
UNIVERSIDADE DE PERNAMBUCO (UPE)

ESCOLA POLITÉCNICA DE PERNAMBUCO (POLI)

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

BRENO MEDEIROS DE OLIVEIRA

# REELABORANDO O SISTEMA DE PÓS-GRADUAÇÃO E PESQUISA (SISPG) DA UPE

Recife, Junho de 2019


UNIVERSIDADE DE PERNAMBUCO (UPE)

ESCOLA POLITÉCNICA DE PERNAMBUCO (POLI)

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

BRENO MEDEIROS DE OLIVEIRA

# REELABORANDO O SISTEMA DE PÓS-GRADUAÇÃO E PESQUISA (SISPG) DA UPE

Orientador: Prof. Dr. Joabe Bezerra Jesus Júnior

Artigo apresentado à Universidade de Pernambuco como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação

Recife, Junho de 2019

## MONOGRAFIA DE FINAL DE CURSO

**Avaliação Final (para o presidente da banca)***

No dia 02/07/2019, às 10h, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **BRENO MEDEIROS DE OLIVEIRA**, orientado(a) pelo(a) professor(a) **JOABE BEZERRA DE JESUS JÚNIOR**, sob título REELABORANDO O SISTEMA DE PÓS-GRADUAÇÃO E PESQUISA (SISPG) DA UPE, a banca composta pelos professores:

**BYRON LEITE DANTAS BEZERRA (PRESIDENTE)**
**JOABE BEZERRA DE JESUS JÚNIOR (ORIENTADOR)**

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

☐ Aprovada ☒ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 7,0 ( sete )

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

O(A) discente terá 25 dias para entrega da versão final da monografia a contar da data deste documento.

AVALIADOR 1: Prof (a) **BYRON LEITE DANTAS BEZERRA**

AVALIADOR 2: Prof (a) **JOABE BEZERRA DE JESUS JÚNIOR**

AVALIADOR 3: Prof (a)

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

## Autorização de publicação de PFC

Eu, **Breno Medeiros de Oliveira** autor(a) do projeto de final de curso intitulado: **REELABORANDO O SISTEMA DE PÓS-GRADUAÇÃO E PESQUISA (SISPG) DA UPE;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

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

Breno Medeiros de Oliveira
**Breno Medeiros de Oliveira**

Orientador(a): **Joabe Bezerra de Jesus Júnior**

Coorientador(a):

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 02/07/2019

# Reelaborando o Sistema de Pós-Graduação e Pesquisa (SISPG) da UPE

*Manuscript template for theREPA Journal*

**Breno Medeiros**[1,2] orcid.org/0000-0003-2270-3076

**JoabeJunior**[1,3] orcid.org/0000-0002-1518-0572

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,
[2] Graduação em Engenharia daComputação, Escola Politécnica de Pernambuco, Pernambuco, Brasil,
[3]Centro de Informática, Universidade Federal de Pernambuco, Pernambuco, Brasil.

**E-mail do autor principal: Breno Medeiros**breno.medeiros@icloud.com

**Resumo**

Este Artigo visa o estudo de caso da arquitetura atual do Sistema de Pós-Graduação e Pesquisa (Sispg) da Universidade de Pernambuco (UPE), bem como a elaboração de uma nova arquitetura para ela. E para isso, utilizaremos da Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) **[5]** para gerar os Diagramas de Casos de Uso, bem como os Diagramas de Classe de Análise e de Projeto **[6]** e **[7]**. Como resultado, a apresentação desses modelos permitirá a implementação(e reelaboração) de um novo Sispg, utilizando .NET Entity Framework Core e SQL Server, de forma a unificar as tecnologias utilizadas desse sistema com o de outros sistemas da UPE. Por fim, este trabalho tem como contribuição principal fazer parte de um conjunto de outros trabalhos/artigos que estão sendo elaborados neste semestre, e nos subsequentes, para a criação de novos sistemas(e reformulação dos antigos) de forma a preservar a consistência de dados no banco, evitar e tratar possíveis erros de execução/regras, e falta de responsividade nas páginas **[4]**.

**Palavras-Chave:**Sispg; UML; Reelaboração; Diagramas.

***Abstract***

*This article aims at the case study of the current architecture of the Graduate System and Research (Sispg) of the University of Pernambuco (UPE), as well as the elaboration of a new architecture for it. And for that, we will use the Unified Modeling Language (UML) [5] to generate the Use-Case Diagrams as well as the Analysis and Design Class Diagrams [6] and [7]. As a result, the presentation of these models will allow the implementation (and re-elaboration) of a new Sispg, using .NET Entity Framework Core and SQL Server, in order to unify the technologies used of this system with other systems of the UPE. Finally, this work has as main contribution to be part of a set of other works / articles that are being elaborated this semester, and in the subsequent, for the creation of new systems (and reformulation of the old ones) in order to guarantee data consistency on database, and avoid possible execution/rules errors, and lack of responsiveness on pages.*

***Key-words:***Sispg; UML*; Re-elaboration; Diagram.*



DOI: 10.xxxx/s11468-014-9759-3

# 1 Introdução

O Sistema de informações Sobre Pós-Graduação e Pesquisa (SISPG) surgiu no intuito de servir como apoio administrativo, e de gerenciamento, das pesquisas realizadas na Universidade de Pernambuco (UPE). Dada sua grande importância para a melhoria da governança organizacional acadêmica, este sistema foi concebido inicialmente para gerir quaisquer atividades de pesquisa da pós-graduação, bom como de disponibilizá-las para o conhecimento de toda a comunidade acadêmica. Porém, esse sistema contém apenas um número reduzido de funções, das quais nem todas se encontram facilmente acessíveis em diferentes dispositivos, que hoje é um requisito bastante exigido nos sistemas modernos.

Com base em dados fornecidos por uma pesquisa realizada pelo CETIC em 2017 **[1]**, há um crescimento no uso de dispositivos móveis cada vez maior nos acessos a sites e sistemas, em detrimento de uma queda de acessos em microcomputadores convencionais. Tal pesquisa também explicita que, pela primeira vez em nenhuma das 27 unidades da federação brasileira, o acesso exclusivo pelos computadores de mesa convencionais (*Personal Computers, PCs*) deixou de ser a principal forma de acesso, perdendo posição para as conexões via rede *internet* pelo celular. Também vale salientar que, neste mesmo período, a adoção por parte dos professores foi ainda maior, indo de 66% para 85% conforme citado em **[2]**.

Desta forma, quando olhamos para os estudantes (que serão futuros universitários) e os universitários veteranos/formados no que diz respeito ao acesso e Aplicações e Sistemas de Informação, fica notável a tal diferença de perfil. Também vale salientar que, de acordo com **[3]**, um software que se mostra falho implica que experiências desagradáveis podem ocorrer, e, normalmente, acontecem. Tais experiências como erros de execução e dificuldade de utilização deixam os usuários insatisfeitos, podendo inclusive levar a prejuízos.

Tratando do nosso material de trabalho, o Sistema de Pós-Graduação e Pesquisa (SISPG) vigente da UPE possui uma interface com o usuário de difícil acesso, desatualizada e que deixou de atender aos atuais requisitos de usabilidade, como a responsividade. Além disso, a Base de dados (*Database, DB*) utilizada pelo SISPG se encontra com sérios problemas estruturais (tais quais a falta de restrições de índice e de chaves estrangeiras, que tem causado inconsistências na DB) e as regras de negócio implantadas não estão totalmente aderentes aos processos da UPE.

# 2 Fundamentação Teórica

Esta seção apresentará o estado da arte necessário para o entendimento deste trabalho. Faremos o uso das seguintes técnicas e metodologias:

## 2.1 Processo Unificado Rational (RUP)

O RUP é um processo proprietário da Rational Software Corporation, que foi adquirida posteriormente pela IBM. Ela foi definida no intuito de aumentar as chances de sucesso de um projeto de software, e para garantir sua qualidade **[8]**.

Esta metodologia utiliza-se de orientação a objeto em sua concepção, e foi projetado e documentado utilizando a notação da Linguagem de Modelagem Unificada (*Unified Modeling Language*, UML) para ilustrar seus processos em ação. Tem como características principais ser: Iterativo (feito por meio de ciclos, focando na execução inicialmente, e depois no aperfeiçoamento) e incremental (entrega do software em forma de pacote, em que cada pacote tenha um valor/utilidade, e que ele consiga utilizar aquele pacote de código ainda que o software não esteja completamente pronto). Desta forma, cada ciclo de vida possui quatro fases, são elas:

- Concepção: Também conhecida como iniciação, esta fase é, em sua essência, estabelecer os objetivos da criação do sistema, e se é possível viabiliza-lo (tempo, recursos financeiros, pessoal para implementar/gerenciar, etc...) em um prazo (se houver) necessário.
- Elaboração: Esta fase serve para pré-aprovar o plano do projeto, a especificação de suas características, e a definição da linha base da arquitetura.
- Construção: É o momento em que se realiza/implementa de fato o produto.
- Transição: Serve para garantir que o software esteja disponível para os seus usuários finais (servidores no ar, domínio de divulgação de um site esteja comprado, aplicativo disponível em uma loja de sua plataforma, etc...).

Nessas fases, são sempre trabalhadas pelo menos nove disciplinas, demonstradas na Figura 1.

