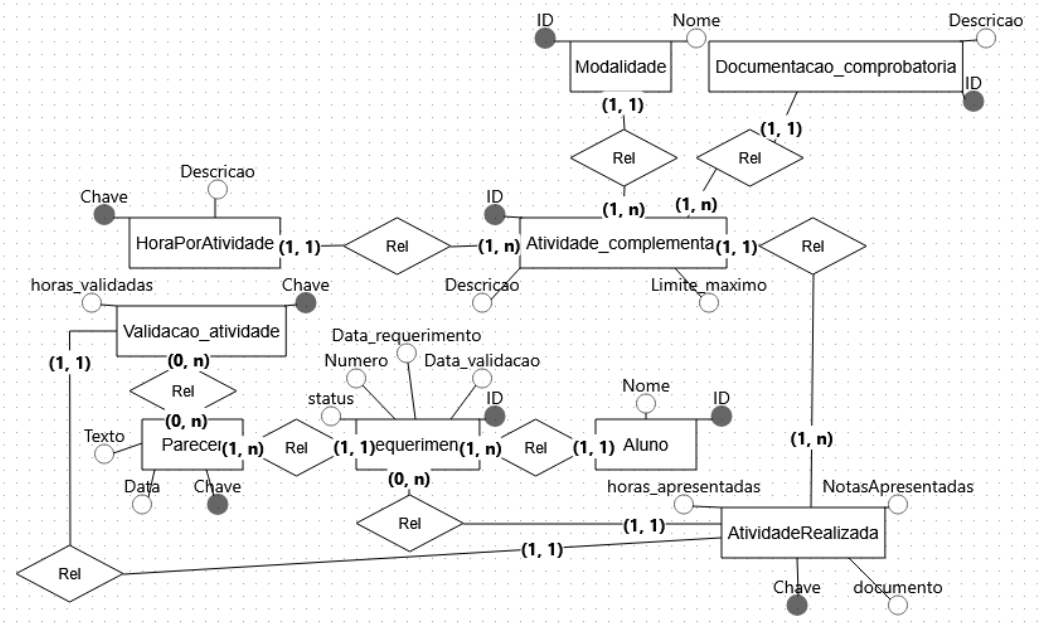
Contexto: O Instituto Federal de Santa Catarina – Câmpus Tubarão adota um regulamento para atividades complementares que classifica as atividades em quatro modalidades (Ensino, Extensão, Pesquisa e Inovação, Complementação), cada qual com seus requisitos de documentação, limites de horas e critérios específicos de aproveitamento.

É necessário desenvolver um programa em Java que realize a avaliação automatizada de requerimentos de validação de atividades complementares, conforme as regras definidas na resolução, onde estão especificadas as modalidades, a documentação comprobatória exigida, os limites máximos de horas e demais critérios de validação.

Este projeto utiliza Java & PostgreSQL, é necessário possuir ambos instalados e o banco de dados configurado conforme o script presente no arquivo `Dao/AtividadesComplementares.sql`. Execute a aplicação através da classe Main.

Banco De Dados --PostgreSQL

O Banco de Dados é modelado conforme o Modelo Conceitual e implementado conforme o formulário da resolução anexados a seguir.



