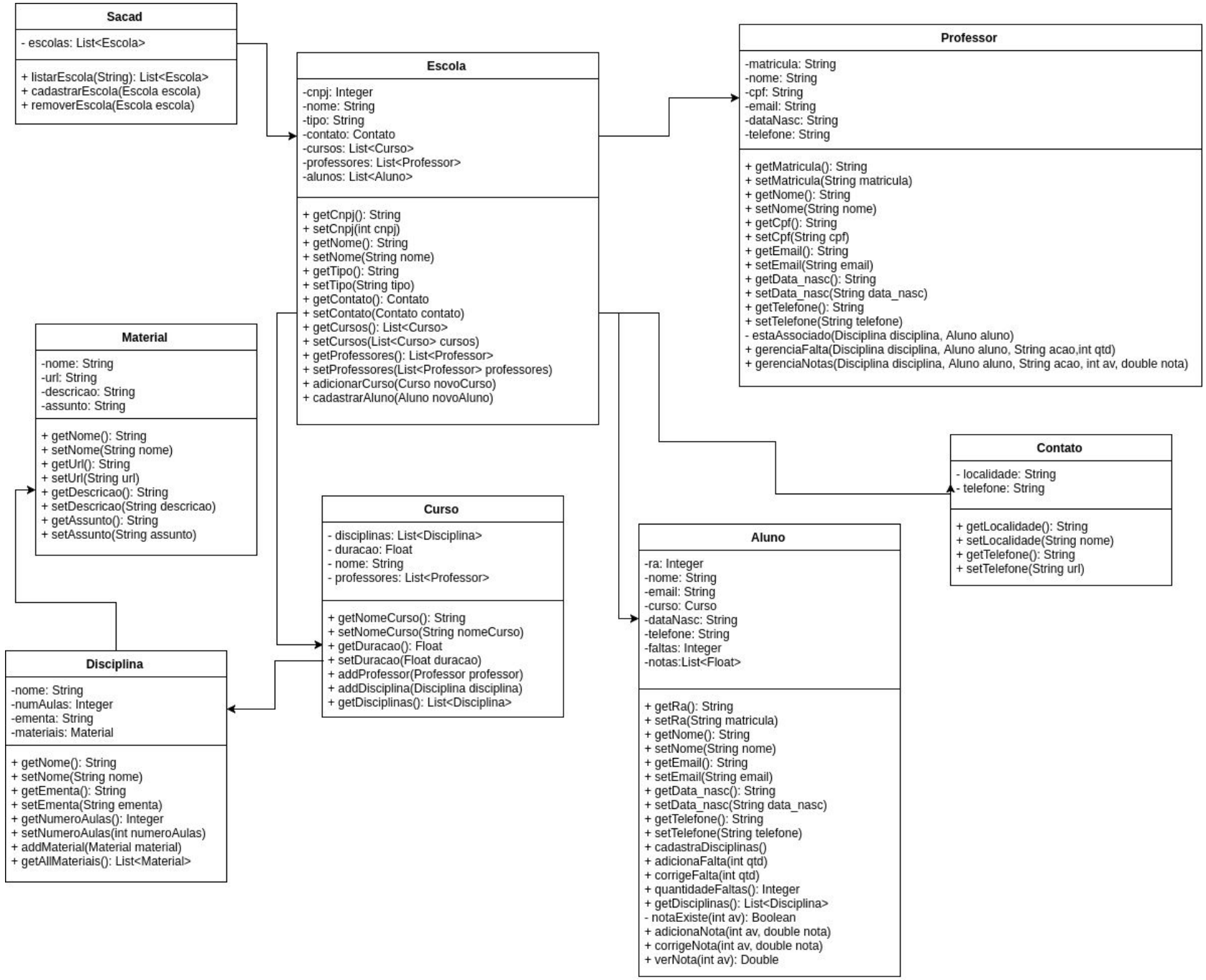
Disciplinas Matriculadas no Semestre:	Periodo inicial de Revisão	Final
	01/08/2018	16/08/2018

Não é permitido solicitar revisões de médias ou faltas fora do período!