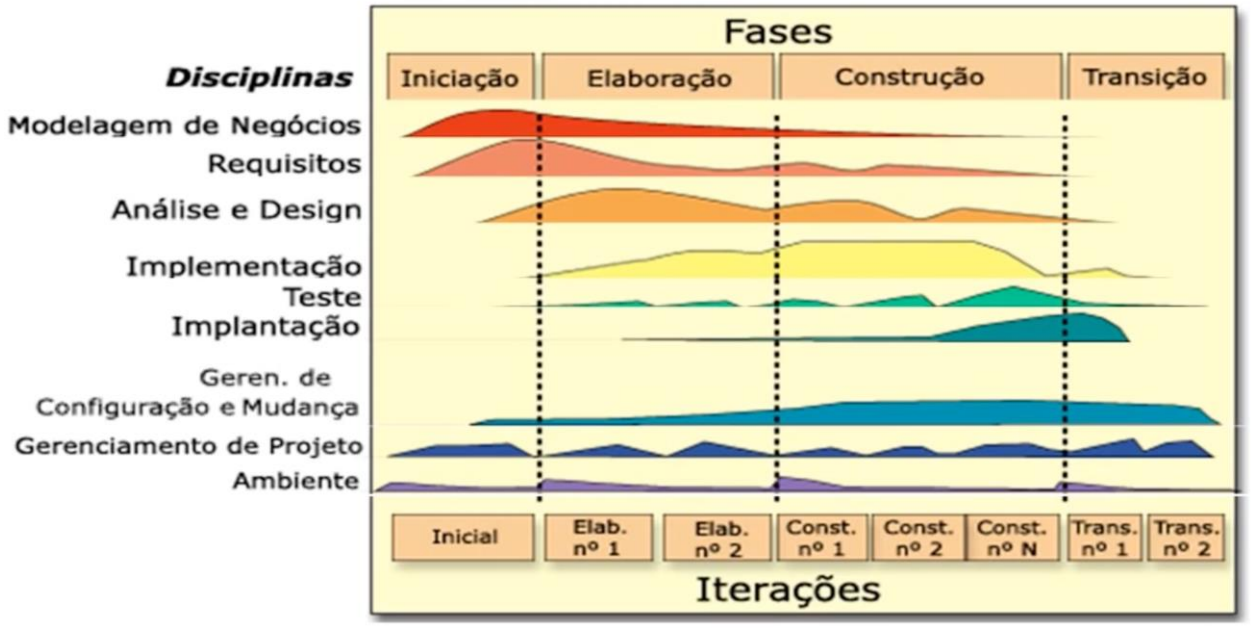



**Figura 1:** Relação das demandas de Disciplinas em cada Fase do RUP

## *2.2 Unified Modeling Language (UML)*

A UML é uma linguagem orientada a objetos com ênfase em modelagem de sistemas **[9]**. Diferentemente do RUP (que inclusive, é feito em UML). A UML não é proprietária, e foi criada a partir da fusão de três grandes métodos do BOOCH, OMT e OOSE. A sua proposta é que a equipe de desenvolvimento consiga visualizar, em forma de desenhos/diagramas padronizados, um planejamento prévio de estruturação/planejamento do escopo do projeto, que por inclusive tem sido muito utilizada na elaboração de softwares modernos.

As definições básicas para o entendimento deste artigo são:

- Caso de Uso: Também chamado de *Use Case (UC)*, trata-se de uma (ou uma sequência) de funcionalidades, que podem ser realizadas por um Ator (ver Figura 2). Sendo que cada tipo de ator pode ser o único com permissões para executar esse caso de uso, ou não. Obs: Um ator pode ser uma pessoa, ou até um outro sistema que poderá executar funcionalidades no sistema principal.

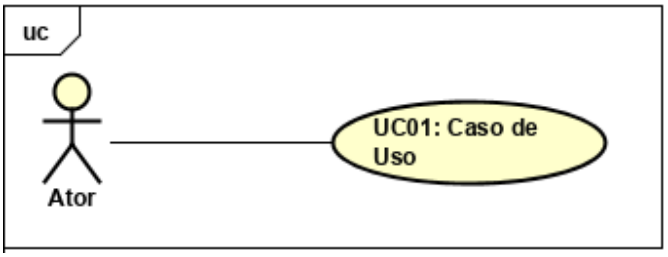



**Figura 2:** Definição de ator e de caso de uso.

- Classe: Trata-se de uma representação de um objeto qualquer. E que, como pode ser observado na Figura 3, salientamos que terão um nome (como é exemplo de "CadastroUsuarios", que está sendo estereotipado como uma 'Coleção'); atributos de um determinado tipo(Ex: "usuarios" do tipo Array de 'Usuario', que por sua vez é uma outra classe denominada 'Entidade' por possui uma tabela em um banco); e métodos que retornam valores (Ex: "procurarUsuario()", que nesse caso, está


DOI: 10.xxxx/s11468-014-9759-3

'Entidade' por possui uma tabela em um banco); e métodos que retornam valores (Ex: "procurarUsuario()", que nesse caso, estáretornandouma entidade "Usuario"). Além desses dois estereótipos, na Categorização BCE que estaremos utilizando, também haverá uma 'Fronteira' como "TelaLogar"(em que osatores utilizam para interagir com o sistema), e um 'Controle' a exemplo de "ControladorLogar" (que serve como intermediários entre objetos de controle e de fronteira). Obs: Os cenários em que entidades possuem atributos que tenham como tipo outras entidadessão denominadosrelacionamentos.

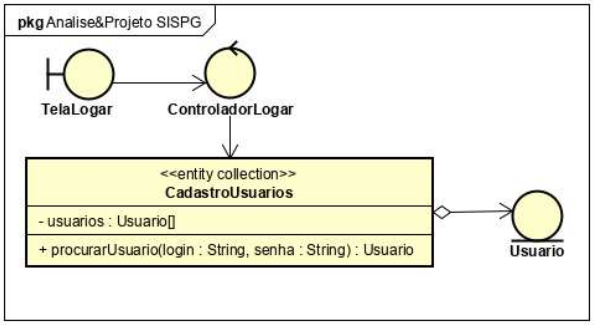



**Figura 3:**Definição de Classe.

### *2.3 Análise e Projeto de Software*

São duas etapas da criação do escopo do desenvolvimento de um Software. Primeiramente, a análise é o que modela o problema e qualquer outra atividade investigativa do domínio do problema[11]. Já o projeto é o momento de modelagem de uma solução e de sua implementação (como fazer).

A sua arquitetura de software envolverá definir uma solução estruturada que atenda a todos os requisitos técnicos e operacionais, e que, ao mesmo tempo, otimize atributos de qualidade comuns, como desempenho, segurança e capacidade de gerenciamento [10]. Ela implica em uma série de decisões baseadas em inúmeros fatores, e cada uma dessas decisões pode ter impacto considerável sobre a qualidade, o desempenho, a facilidade de manutenção e o sucesso geral do software.

## 3 Materiais e Métodos

Este artigosobre a reformulação de um novo SISPG,foi executado por Breno Medeiros (Discente de iC), em parceria com JoabeJunior (PesquisadorDocente), e trata-se de um trabalho incremental a um outro trabalho (também orientado pelo Professor Joabe Jesus) em **[4]**.

O novo material de contribuição a este projeto foi obtido utilizando-se dos dados coletados a partir do *FrontEnd Web* do SISPG, através das credenciais de Login (Usuário/Senha) dos próprios autores deste artigo, sem considerar os demais perfis de acesso, ou seja, um escopo reduzido.

Como método de elaboração da nova versão para o SISPG, serão definidos os Casos de Uso em forma de diagramas UML (*Use Case Diagram, UC*) e seus diagramas Classe de Análise e de Projeto (*Class Diagram, CD*), afim de situar as finalidades do SISPG e de contextualizar como tal sistema deve ser estruturado.

## 4 Reelaboração do SISPG

A seguir serão demonstrados os diagramas de Caso de Uso, que documentam as funcionalidades disponíveis (executadas ou não) para seus usuários, os quais serão as principais funcionalidades do sistema, incluindo as interações dessas funcionalidades entre usuários do próprio SISPG. Para isso, foram catalogadas funcionalidades disponíveis para dois tipos de Atores, são eles:

- **Usuário não Logado**: Usuários/Acadêmicos não cadastradas no Sispg, e que tenham interesse em conhecer Projetos, Cursos, Pesquisadores, Noticias, etc...
- **Usuário Logado**: Pesquisadores e/ou discentes cadastrados no Sispg, e que estão portanto, fazendo parte de alguma pesquisa (em Projetos ou Cursos)

Cada um desses atores terá seu próprio diagrama de Casos de Uso, com seus devidos acessos/funcionalidades.

### *4.1 UC Diagram: Usuário não Logado*

Ao acessar a página inicial do SISPG, todos os atores são inicializados como um ator genérico chamados internamente de Usuário, e isso implica que, ao inicializar no sistema, (sem necessidade de autenticação) um conjunto fixo de funcionalidades já serão habilitadas por padrão, e que são por tanto, de domínio para o público, como é mostrado na Figura 4.