Modalidade	Atividade	Documentação Comprobatória	Horas por atividade	Limite Máximo	Horas Apresentadas	Horas Validades
Ensino	Disciplinas cursadas com aproveitamento, não previstas no currículo do curso.	Certificado ou declaração de conclusão	Conforme documento comprobatório*	80		
	Semana acadêmica dos cursos, quando não obrigatória.	Certificado ou declaração de participação	Conforme documento comprobatório*	40		
	Participação em atividades de monitoria.	Declaração de participação	Conforme documento comprobatório*	70		
	Atividades realizadas em laboratórios e/ou oficinas do Instituto.	Declaração de participação	4 horas por atividade	40		
	Visita Técnica relacionada à área.	Declaração de participação	4 horas por atividade	40		
	Participação em cursos de qualificação na área afim do curso com certificado de aproveitamento.	Certificado de participação	Conforme documento comprobatório*	80		
	Participação como ouvintes em bancas de projetos integradores de assuntos relacionados à área	Documento assinado pelo presidente da banca ou coordenador de curso	1 hora por atividade.	20		
	Participação como ouvintes em bancas de TCC, dissertações ou teses de assuntos relacionados à área	Documento assinado pelo presidente da banca ou coordenador de curso	2 horas por atividade	20		
	Desenvolvimento de material didático ou instrucional	Declaração da instituição promotora	3 horas por atividade.	30		
	Instrutor de cursos abertos à comunidade	Declaração da instituição promotora	Conforme documento comprobatório*	70		
Extensão	Participação em programa ou projeto de extensão.	Declaração de participação	Conforme documento comprobatório*	60		
	Apresentação de projeto de extensão.	Declaração de apresentação	4 horas por atividade	20		
	Participação em ações sociais cívicas e comunitárias.	Declaração de participação	4 horas por atividade	40		
	Texto em jornal ou revista da área.	Texto.	4 horas por atividade.	40		
	Intercâmbio com instituições de ensino no Brasil ou no exterior.	Declaração de participação.	50 horas por mês	100		
	Estágio não-obrigatório na área do curso, formalizado pelo IFSC.	Declaração do empregador	25 horas por mês	100		
	Exercício profissional com vínculo empregatício, desde que na área do curso.	Declaração do empregador ou Carteira Profissional	50 horas por mês	100		
Pesquisa e Inovação	Participação em programa ou projeto de pesquisa relacionados a área.	Declaração de participação	Conforme documento comprobatório*	60		
	Apresentação de projeto de pesquisa relacionado à área.	Declaração de apresentação	4 horas por atividade	20		
	Autoria e coautoria em artigo publicado em Periódico na área afim.	Capa do artigo ou aceite	10 horas por item	30		
	Livro na área afim.	Cópia da ficha catalográfica	50 por item	100		
	Capítulo de livro na área afim.	Cópia da ficha catalográfica e capa do capítulo	10 horas por item	30		
	Publicação em Anais de Evento Técnico Científico.	Cópia do artigo	5 horas por atividade	30		
	Apresentação de trabalho em Evento Técnico Científico.	Certificado ou declaração de participação	4 horas por atividade	20		
	Participação de Programa de Iniciação Científica.	Declaração de participação	Conforme documento comprobatório*	60		
	Participação como palestrante, conferencista, integrante de mesa-redonda, ministrante de mini-curso em evento científico.	Certificado ou declaração de participação	15 horas por atividade	60		
	Prêmios concedidos por instituições acadêmicas, científicas e profissionais.	Declaração de premiação.	15 horas por atividade	60		
	Participação na criação de Produto ou Processo Tecnológico com propriedade intelectual registrada.	Registro da propriedade intelectual.	100 horas por atividade	200		
	Participação em grupo de pesquisa na área.	Declaração do líder do grupo.	4 horas por mês de participação	60		
Complementação	Participação em congressos, jornadas, simpósios, fóruns, seminários, encontros, palestras, festivais e similares, com certificado de aproveitamento e/ou frequência.	Declaração de participação	4 horas por atividade	20		
	Comissão organizadora de congressos, jornadas, simpósios, fóruns, seminários, encontros, palestras, festivais e similares.	Declaração de instituição promotora.	10 horas por atividade	20		
	Premiação em eventos que tenha relação com os objetos de estudo do curso.	Declaração do participação	10 horas por atividade	30		
	Curso de língua estrangeira.	Certificado ou declaração de participação	Conforme documento comprobatório*	80		
	Premiação em atividades esportivas como representante do Instituto.	Cópia da declaração.	15 horas por atividade	60		
	Representação estudantil em colegiado, grêmio estudantil, centro acadêmico, comissão de formatura, associação esportiva e afins.	Declaração da instituição	4 horas por mês	40		
	Representação de turma (inclui a participação em conselhos de classe).	Declaração do Coordenador do Curso.	10 horas por semestre inteiro.	30		
	Participação em Empresa Júnior.	Declaração do dirigente da empresa.	4 horas por mês	40		
	Classificação em concursos culturais	Certificado, declaração e/ou endereço eletrônico do resultado classificatório	10 horas por item	20		
	Participação em projetos sociais, trabalho voluntário em entidades vinculadas a compromissos sociopolíticos	Declaração assinada pelo responsável.	Conforme documento comprobatório*	20		
	Desenvolvimento de atividades socioculturais, artísticas e esportivas (coral, música, dança, bandas, vídeos, cinema, cineclubes, teatro, campeonatos esportivos etc)	Declaração assinada pelo responsável	Conforme documento comprobatório*	20		

Portanto, possui as tabelas:

aluno(id, nome);

modalidade(id, nome);

documentacao_comprobatoria(id, descricao);

horas_por_atividade(id, descricao);

atividade_complementar(id, descricao);

requerimento(id, aluno_id, data_requerimento, status, data_validacao);

parecer(id, requerimento_id, texto, data_parecer);

atividade_realizada(id, requerimento_id, atividade_id, horas_apresentadas, documento);

Os dados inseridos (Horas apresentadas), fazem necessária a existência do aluno para a criação de um requerimento com seu ID e um parecer com o ID do requerimento recém criado.