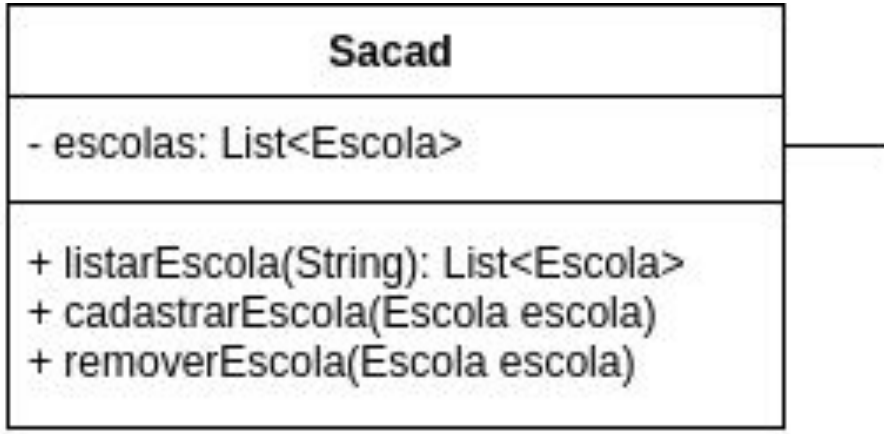
Video: Solicitação de Revisoes

Figura 15 – Ajuda com o recurso.




Sacad
- escolas: List<Escola>
+ listarEscola(String): List<Escola>
+ cadastrarEscola(Escola escola)
+ removerEscola(Escola escola)
Escola
-cnpj: Integer
-nome: String
-tipo: String
-contato: Contato
-cursos: List<Curso>
-professores: List<Professor>
-alunos: List<Aluno>
+ getCnpj(): String
+ setCnpj(int cnpj)
+ getNome(): String
+ setNome(String nome)
+ getTipo(): String
+ setTipo(String tipo)
+ getContato(): Contato
+ setContato(Contato contato)
+ getCursos(): List<Curso>
+ setCursos(List<Curso> cursos)
+ getProfessores(): List<Professor>
+ setProfessores(List<Professor> professores)
+ adicionarCurso(Curso novoCurso)
+ cadastrarAluno(Aluno novoAluno)
Professor
-matricula: String
-nome: String
-cpf: String
-email: String
-dataNasc: String
-telefone: String
+ getMatricula(): String
+ setMatricula(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getCpf(): String
+ setCpf(String cpf)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
- estaAssociado(Disciplina disciplina, Aluno aluno)
+ gerenciaFalta(Disciplina disciplina, Aluno aluno, String acao,int qtd)
+ gerenciaNotas(Disciplina disciplina, Aluno aluno, String acao, int av, double nota)
Material
-nome: String
-url: String
-descricao: String
-assunto: String
+ getNome(): String
+ setNome(String nome)
+ getUrl(): String
+ setUrl(String url)
+ getDescricao(): String
+ setDescricao(String descricao)
+ getAssunto(): String
+ setAssunto(String assunto)
Contato
- localidade: String
- telefone: String
+ getLocalidade(): String
+ setLocalidade(String nome)
+ getTelefone(): String
+ setTelefone(String url)
Curso
- disciplinas: List<Disciplina>
- duracao: Float
- nome: String
- professores: List<Professor>
+ getNomeCurso(): String
+ setNomeCurso(String nomeCurso)
+ getDuracao(): Float
+ setDuracao(Float duracao)
+ addProfessor(Professor professor)
+ addDisciplina(Disciplina disciplina)
+ getDisciplinas(): List<Disciplina>
Aluno
-ra: Integer
-nome: String
-email: String
-curso: Curso
-dataNasc: String
-telefone: String
-faltas: Integer
-notas:List<Float>
+ getRa(): String
+ setRa(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
+ cadastraDisciplinas()
+ adicionaFalta(int qtd)
+ corrigeFalta(int qtd)
+ quantidadeFaltas(): Integer
+ getDisciplinas(): List<Disciplina>
- notaExiste(int av): Boolean
+ adicionaNota(int av, double nota)
+ corrigeNota(int av, double nota)
+ verNota(int av): Double
Disciplina
-nome: String
-numAulas: Integer
-ementa: String
-materiais: Material
+ getNome(): String
+ setNome(String nome)
+ getEmenta(): String
+ setEmenta(String ementa)
+ getNumeroAulas(): Integer
+ setNumeroAulas(int numeroAulas)
+ addMaterial(Material material)
+ getAllMateriais(): List<Material>

Desenvolvimento




```java
public class Sacad {
  public List<Escola> escolas = new LinkedList<Escola>();

  public void cadastrarEscola(Escola novaEscola){
    escolas.add(novaEscola);
  }

  public void removerEscola(Escola novaEscola){
    escolas.remove(novaEscola);
  }

  public List<Escola> listarEscolas() {
      return this.escolas;
  }
}
```

Aluno

-ra: Integer
-nome: String
-email: String
-curso: Curso
-dataNasc: String
-telefone: String
-faltas: Integer
-notas:List<Float>