A finalidade disso é permitir justamente que andamentos de pesquisa, bem como seus pesquisadores, sejam de conhecimento amplo, para que se possa propagar o conhecimento cientifico. E uma dessas funcionalidades está exercendo uma relação de extensão(*extend*), que é o Caso de Uso "Efetuar Cadastro" de outro caso de Uso, que é a funcionalidade de Logar no sistema, pois ela funcionará como uma ramificação do ato de realizar login, visto que ambos esses casos de uso normalmente se apresentam em pares na maiorias dos sistemas, e sempre é recomendável seguir padrões em projetos.

Observe que alguns *UC* estão inclusos(*include*) em outros, por fazerem parte na execução de um outro *UC* previamente iniciado, como é o caso do "Validar Dados", que ocorre na hora do Login, que também poderá ocorrer quando o próprio discente/docente vai validar que realizou cadastro com um endereço de *e-mail* de sua posse, ao clicar em um link enviado para sua caixa de Entrada, ou ainda ao clicar em "Esqueci minha senha" e validar uma nova senha. Na qual, em projeto, qualquer um desses 3 possíveis casos, o *UC* "Validar Dados" exigirá uma execução previa (não necessariamente no mesmo fluxo do Login) de um acesso ao sistema.

Ressaltamos também a herança de Casos que ocorre entre as consultas de Informações Básicas, para as outras funcionalidades públicas: "Consultar Tutoriais", "Consultar Noticia", "Consultar Arquivos" e "Consultar Perguntas Frequentes". Isto ocorre porque, além das informações apresentarem dados e comportamentos similares, internamente veremos nas próximas seções que se tratam de 4 entidades que herdam os atributos e métodos de uma classe chamada "InformacaoBasica".

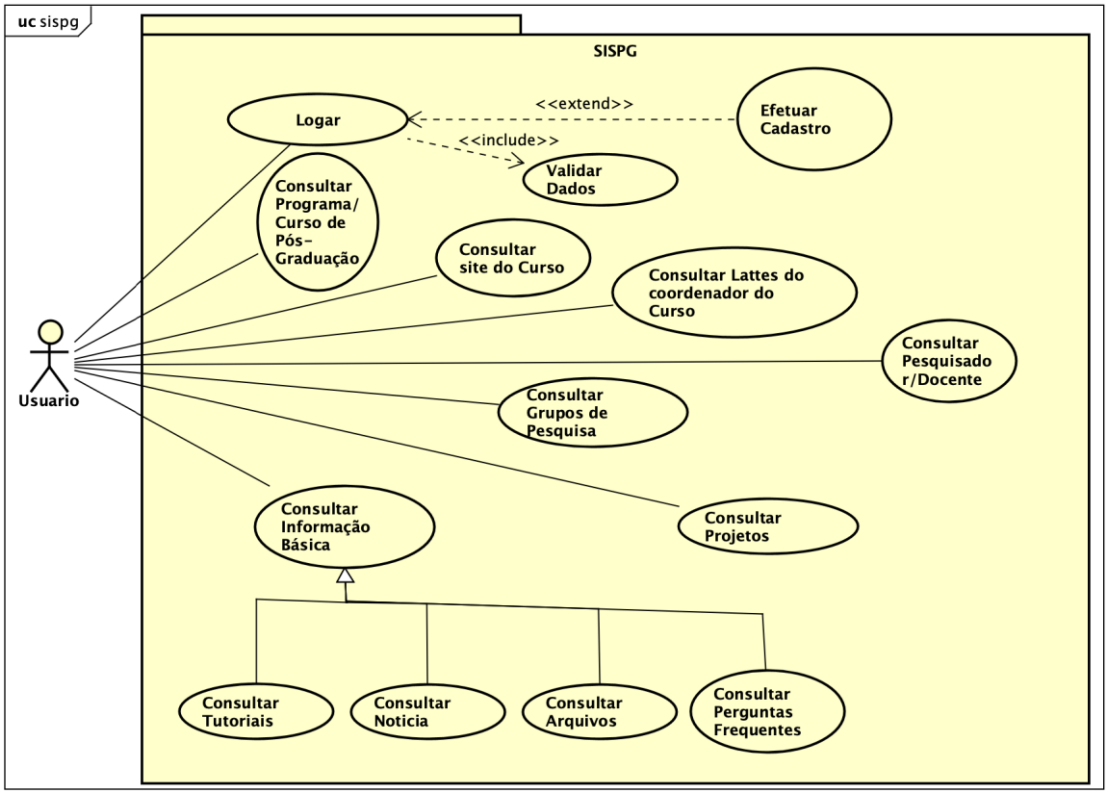



**Figura 4:** *UC Diagram* de Usuário não Logado, contendo todos os casos de uso e Funcionalidades que um usuário não logado pode exercer.


DOI: 10.xxxx/s11468-014-9759-3

## 4.2 U.C. Diagram: Usuário Logado

Já no diagrama de Caso de Uso em que houver o Login no SIPG (ver a Figura 5), as funcionalidades aparecem de forma distribuída para discentes, docentes, pesquisadores, e demais usuarios só sistema. Sendo que a disponibilidade de cada um dos recursos listados neste mesmo diagrama só será habilitada a depender do nível/tipo de permissões ao qual o usuário logado faz parte.

Ou seja, em um cenário em que um usuário logado fosse um universitário cursando a graduação, e que esteja tomando parte em um novo projeto de Iniciação Cientifica IC, ele não poderá/conseguirá usar o caso de uso "Gerenciar Curso de Pós-Graduação Stricto Sensu", pois esta funcionalidade só estará disponível apenas para docentes qualificados como "coordenador" de um Curso da Pós-graduação Stricto Sensu.

Como as mesmas credenciais(Login/Senha) serão única para cada utilizador (em detrimento de haver várias combinações de credenciais, para diversas finalidades distintas, de um mesmo usuário), os seus casos de uso serão os mesmos, e que estarão habilitados(ou não) a depender das circunstancias de permissões de um dado usuário, em uma dada tela/interface do sistema, e isso é feito através de herança de perfis de usuário.

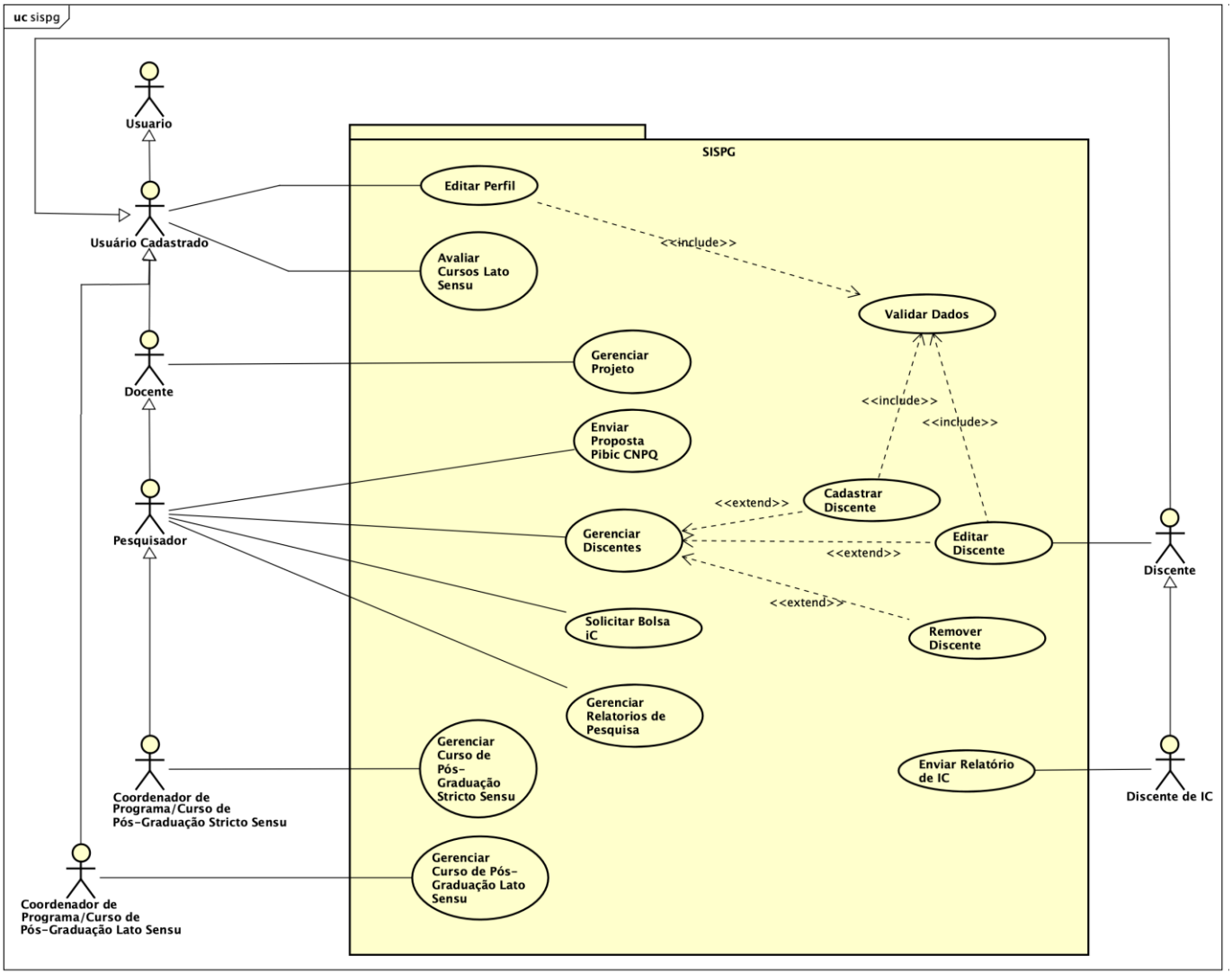



**Figura 5:** *UC Diagram* de Usuário Logado, contendo todos os casos de uso e Funcionalidades de um Usuário Logado.

# 5 Diagramas de Classes

No diagrama de Classes (*Class Diagram, CD*), iremos apresentar a especificação das classes, de forma a explicitar seus atributos (e tipos), os relacionamentos entre elas, além de alguns métodos de consulta de uma coleção (*List*/*Array*) de atributos de outras classes correlacionadas.