Quando as horas são apresentadas para uma atividade_complementar os dados são inseridos em uma nova atividade_realizada (requerimento_id, atividade_id, horas_apresentadas e documento) no banco de dados;

Uma validacao_atividade é criada recebendo o id da atividade_realizada, as horas_apresentadas(transformadas em horas_validadas através do código validarHoras) e o id do parecer.

Por fim, quando todas as horas são apresentadas pelo usuário e validadas automaticamente, o parecer é emitido, da seguinte forma:

=== PARECER DE VALIDAÇÃO ===

Aluno: ____, Matrícula: ____

Data emissão: ____

Atividade _:

Descrição: _____

Horas declaradas: ___h

Limite Máximo: ___h

Horas validadas: ___h

Observação: Horas declaradas (__h) excedem o limite (__h); ajustadas para __h.

Resumo geral:

Total de horas declaradas: __h

Total de horas validadas: __h

== Menu encerrado. ==

Interface do usuário em terminal -Java

A aplicação é inicializada através da classe Main:

```
public class Main {
    public static void main(String[] args) {
        // Cria um aluno e exibe o menu de modalidades
        // O aluno é criado através do AlunoDao e o menu é gerado pela
classe menuModalidades
        // O método exibir() do Objeto Menu retornado por menuModalidades
é chamado para mostrar o menu
        new GeraMenu().menuModalidades(new
AlunoDao().criarAluno()).exibir();
    }
}
```

O método menuModalidades da classe GerarMenu recebe um Aluno e retorna um Menu, que é imediatamente exibido na classe Main através do método exibir() da classe Menu.

O aluno recebido é primeiro instanciado através da classe AlunoDao, que o busca no banco de dados através do id fornecido:

AlunoDao.criarAluno()

Solicita ao usuário que insira o ID do aluno;

Chama o método AlunoDao.consultarAlunoPorId() para verificar se este aluno existe;

Se o aluno for encontrado

```
Digite o ID do aluno:
27
Aluno encontrado: murilo, Matrícula: 948226759
```

Exibe suas informações e retorna objeto Aluno;

Se não for encontrado

```
Digite o ID do aluno:
165
Aluno não encontrado com o id: 165
Aluno não encontrado com este id, deseja criar um novo aluno? (S/N)
s
Digite o nome do novo aluno:
murilo
Aluno salvo com sucesso:
 Id:27
 Nome:murilo,
 Matrícula: 341126824
```

Permite o usuário criar o Aluno;

Se optar por criar

Cria um objeto Aluno e chama o método AlunoDao.salvar() para armazená-lo no banco de dados.

Se optar por não criar

A operação é cancelada.

O Aluno retornado pela classe então é fornecido à

GerarMenu.menuModalidades(Aluno):

Um novo objeto Menu é criado, com o título "== Modalidades ==" e uma lista de itens de menu.

```
Menu menuModalidades = new Menu("== Modalidades ==", new
ArrayList<ItemMenu>() {{
final int[] c = {1};// Contador de Modalidades
```

Um requerimento é criado para o aluno, e um parecer é gerado para esse requerimento.

```
Requerimento requerimento = new
RequerimentoDao().insertRequerimento(aluno);
        new ParecerDao().insertParecer(requerimento);
```

Uma lista com as modalidades presentes no Banco de Dados é criada.

```
== Modalidades ==
1) Ensino
2) Extensão
3) Pesquisa e Inovação
4) Complementação
0) Finalizar e emitir parecer
Escolha uma opção:
3
```

```
List<Modalidade> modalidades = new
ModalidadeDao().consultarModalidades();
```

Um foreach cria uma OpcaoComSubmenu para cada Modalidade encontrada.

```
for (Modalidade modalidade : modalidades) {
        add(new OpcaoComSubmenu(c[0], modalidade.nome(), new
        Menu(modalidade.nome(), new ArrayList<ItemMenu>() {{
```

O Submenu é um novo Menu dentro do ArrayList<ItemMenu>(), com um novo ArrayList<ItemMenu>() próprio, contendo cada **OpcaoMenuAtividadeComplementar(numero, AtividadeComplementar, Requerimento).**