+ getRa(): String
+ setRa(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
+ cadastraDisciplinas()
+ adicionaFalta(int qtd)
+ corrigeFalta(int qtd)
+ quantidadeFaltas(): Integer
+ getDisciplinas(): List<Disciplina>
- notaExiste(int av): Boolean
+ adicionaNota(int av, double nota)
+ corrigeNota(int av, double nota)
+ verNota(int av): Double

```java
import java.util.LinkedList;

public class Aluno {
    private String nome, ra, email, dataNasc, nomeResp;
    private Curso cursoMatriculado;
    private int faltas = 0;
    private List<Disciplina> disciplinas = new LinkedList<Disciplina>();
    private List<Double> notas = new LinkedList<Double>();

    // CONSTRUCTOR
    public Aluno(String nome, String ra, String email, String dataNasc, String nomeResp, Curso curso) {
        this.nome = nome;
        this.ra = ra;
        this.email = email;
        this.dataNasc = dataNasc;
        this.nomeResp = nomeResp;
        this.cursoMatriculado = curso;
    }

    // GETTERS AND SETTERS
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getRa() {
        return ra;
    }

    public void setRa(String ra) {
        this.ra = ra;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

Aluno

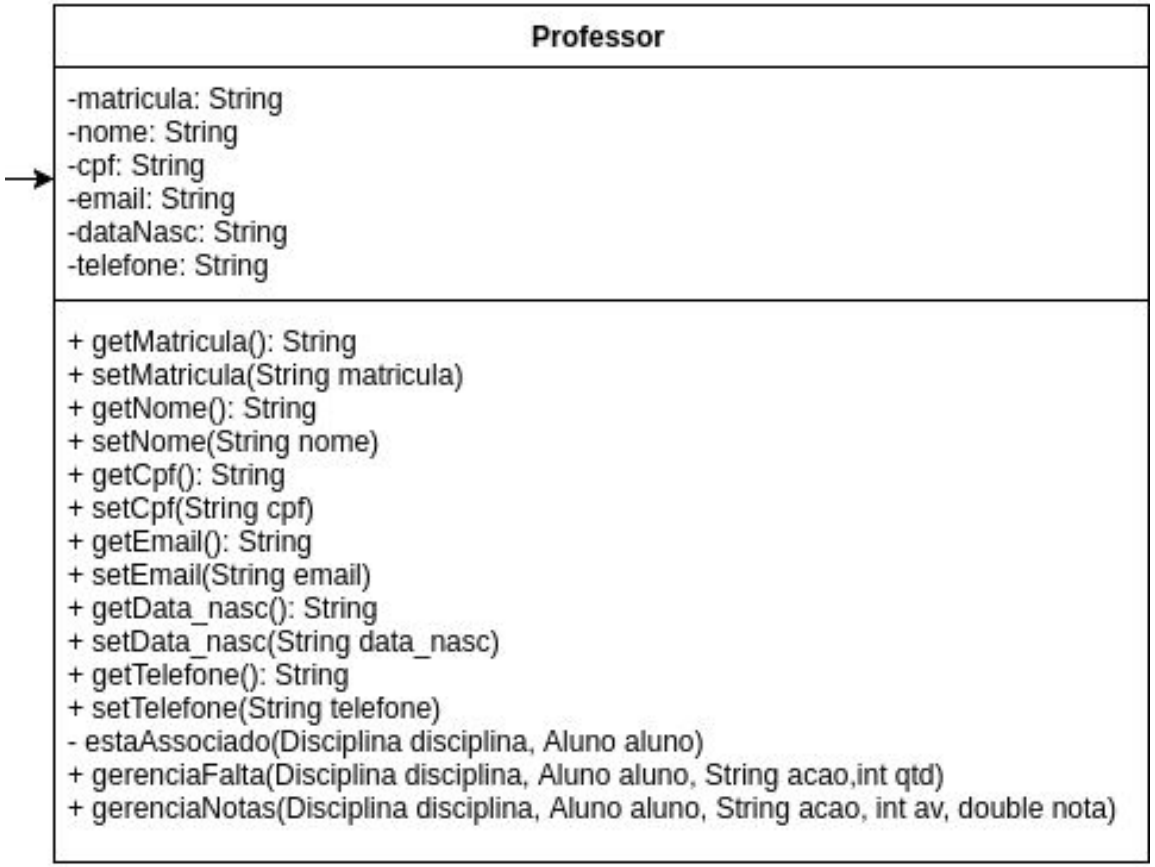
-ra: Integer
-nome: String
-email: String
-curso: Curso
-dataNasc: String
-telefone: String
-faltas: Integer
-notas:List<Float>

+ getRa(): String
+ setRa(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
+ cadastraDisciplinas()
+ adicionaFalta(int qtd)
+ corrigeFalta(int qtd)
+ quantidadeFaltas(): Integer
+ getDisciplinas(): List<Disciplina>
- notaExiste(int av): Boolean
+ adicionaNota(int av, double nota)
+ corrigeNota(int av, double nota)
+ verNota(int av): Double