Para melhor apresentar o *CD,* foram organizadas e distribuídas conjuntos de classes em seus devidos pacotes (*package, pkg*), tanto para que haja um melhor entendimento do sistema, quanto para que se torne visualmente mais apresentável neste artigo.

## *5.1 PKG01: Pacote Sispg (Inicial)*

A Figura 6 apresenta o pacote raiz(*root*) do nosso *CD*, e ele será o único *pkg* com conterá outros pacotes *dentro* dele por questões didáticas. Nele podemos ver os outros 5 pacotes que temos no nosso projeto, e que estão numerados como subtítulos deste artigo de **5.2** até **5.6**.

Este *package* tem como objetivo principal apresentar os relacionamentos entre classes de pacotes distintos, e garantir o entendimento das entidades e seus relacionamentos mais importantes, tentando exibir o mínimo de informações necessárias para isso. Lembrando que o único package que houve ocultação de dados foi este, mas que as próximas seções deste artigo trarão de forma detalhada e completa todas essas informações abstraídas neste pkg.

De maneira especifica, neste pacote houve ocultação de classes que não se relacionam com classes de outros *packages* e os atributos de todas as classes. Porém não foram ocultadas as classes do "pkg packageOutros" (por esta pacote ser justamente um pkg que serve apenas para agrupar as classes que não

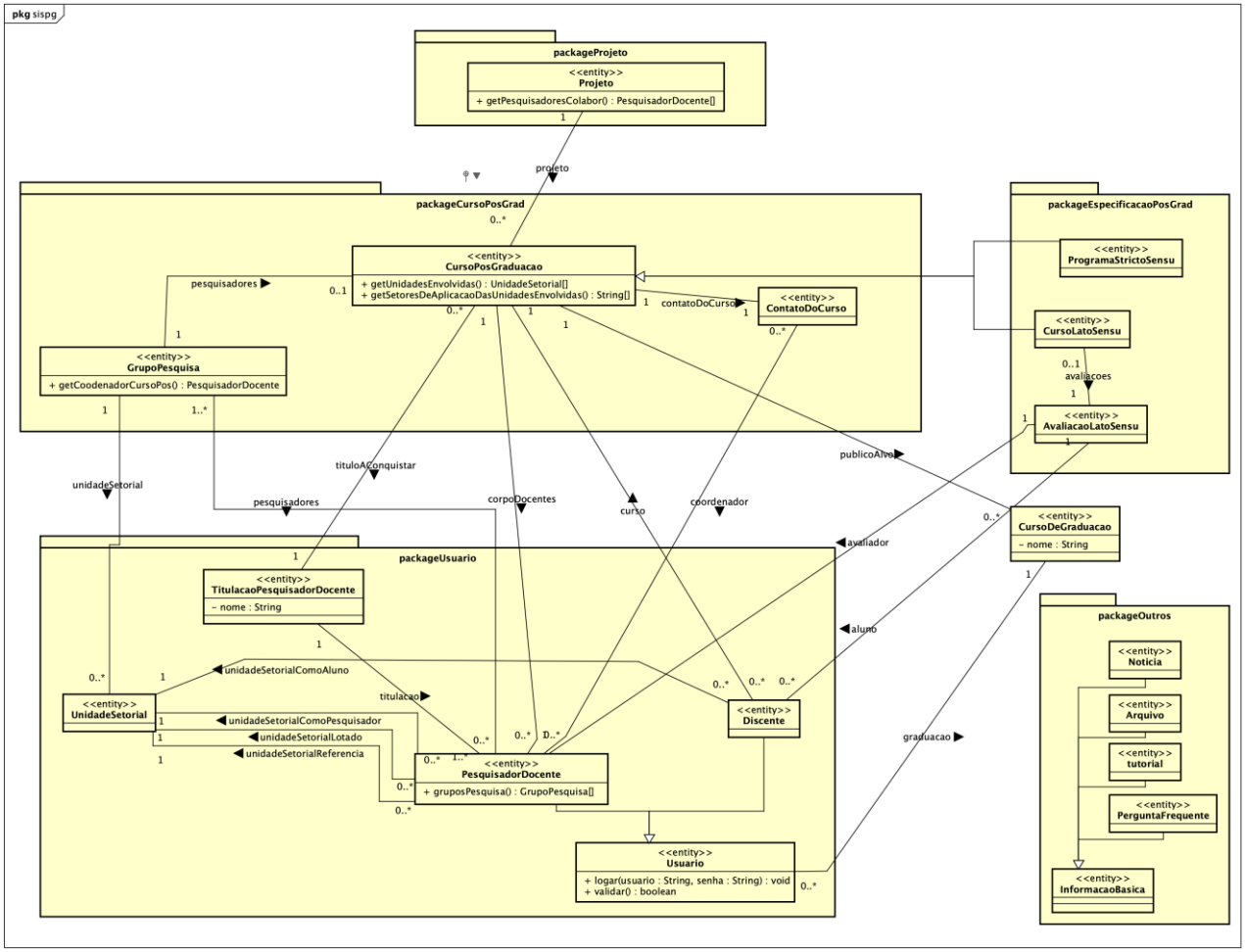



**Figura 6:** Pacote "pkg sispg", contendo os 5 outros pacotes desse sistema. Ele é também o único *package* que contém outro package.


DOI: 10.xxxx/s11468-014-9759-3

se relacionam com nenhuma classe de outros pacotes) e os métodos das classes que aparecerem neste pacote. Obs: Por questões intuitivas e estruturais, foi definido que a classe de "CursoDeGraduacao" será a única que ficará nesse pacote inicial.

## 5.2 PKG02: Pacote Projeto

A criação de pacotes para organização de diagramas é utilizada tanto pela questão didática/organizacional, quanto pelo interesse em manter abstrair informações. O mesmo ocorre na criação de classes, para que sirva de enumeração de um conjunto fixo de possibilidades(Ex: a classe "GrandeAreaConhecimento" possuir, em um dado momento, apenas 3 registros no banco, com valores: "Saúde", "Humanas" e "Exatas") que venha a aparecer na criação/edição de um projeto como uma listagem a ser escolhida, ou mesmo para que se faça uma distribuição dos atributos/informações de uma classe que já contém uma quantidade consideravelmente alta de atributos(Ex: classe "Projeto") para uma outra classe que possa armazenar estes outros atributos(Ex: classe "PeriodoRealização").

Este package da figura 7, foi criado especialmente para isso, visto que se observarmos o pacote "pkg