O Submenu vem de um novo foreach das AtividadesComplementares encontradas para a modalidade atual, obtidas através do método **AtividadeComplementarDao.consultarPorModalidade(idModalidade);:**

```
Pesquisa e Inovação
1) Participação em programa ou projeto de pesquisa relacionados à área. (limite: 60h)
2) Apresentação de projeto de pesquisa relacionado à área. (limite: 20h)
3) Autoria e coautoria em artigo publicado em Periódico na área afim. (limite: 30h)
4) Livro na área afim. (limite: 100h)
5) Capítulo de livro na área afim. (limite: 30h)
6) Publicação em Anais de Evento Técnico Científico. (limite: 30h)
7) Apresentação de trabalho em Evento Técnico Científico. (limite: 20h)
8) Participação de Programa de Iniciação Científica. (limite: 60h)
9) Participação como palestrante, conferencista, integrante de mesa-redonda, ministrante de mini
-curso em evento científico. (limite: 60h)
10) Prêmios concedidos por instituições acadêmicas, científicas e profissionais. (limite: 60h)
11) Participação na criação de produto, processo ou propriedade intelectual. (limite: 200h)
12) Participação em grupo de pesquisa certificado. (limite: 60h)
0) Voltar
Escolha uma opção:
10
```

```
List<AtividadeComplementar> atividadesComplementares = new
AtividadeComplementarDao().consultarPorModalidade(c[0]);
int cc = 1;//Contador de atividades
for (AtividadeComplementar atividade : atividadesComplementares) {
      add(new OpcaoMenuAtividadesComplementares(cc, atividade,
requerimento));
      cc++;
      }
```

Cada **OpcaoMenuAtividadesComplementares** recebe a atividade e o requerimento quando selecionada, permanece a atividade realizada com as horas declaradas; Busca o parecer recém criado referente ao requerimento recebido e Insere a atividade validada, através do método ValidacaoAtividadeDao.insertValidacaoAtividade(), que primeiro valida as horas apresentadas e depois permanece as horas validadas.

```
Horas declaradas para Prêmios concedidos por instituições acadêmicas, científicas e profissionais. (limite: 60h):
65
Atividade adicionada ao requerimento.
```

```
System.out.println("Horas declaradas para "+titulo+":");
int horas_declaradas = input.nextInt();

AtividadeRealizada atividadeRealizada = new
AtividadeRealizadaDao().insertAtividadeRealizada(requerimento,
atividadeComplementar, horas_declaradas);
Parecer parecer = new
ParecerDao().ultimoParecerDoRequerimento(requerimento);
new ValidacaoAtividadeDao().insertValidacaoAtividade(atividadeRealizada,
atividadeComplementar, parecer, horas_declaradas);
```

A última opção do Submenu da Modalidade é uma OpcaoVoltar, filtrada automaticamente pela classe **Menu**

```
add(new OpcaoVoltar(0, "Voltar"));
```

```
if (item.numero() == opcao) {
      item.exibir();
      return !(item instanceof OpcaoVoltar);
}
```

Por fim, é adicionada uma OpcaoFinal(Filha de OpcaoVoltar) para o menu de Modalidades, que recebe o aluno e exibe o Parecer.

```
      }})));
        c[0]++; // Incrementa o contador de modalidades
        }
        add(new OpcaoFinal(0, "Finalizar e emitir parecer", aluno));
}});
        return menuModalidades;
```

O Parecer é então exibido através do método

```
System.out.println("=== PARECER DE VALIDAÇÃO ===");
new OpcaoFinalDao().emitirUltimoParecer(this.aluno);
System.out.println("== Menu encerrado. ==");
```

OpcaoFinalDao.emitirUltimoParecer(Aluno):

```
=== PARECER DE VALIDAÇÃO ===
Aluno: Carlos Eduardo Silva, Matrícula: 939655170
Data emissão: 2025-07-14
Atividade 1:
  Descrição:        Instrutor de cursos abertos à comunidade.
  Horas declaradas: 451h
  Limite Máximo:    70h
  Horas validadas:  70h
  Observação:       Horas declaradas (451h) excedem o limite (70h); ajustadas para 70h.

Atividade 2:
  Descrição:        Instrutor de cursos abertos à comunidade.
  Horas declaradas: 10h
  Limite Máximo:    70h
  Horas validadas:  10h
  Observação:       --(sem ajuste)

Resumo geral:
  Total de horas declaradas: 461h
  Total de horas validadas:  80h
== Menu encerrado. ==
```

Busca o Requerimento recém inserido e o ultimo parecer deste requerimento

```
Requerimento ultimoRequerimento = new
RequerimentoDao().ultimoRequerimentoDoAluno(aluno.id());
Parecer ultimoParecer = new
ParecerDao().ultimoParecerDoRequerimento(ultimoRequerimento);
```