```java
public String getDataNasc() {
    return dataNasc;
}

public void setDataNasc(String dataNasc) {
    this.dataNasc = dataNasc;
}

public String getNomeResp() {
    return nomeResp;
}

public void setNomeResp(String nomeResp) {
    this.nomeResp = nomeResp;
}

public Curso getCursoMatriculado() {
    return cursoMatriculado;
}

public void setCursoMatriculado(Curso curso) {
    this.cursoMatriculado = curso;
}

// METHODS
public void cadastraDisciplinas() {
    List<Disciplina> disciplinasDoCurso =  this.cursoMatriculado.getDisciplinas();
    for(Disciplina disciplina:disciplinasDoCurso) {
        this.disciplinas.add(disciplina);
    }
}

public void adicionaFalta(int qtd) {
    this.faltas += qtd;
}

public void corrigeFalta(int qtd) {
    this.faltas -= qtd;
}

public int quantidadeFaltas() {
    return this.faltas;
}
```

Aluno
-ra: Integer -nome: String -email: String -curso: Curso -dataNasc: String -telefone: String -faltas: Integer -notas:List<Float>
+ getRa(): String + setRa(String matricula) + getNome(): String + setNome(String nome) + getEmail(): String + setEmail(String email) + getData_nasc(): String + setData_nasc(String data_nasc) + getTelefone(): String + setTelefone(String telefone) + cadastraDisciplinas() + adicionaFalta(int qtd) + corrigeFalta(int qtd) + quantidadeFaltas(): Integer + getDisciplinas(): List<Disciplina> - notaExiste(int av): Boolean + adicionaNota(int av, double nota) + corrigeNota(int av, double nota) + verNota(int av): Double

```java
    public List<Disciplina> getDisciplinas() {
        return this.disciplinas;
    }

    private boolean notaExiste(int av) {
        if(this.notas.get(av-1) instanceof Double) {
            return true;
        }
        return false;
    }

    public void adicionaNota(int av, double nota) {
        this.notas.add(av-1, nota);
    }

    public void corrigeNota(int av, double nota) {
        if(this.notaExiste(av)) {
            adicionaNota(av, nota);
        }
    }

    public double verNota(int av) {
        if(this.notaExiste(av)) {
            return this.notas.get(av-1);
        }
        return -1.0;

    }
}
```




```java
import java.util.List;

public class Professor {

  private String matricula, nome, cpf, email, data_nasc, telefone;

  // CONSTRUCTOR
  public Professor(String matricula, String nome, String cpf, String email, String data_nasc, String telefone)
      this.matricula = matricula;
      this.nome = nome;
      this.cpf = cpf;
      this.email = email;
      this.data_nasc = data_nasc;
      this.telefone = telefone;
  }