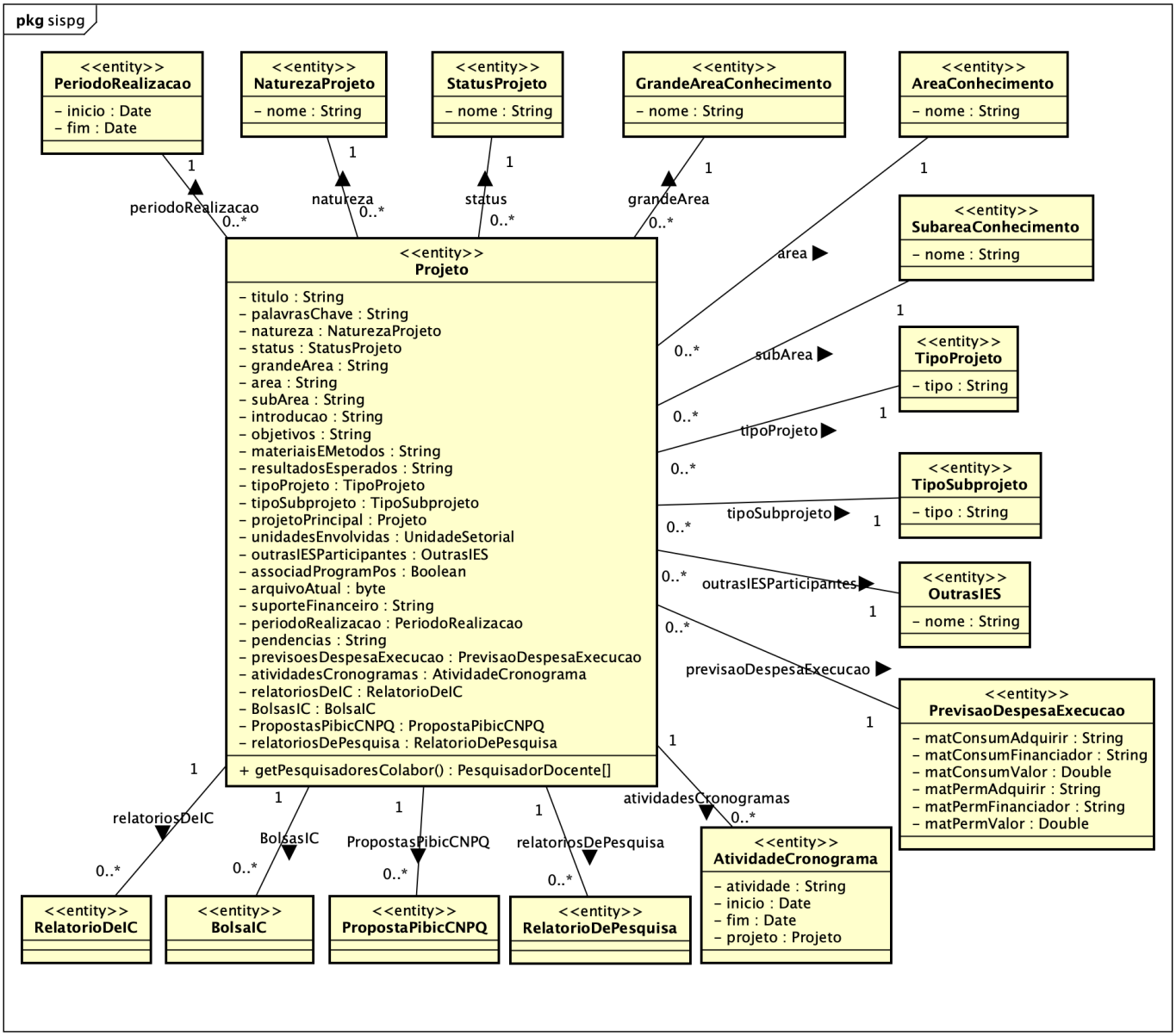



**Figura 7:** Pacote "pkg packageProjeto", contendo todas as classes que qualificam um Projeto de pesquisa.

sispg" na figura 6, veremos que temos a única classe do nosso *package* que se relaciona com classes de outros pacotes é a classe "Projeto". Vale ressaltar também que, apesar da falta de acesso a seus atributos e métodos, as entidades "RelatorioDeIC", "BolsaIC", "PropostaPibicCNPQ" e "RelatorioDePesquisa" puderam ser evidenciados como existentes no sistema.

### *5.3 PKG03: Pacote Usuário*

Aqui se encontrarão as classes de usuários (veja a figura 8), suas classes auxiliares para seus devidos cadastros (como "Nacionalidade", "Estado" e "OrgaoEmissor" de identidade), 3 tipos de "UnidadeSetorial" para "PesquisadorDocente" (que são de referência, como pesquisador e de lotação), e 1 tipo para usuários cadastrados como discentes (que seria a unidade setorial como aluno).

Ressaltamos aqui que, no caso da classe "Usuario", existe herança de seus atributos e métodos (bem como seus relacionamentos) para as suas classes filhas, que são as classes "PesquisadorDocente" e "Discente". Em casos assim, como as credenciais do sistema sempre se referem a um Pesquisador/Docente, ou a um Discente, evita-se a criação de uma tabela que venha a guardar registros da classe Usuário em banco que venha a ser desenvolvido a partir deste diagrama de classe. Isso porque, como um usuário sempre será ou Pesquisador/Docente ou Discente, faz mais sentido que se crie duas tabelas de banco, que seria uma para cada classe filha da classe Usuário.

Quando reparamos no relacionamento que existe entre a classe "PesquisadorDocente" com as classes do pkg "packageCursoPosGrad", tanto neste pacote quanto no package "sispg", evidencia-se que não há uma forma de verificar todos os grupos de pesquisa (classe "GrupoPesquisa") ao qual um

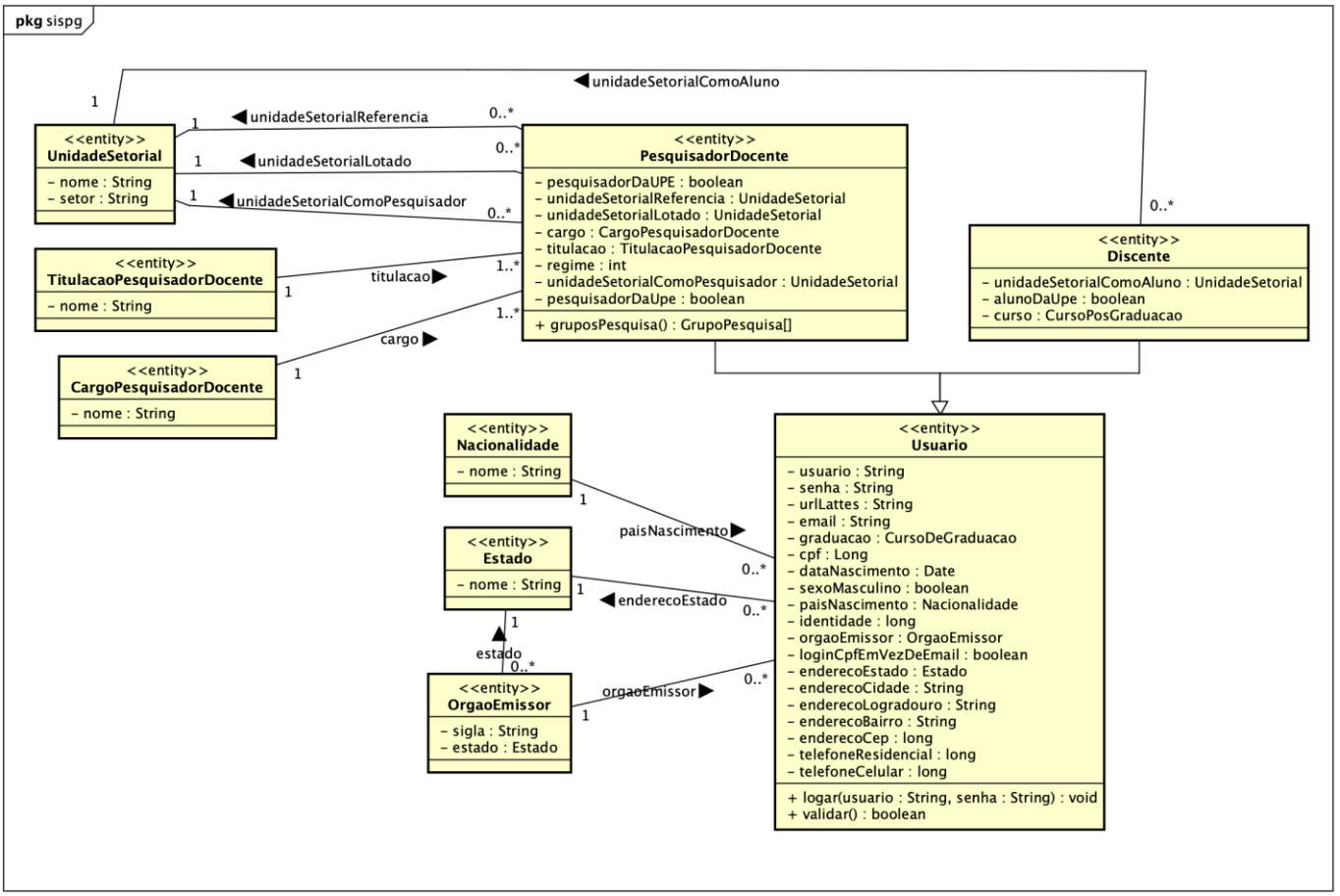



**Figura 8:** Pacote "pkg packageUsuario", é o mais conectado entre os pacotes. A maioria de suas classes se relaciona com classes de outros pacotes.


DOI: 10.xxxx/s11468-014-9759-3

pesquisador/docente está relacionado considerando-se também aqueles em que ele atua como coordenador de um curso (classe "ContatoDoCurso").

Como esta necessidade é algo que existe, e já está implementado no atual SISPG, definimos então o método gruposPesquisa(), que realizará uma consulta afim de retornar uma lista/*array* de um Objeto de Transferência de Dados (*Data Transfer Objects*, DTO) que poderá conter os valores dos atributos dos registros tanto de "GrupoPesquisa" quanto de "ContatoDoCurso" ao qual o Pesquisador/Docente que fez sua *call* pertença.

## 5.4 PKG04: Pacote Curso de Pós-Graduação

Este pacote de Curso de Pós-graduação (ver figura 9) é o pacote que tem o maior número de relacionamentos com outras classes, oriundas de outros pacotes, e isso é facilmente observado quando nos revemos o pacote raiz ("pkg sispg") na figura 6.

Inclusive, ele é o único *package* que se relaciona com classes de todos os pacotes (excluindo o pkg "packageOutros", pois este é um pacote reservado para as classes que são independentes de relacionamentos entre classes). No entanto, não é de se surpreender que isso ocorra, visto que o propósito principal do SISPG é catalogar seus cursos de Pós-graduação, bem como todos os dados que o englobar.