Cria uma lista com as atividades_realizadas do requerimento.

```
List<AtividadeRealizada> atividadesRealizadas = new
AtividadeRealizadaDao().consultarAtividadesRealizadasPorRequerimento(ulti
moRequerimento);
```

Um foreach passa por cada atividade_realizada encontrada e exibe suas informações, junto da atividade_complementar e validacao_atividade equivalentes e conta as horas_declaradas e horas_validadas. Por fim exibe um resumo com o total de horas.

```
int c = 1, totHorasDeclaradas = 0, totHorasValidadas = 0;
for (AtividadeRealizada atividadeRealizada : atividadesRealizadas){
    System.out.println("Atividade " + c + ":");
    AtividadeComplementar atividadeComplementar = new
AtividadeComplementarDao().consultarPorId(atividadeRealizada.atividade_id
());
    System.out.println("  Descrição:        " +
atividadeComplementar.descricao());
    System.out.println("  Horas declaradas: " +
atividadeRealizada.horas_apresentadas() + "h");
    System.out.println("  Limite Máximo:    " +
atividadeComplementar.limite_horas() + "h");
    ValidacaoAtividade validacaoAtividade = new
ValidacaoAtividadeDao().consultarValidacaoPorAtividadeRealizada(atividade
Realizada);
    System.out.println("  Horas validadas:  " +
validacaoAtividade.horas_validadas() + "h");
    System.out.println(gerarObservacao(atividadeRealizada.horas_apresenta
das(), atividadeComplementar.limite_horas()));
    System.out.println("");
    totHorasDeclaradas += atividadeRealizada.horas_apresentadas();
    totHorasValidadas += validacaoAtividade.horas_validadas();
    c++;
}
System.out.println("Resumo geral:");

System.out.println("  Total de horas declaradas: "+ totHorasDeclaradas +
"h");
System.out.println("  Total de horas validadas:  "+ totHorasValidadas +
"h");
```

Por fim, o Menu retornado é exibido através do método

Menu.exibir():

Lista as escolhas e as testa através do método Menu.escolha(List<ItemMenu>)

```
boolean continuar = true;
```

```
        do{
            System.out.println(titulo);
            for (ItemMenu item : itensDeMenu) {
                System.out.println(item.numero() + ") " + item.titulo());
            }
            continuar = Menu.escolha(this.itensDeMenu);
        }while(continuar);
```

Menu.escolha(List<ItemMenu>):

Exibe o **ItemMenu** (OpcaoComSubmenu/OpcaoMenuAtividadesComplementares/OpcaoVoltar/OpcaoFinal)

Retorna true para ‘continuar’ e continua listando caso o usuário não opte por Voltar.

```
System.out.println("Escolha uma opção: ");

        try{
            int opcao = input.nextInt();
            for (ItemMenu item : itensDeMenu) {
                if (item.numero() == opcao) {
                    item.exibir();
                    return !(item instanceof OpcaoVoltar);
                }
            }
            return true;
        }catch(Exception e) {
            System.out.println("Opção inválida. Tente novamente.");
            input.nextLine();
            return true;
        }
```

Padrões de Projeto utilizados:

Data Access Object (DAO)

Cada entidade persistida no banco (Aluno, Requerimento, Parecer, atividade_complementar, etc.) tem sua própria classe **Dao** responsável por encapsular toda a lógica de acesso a dados (inserir, consultar e atualizar).

Isola o código SQL do restante da aplicação, facilitando manutenção e testes.

Composite

A classe Menu e as classes que implementam ItemMenu formam uma estrutura em árvore: Menu mantém uma lista de ItemMenu (pode conter submenus ou opções-folhas), OpcaoComSubmenu carrega internamente outro Menu.

Permite tratar de forma uniforme tanto itens de menu (folhas) quanto submenus.

Command

Cada opção de menu (OpcaoMenuAtividadesComplementares, OpcaoVoltar, OpcaoFinal etc.) encapsula, em seu método exibir(), todo o comportamento que deve ocorrer quando o usuário a seleciona.

O menu invoca item.exibir() sem conhecer os detalhes de implementação de cada ação.

Factory Method (Geração de Menus)

A classe GeraMenu expõe o método que atua como "fábrica" de objetos Menu, configurados a partir dos dados no banco (Modalidades e AtividadesComplementares).

Centraliza a lógica de construção da estrutura de menus, isolando-a do Main.

Template Method

O próprio Menu.exibir() define o "esqueleto" do fluxo de interatividade:

- Mostrar título e lista de opções
- Ler escolha do usuário
- Delegar a execução chamando item.exibir()
- Repetir até o comando de sair

A variação (o que cada opção faz) fica nas subclasses de ItemMenu.