    // GETTERS AND SETTERS
    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getEmail() {
        return email;
    }
```

Professor

-matricula: String
-nome: String
-cpf: String
-email: String
-dataNasc: String
-telefone: String

+ getMatricula(): String
+ setMatricula(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getCpf(): String
+ setCpf(String cpf)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
- estaAssociado(Disciplina disciplina, Aluno aluno)
+ gerenciaFalta(Disciplina disciplina, Aluno aluno, String acao,int qtd)
+ gerenciaNotas(Disciplina disciplina, Aluno aluno, String acao, int av, double nota)

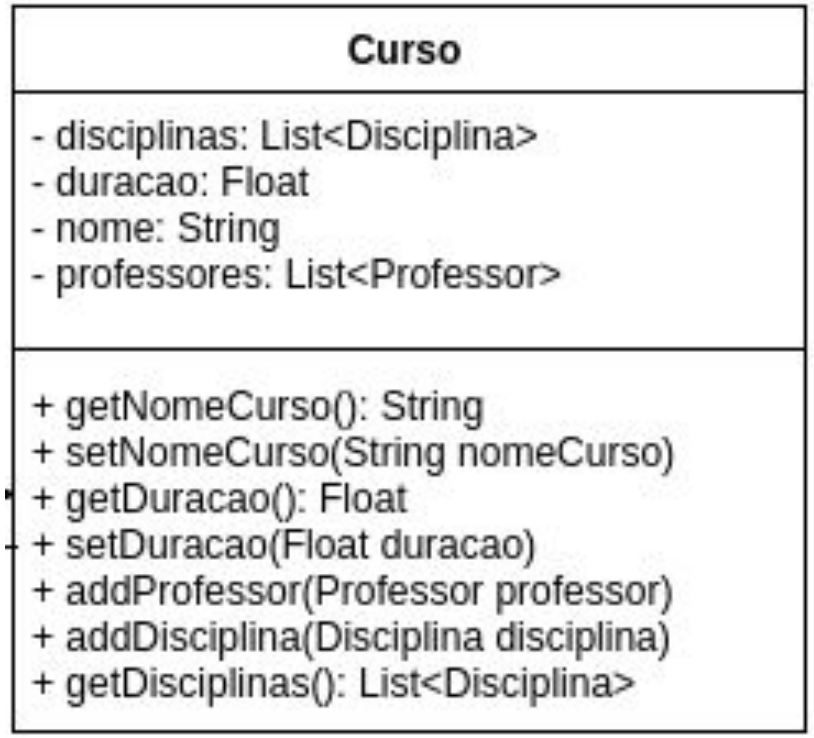
```java
public void setEmail(String email) {
    this.email = email;
}

public String getData_nasc() {
    return data_nasc;
}

public void setData_nasc(String data_nasc) {
    this.data_nasc = data_nasc;
}

public String getTelefone() {
    return telefone;
}

public void setTelefone(String telefone) {
    this.telefone = telefone;
}

// METHODS
private boolean estaAssociado(Disciplina disciplina, Aluno aluno) {
    List<Disciplina> disciplinasDoAluno = aluno.getDisciplinas();
    for (Disciplina disciplinaDoAluno:disciplinasDoAluno) {
        if(disciplina == disciplinaDoAluno) {
            return true;
        }
    }
    return false;
}

public void gerenciaFalta(Disciplina disciplina, Aluno aluno, String acao,int qtd) {
    if (this.estaAssociado(disciplina, aluno)) {
        if(acao.equals("adiciona")) {
            aluno.adicionaFalta(qtd);
        } else if (aluno.quantidadeFaltas() >= 0){
            aluno.corrigeFalta(qtd);
        }
    }
}