## 5.5 PKG05: Pacote Especificação de Curso de Pós-Graduação

Além daquele visto no "packageUsuario" (na seção **5.3**, figura 8), neste pkg de especialização do curso de pós-graduação também ocorrerá herança de classes. Sendo que aqui (ver figura 10) o que ocorre é uma herança entre classes de package diferentes: os filhos "ProgramaStrictoSensu" e "CursoLatoSensu" se encontram no pkg "package

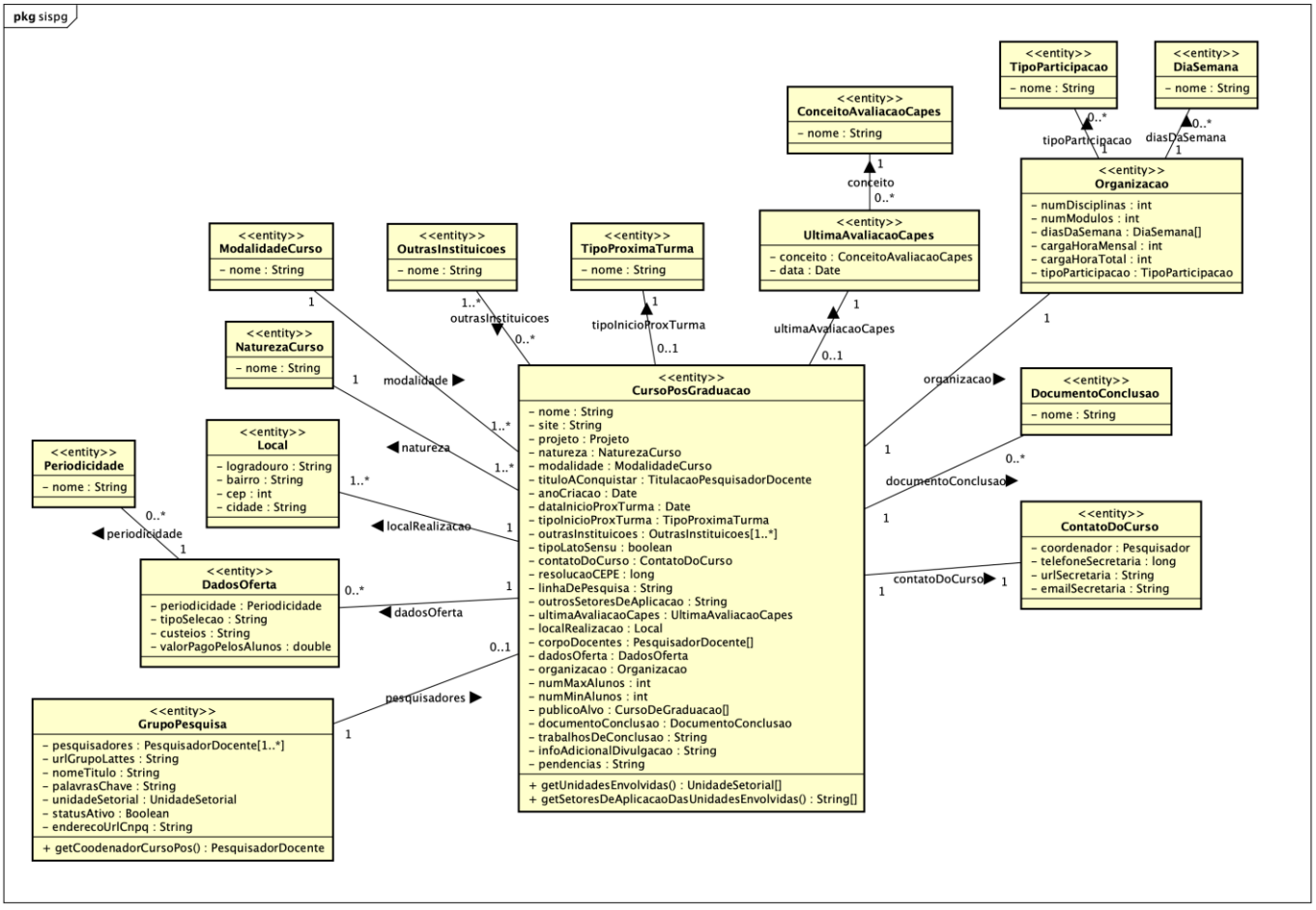



**Figura 9:** Pacote "pkg packageCursoPosGrad", contendo as principais classes sistema, que são os cursos de Pós-Graduação.

EspecificacaoPosGrad"; e o pai de encontra no pkg "packageCursoPorGrad".

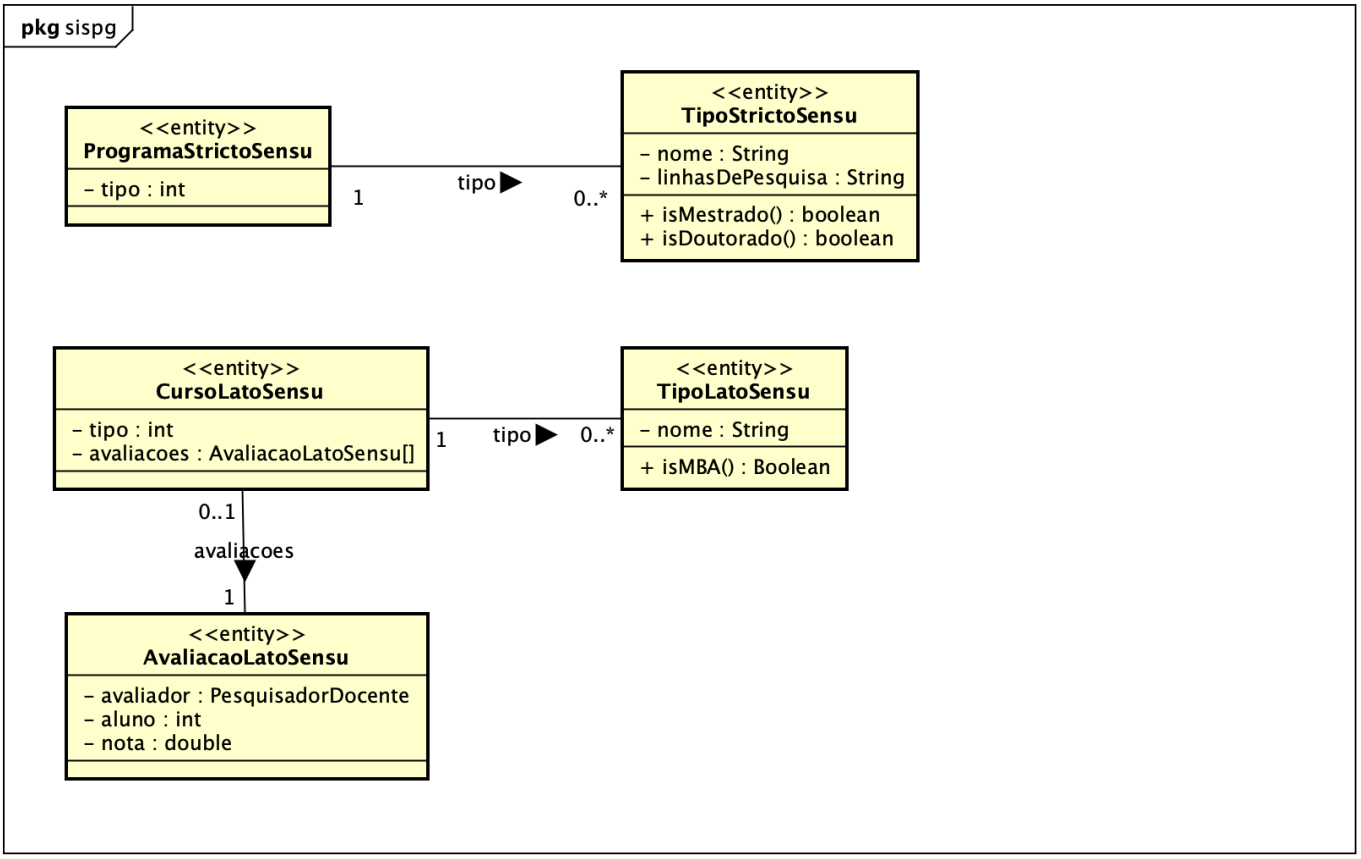



**Figura 10:** Pacote "packageEspecificacaoPosGrad", contendo o segundo e último caso de herança entre classes, que dá origem aos cursos Lato Sensu, e aos Programas Stricto Sensu.

## *5.6 PKG06: Pacote Outros*

Por fim, este é o ultimo pkg, e nele contém as 4 únicas classes independentes (sem relacionamentos com outras classes externas a este próprio pacote) do sistema, isso porque elas aparecem apenas como listagem de informações e consultas autossuficientes.

Porém como suas telas de listagem/consulta e comportamentos se apresentam de formas parecidas/similares no sistema, a elas serão feitas herança para uma outra classe denominada "InformacaoBasica", a ser implementada na etapa de Projeto.

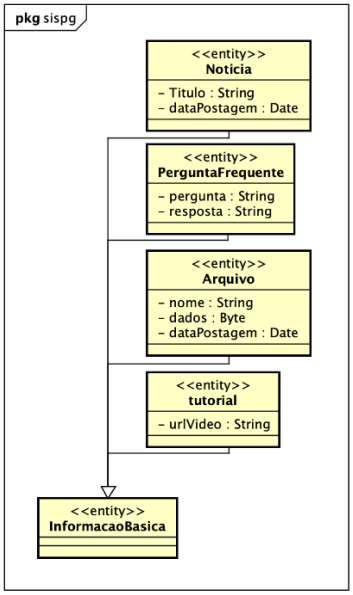



**Figura 11:** Pacote packageOutros, contendo classes que não têm relacionamento com outras classes.


DOI: 10.xxxx/s11468-014-9759-3

# 6 Diagramas de Classe de Análise e Projeto

De forma a complementar os diagramas de Classe demonstrados na seção anterior, e afim de trazer uma nova proposta de implementação do Sispg, serão apresentados nas próximas duas subseções o Diagrama de Análise & Projeto.

## 6.1 Diagrama de Análise

Este diagrama visa o estudo do sistema como ele se apresenta hoje, e que sua versão resumida pode ser consultada na figura 12. Ela é representada através dos elementos de Fronteira(*boundary*), Controle(*controller*), Coleção (*entity collection*), e Entidade(*entidade*), todos eles seguindo a categorização BCE como foi demonstrado na seção **2.2**, figura 3. Obs: Versão completa disponível no Apêndice 1.

## 6.2 Diagrama de Projeto

Nesta seção trataremos de uma nova proposta realizada a partir do Diagrama de Análise. Aqui apresentaremos uma nova proposta para o Sispg, em que a versão resumida apresenta-se na figura 13, e a versão completa estará disponível no Apêndice 2.

Alguns exemplos dessa mudança aparecem como herança de classes abstratas como "InformacaoBasica", definição de interface de Repositórios, e a definição de uma *Fachada* entre as Fronteiras e seus Controladores.

# 7 Conclusões

O estudo da estrutura interna do SISPG neste Trabalho de Conclusão de Curso TCC foi realizado como mais um passo em direção a reformulação desse sistema, visando uma melhor facilidade e agilidade para estudantes, professores e servidores. Ele tem sido uma continuação de trabalhos passados na própria UPE **[8]** que seguiu o ciclo de vida do RUP e realizou a fase de concepção.

E no intuito de fazer esta reelaboração do Sistema de Pós-graduação e Pesquisa se tornar realidade, este artigo teve como finalidade realizar a fase da elaboração, sendo que, a partir dos acessos como Discente(Breno Medeiros) e Docente(Joabe Jesus), foram trazidos todos os Diagramas de Caso de Uso, bem como os Diagramas Classe de Análise e de Projeto. Por fim, considerando nossas permissões de acesso, esperamos por novos trabalhos que venham a realizar a definição dos atributos das classes que não são as entidades, diagramas de estado e de sequência, e das duas últimas fases (Construção e Transição).

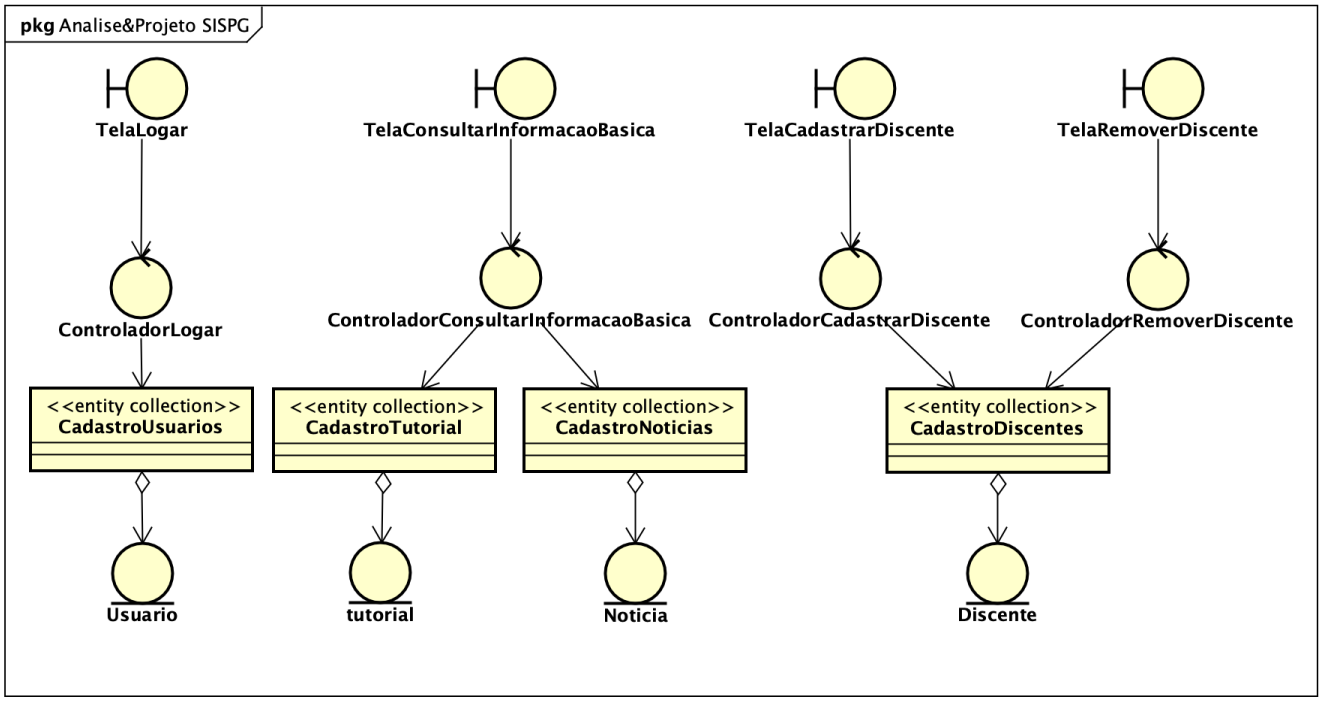



**Figura 12:** Versão resumida do Diagrama de Classe de Análise.

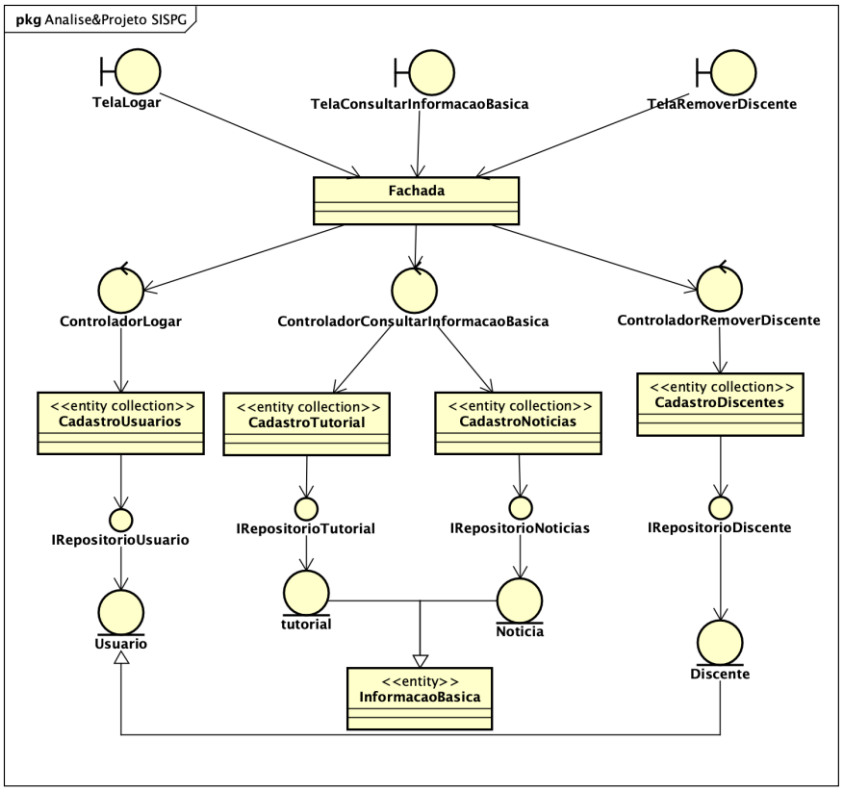



**Figura 13:** Versão resumida do Diagrama de Classe de Projeto.

## Referências


**[1]** CETIC. TIC Domicílios 2015. Disponível em: <http://cetic.br/media/analises/tic_domicilios_2015_coletiva_de_imprensa.pdf>. Acesso em: 22 abril 2019.

**[2]** CETIC. TIC Educação 2015. Disponível em: <http://cetic.br/media/analises/tic_educacao_2015_coletiva_de_imprensa.pdf>. Acesso em: 22 abril 2019.

**[3]** PRESSMAN, R. MAXIM, B. **Engenharia de Software**. 8.ed. São Paulo: McGraw-Hill Interamericana do Brasil,. 2016.

**[4]** SILVA, E. JESUS, J. Análise de Responsividade do Sistema de Informações Sobre Pós-Graduação e Pesquisa (SISPG). **TCC**. UPE, dez,. 2017. Disponível em: <https://tcc.ecomp.poli.br/20172/TCC%20Eduardo.pdf>. Acesso em: 10 junho 2019

**[5]** BOOCH, Grady. **UML. Guia do Usuário.** 1.ed. Rio de Janeiro: Elsevier,. 2006. [6] BLAHA, Michael. RUMBAUGH, James. **Modelagem e Projetos Baseados em Objetos com UML 2**. 2.ed. Rio de Janeiro: Elsevier,. 2006.

**[7]** WAZLAWICK, Raul. **Análises e Projetos de Sistemas de Informação Orientados a Objetos**. 2.ed. Rio de Janeiro: Elsevier,. 2010.