public void gerenciaNotas(Disciplina disciplina, Aluno aluno, String acao, int av, double nota) {
    if (this.estaAssociado(disciplina, aluno)) {
```

Professor

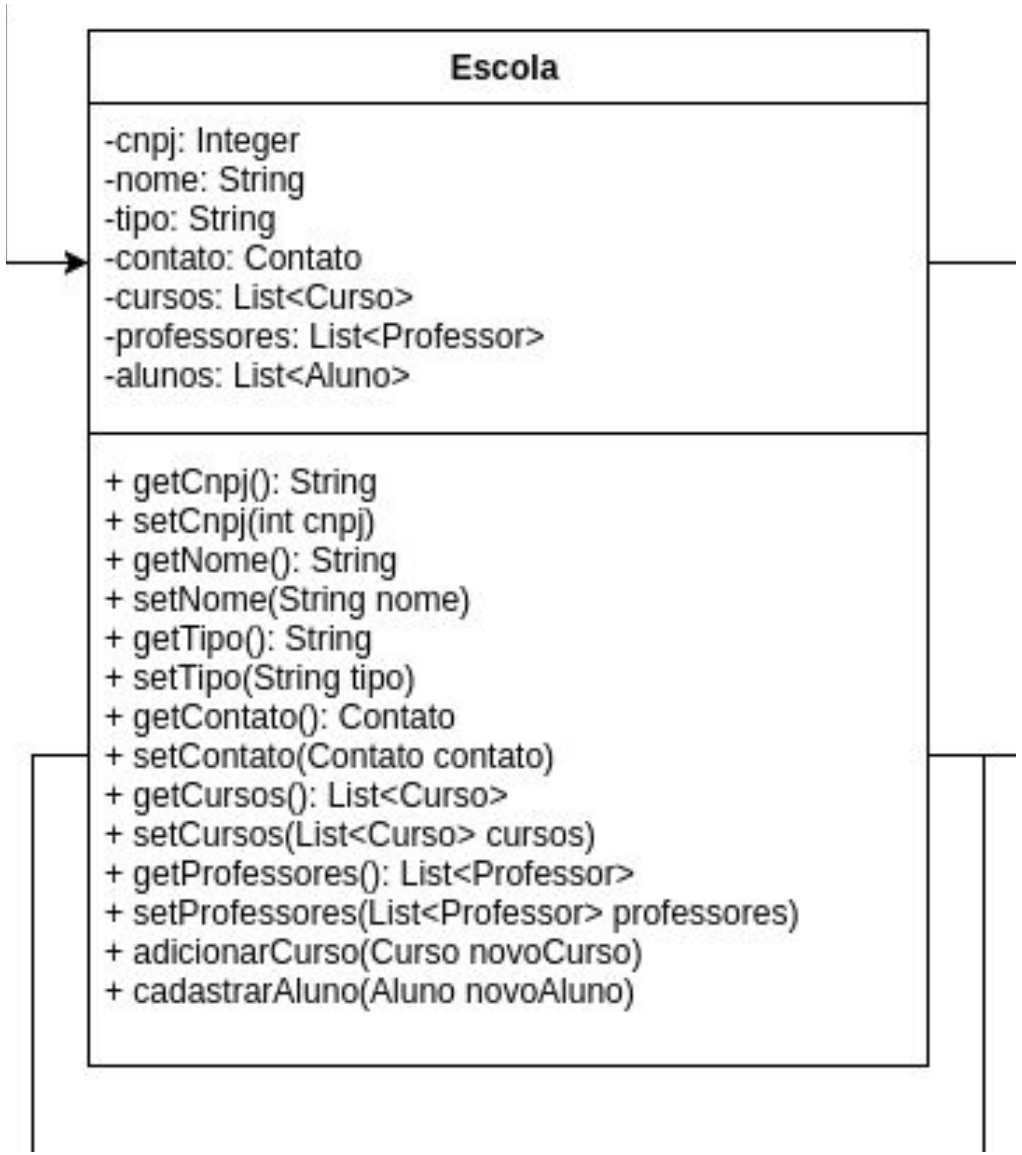
-matricula: String
-nome: String
-cpf: String
-email: String
-dataNasc: String
-telefone: String

+ getMatricula(): String
+ setMatricula(String matricula)
+ getNome(): String
+ setNome(String nome)
+ getCpf(): String
+ setCpf(String cpf)
+ getEmail(): String
+ setEmail(String email)
+ getData_nasc(): String
+ setData_nasc(String data_nasc)
+ getTelefone(): String
+ setTelefone(String telefone)
- estaAssociado(Disciplina disciplina, Aluno aluno)
+ gerenciaFalta(Disciplina disciplina, Aluno aluno, String acao,int qtd)
+ gerenciaNotas(Disciplina disciplina, Aluno aluno, String acao, int av, double nota)

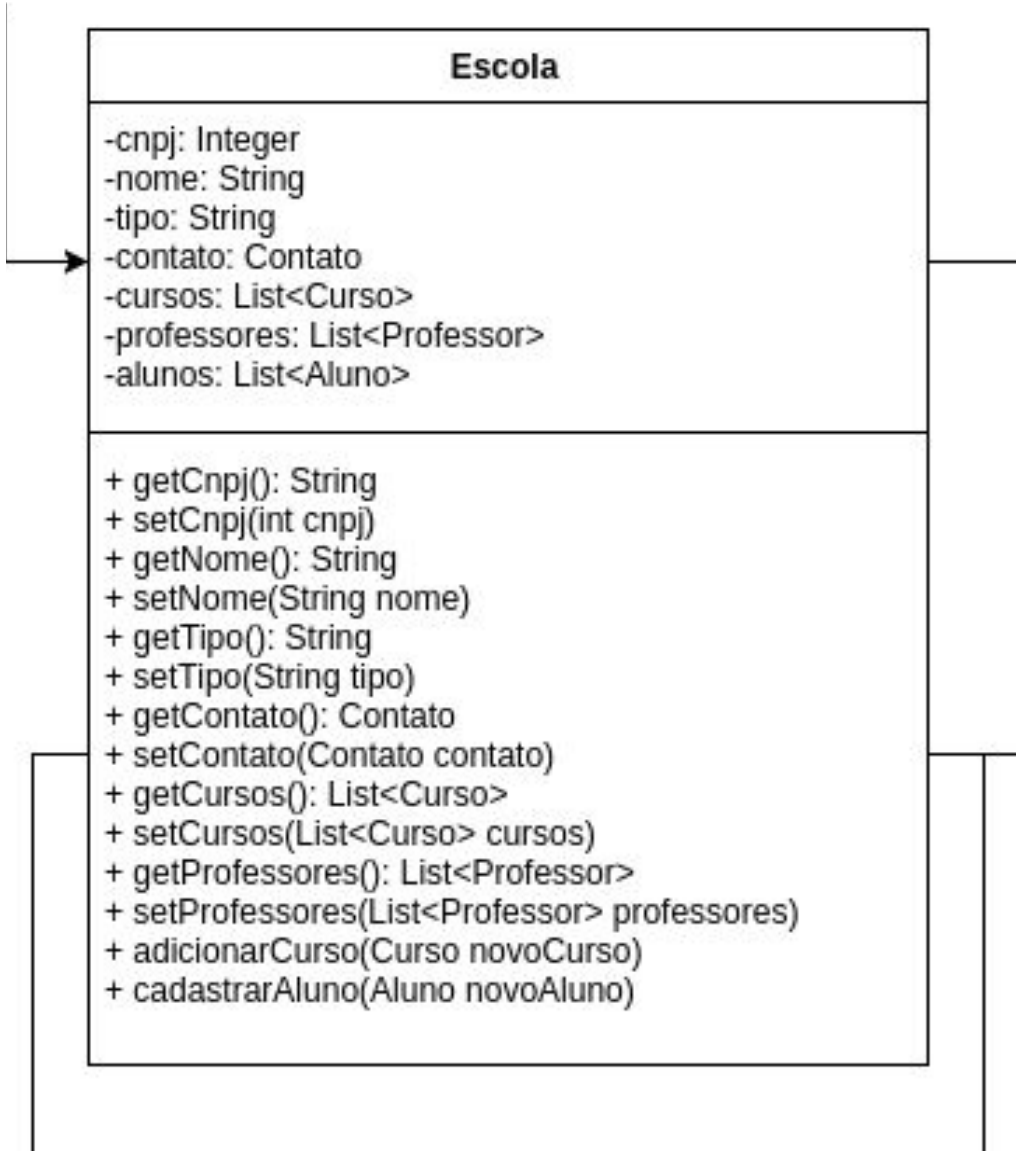
```java
    public void gerenciaNotas(Disciplina disciplina, Aluno aluno, String acao, int av, double nota) {
        if (this.estaAssociado(disciplina, aluno)) {
            if(acao.equals("adiciona")) {
                aluno.adicionaNota(av, nota);
            } else if (aluno.quantidadeFaltas() >= 0){
                aluno.corrigeNota(av, nota);
            }
        }
    }

}
```




```java
import java.util.LinkedList;
import java.util.List;

public class Curso {
    private List<Disciplina> disciplinas = new LinkedList<Disciplina>();
    private List<Professor> professores = new LinkedList<Professor>();
    private Float duracao;
    private String nomeCurso;

    // CONSTRUCTOR
    public Curso(String nome, Float duracao) {
        this.duracao = duracao;
        this.nomeCurso = nome;
    }

    //GETTERS AND SETTERS
    public String getNomeCurso() {
        return nomeCurso;
    }

    public void setNomeCurso(String nomeCurso) {
        this.nomeCurso = nomeCurso;
    }

    public Float getDuracao() {
        return duracao;
    }

    public void setDuracao(Float duracao) {
        this.duracao = duracao;
    }


    // METHODS
    public void addProfessor(Professor professor) {
        professores.add(professor);
    }
    public void addDisciplina(Disciplina disciplina) {
        disciplinas.add(disciplina);
    }

    public List<Disciplina> getDisciplinas() {
        return this.disciplinas;
    }
```




```java
import java.util.LinkedList;

public class Escola {
  private int cnpj;
  private String nome, tipo;
  private Contato contato;
  private List<Curso> cursos = new LinkedList<Curso>();
  private List<Professor> professores = new LinkedList<Professor>();
  private List<Aluno> alunos = new LinkedList<Aluno>();

  // Cons

  public Escola(int cnpj, String nome, String tipo, Contato contato) {
    this.cnpj = cnpj;
    this.nome = nome;
    this.tipo = tipo;
    this.contato = contato;
  }

  // getters and setters

    public int getCnpj() {
        return cnpj;
    }

    public void setCnpj(int cnpj) {
        this.cnpj = cnpj;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public Contato getContato() {
```




```java
    public Contato getContato() {
        return contato;
    }

    public void setContato(Contato contato) {
        this.contato = contato;
    }

    public List<Curso> getCursos() {
        return cursos;
    }

    public void setCursos(List<Curso> cursos) {
        this.cursos = cursos;
    }

    public List<Professor> getProfessores() {
        return professores;
    }

    public void setProfessores(List<Professor> professores) {
        this.professores = professores;
    }