**[8]** FIKANI, A. JESUS, J. Um Sistema de prontuário baseado na plataforma Helthvault. **TCC** . UPE, nov,. 2014. Disponível em: <https://tcc.ecomp.poli.br/20142/monografia-Afif%20FINAL.pdf>. Acesso em: 10 junho 2019.

**[9]** INFOESCOLA. UML. Disponível em: <https://www.infoescola.com/engenharia-de-software/uml/>. Acesso em: 10 junho 2019.

**[10]** MICROSOFT. MSDN. Disponível em: < https://msdn.microsoft.com/pt-br/hh144976.aspx/>. Acesso em: 10 junho 2019.

**[11]** JACQUES, A. **Análise e Projeto Orientados a Objeto**. Disponível em: < http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/intro/intro.htm>. Acesso em: 10 junho 2019.



DOI: 10.xxxx/s11468-014-9759-3

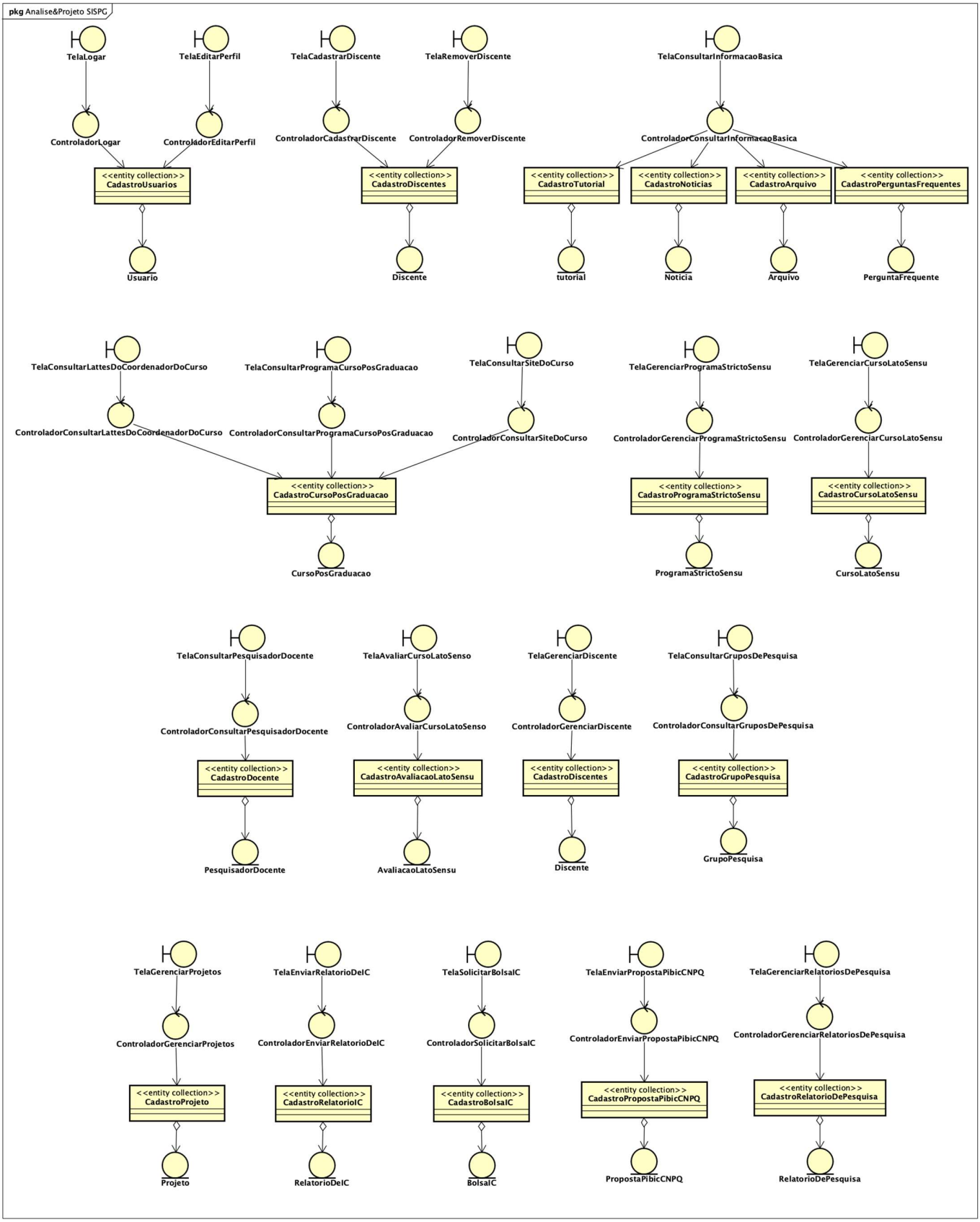
De acordo:

Assinatura do Discente

Assinatura do(a) Orientador(a)

Recife: 10 / 06 / 2019

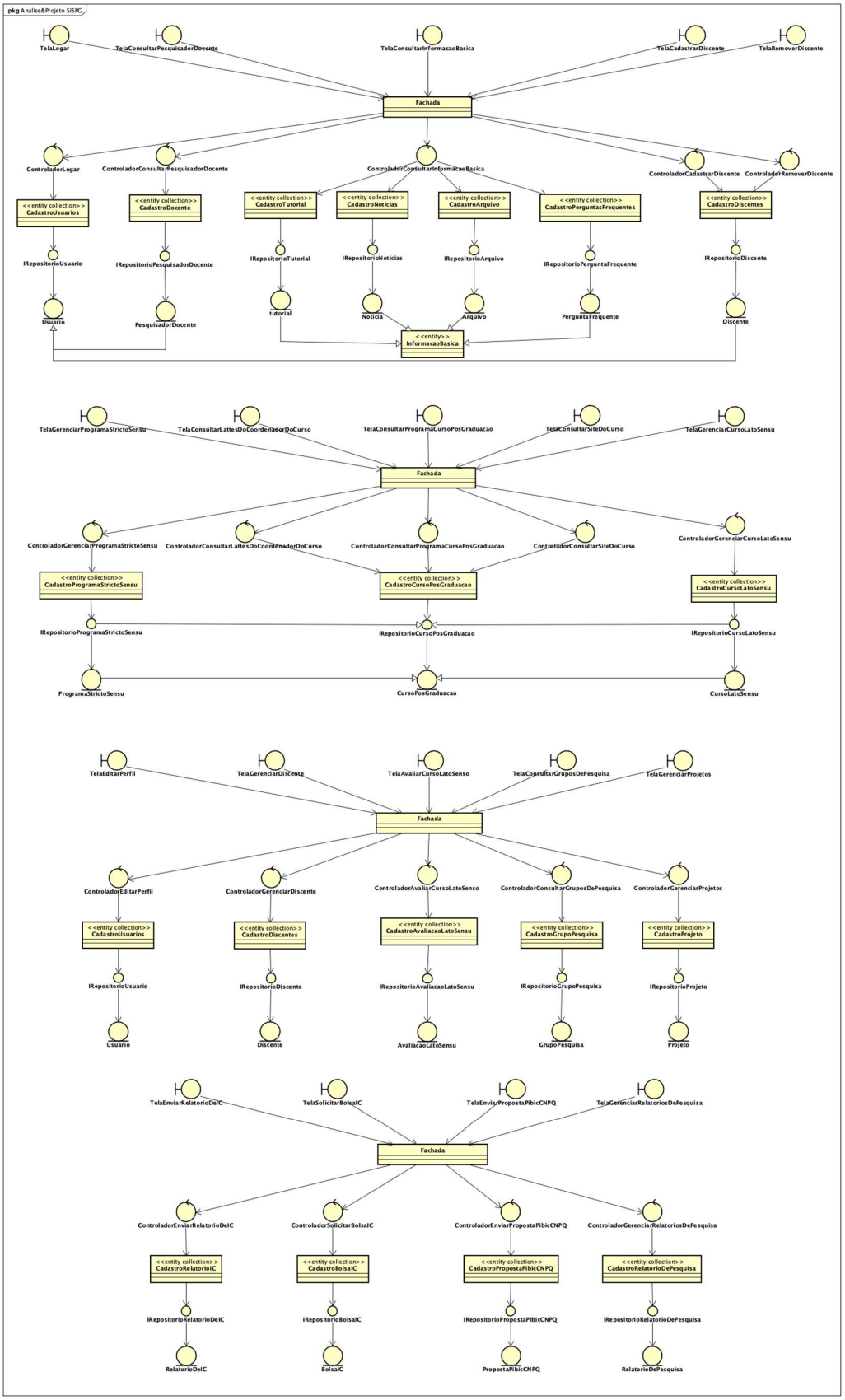
## [Apêndice 1]




**Apêndice 1:** Diagrama de Classe de Análise do Sispg. Disponível em: <https://drive.google.com/file/d/12SB1VkhUb_Pu1JTTM0eSSQQAh-TDxMBi/view?usp=sharing>

DOI: 10.xxxx/s11468-014-9759-3

# [Apêndice 2]




**Apêndice 2:** Diagrama de Classe de Projeto do Sispg. Disponível em: <https://drive.google.com/file/d/12cmNADCMpGbIIkyoeudkmwYxxp4BBmL8/view?usp=sharing>