// Methods

      public void adicionarCurso(Curso novoCurso) {
        cursos.add(novoCurso);
      }

      public void cadastrarAluno(Aluno novoAluno) {
          alunos.add(novoAluno);
      }
}
```

Testes

```
import static org.junit.jupiter.api.Assertions.*;

import java.util.List;

import org.junit.jupiter.api.Test;

class JSacad {

	@Test
	void test() {

		Sacad plataforma = new Sacad();

		Contato contatoEscola = new Contato("Sao Jose dos Campos", "1225646845");

		Escola novaEscola = new Escola(4151654, "ETEP", "Tecnico", contatoEscola);
		plataforma.cadastrarEscola(novaEscola);

		Professor professor = new Professor("026a15", "Paulo Romeiro", "426879512578", "paulinho02@yahoo.com", "02/03/1987",
"12458497612");

		Disciplina informaticaBasica = new Disciplina("Informatica Basica", "Textyo longo", 25);
```

```
Curso informatica = new Curso("Informatica", 2.5f);
        informatica.addDisciplina(informaticaBasica);
        informatica.addProfessor(professor);
        Curso eletronica = new Curso("Eletronica", 2.5f);
        eletronica.addDisciplina(informaticaBasica);
        eletronica.addProfessor(professor);
        novaEscola.adicionarCurso(informatica);
        novaEscola.adicionarCurso(eletronica);
        Material materialGenerico = new Material("Telecurso2000", "http://telecurso.com/materiallegal", "UMa aprofundada na materia de segunda
falando sobre os fundamentos do excel", "Excel");
        informaticaBasica.addMaterial(materialGenerico);
Aluno novoAluno = new Aluno("Joao Pedro" ,"12521581s", "joaozin@pao.com", "02/03/2005", "Maria Angela da Silva", informatica);
        novoAluno.cadastraDisciplinas();
        novaEscola.cadastrarAluno(novoAluno);
        professor.gerenciaFalta(informaticaBasica, novoAluno, "adiciona", 3);
        System.out.println(novoAluno.quantidadeFaltas());
        professor.gerenciaFalta(informaticaBasica, novoAluno, "corrige", 1);
        System.out.println(novoAluno.quantidadeFaltas());
```

```
        professor.gerenciaNotas(informaticaBasica, novoAluno, "adiciona", 1, 7.5);
        System.out.println(novoAluno.verNota(1));
        professor.gerenciaNotas(informaticaBasica, novoAluno, "corrige", 1, 9.5);
        System.out.println(novoAluno.verNota(1));
    }

}
```

eclipse-workspace - Aula/src/JSacad.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Explorer | JUnit

Finished after 0,13 seconds

Runs: 1/1 Errors: 0 Failures: 0

JSacad [Runner: JUnit 5] (0,002 s)

Failure Trace

JSacad.java

```java
import static org.junit.jupiter.api.Assertions.*;

class JSacad {

    @Test
    void test() {

        Sacad plataforma = new Sacad();

        Contato contatoEscola = new Contato("Sao Jose dos Campos", "1225646845");

        Escola novaEscola = new Escola(4151654, "ETEP", "Tecnico", contatoEscola);
        plataforma.cadastrarEscola(novaEscola);

        Professor professor = new Professor("026a15", "Paulo Romeiro", "426879512578", "paulinho02@yahoo.com", "02/03/1987", "124

        Disciplina informaticaBasica = new Disciplina("Informatica Basica", "Textyo longo", 25);

        Curso informatica = new Curso("Informatica", 2.5f);
        informatica.addDisciplina(informaticaBasica);
        informatica.addProfessor(professor);

        Curso eletronica = new Curso("Eletronica", 2.5f);
        eletronica.addDisciplina(informaticaBasica);
        eletronica.addProfessor(professor);

        novaEscola.adicionarCurso(informatica);
        novaEscola.adicionarCurso(eletronica);

        Material materialGenerico = new Material("Telecurso2000", "http://telecurso.com/materiallegal", "UMa aprofundada na mater
        informaticaBasica.addMaterial(materialGenerico);

        Aluno novoAluno = new Aluno("Joao Pedro" ,"12521581s", "joaozin@pao.com", "02/03/2005", "Maria Angela da Silva", informat
        novoAluno.cadastraDisciplinas();
```

Problems | Javadoc | Declaration | Console | Diagrams

<terminated> JSacad [JUnit] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (19 de nov de 2018 18:35:25)

```
3
2
7.5
9.5
Picked up _JAVA_OPTIONS: -Xms256m -Xmx512m
```