Programação Orientada a Objetos Avançada

Framework para Sistemas de Ensino

Caio Ueno	743516
Daniel Heringer	743524
Gabriel T. Nardy	726521

25 de junho de 2019

Prof. Dr. Valter Vieira de Camargo

Universidade Federal de São Carlos

1. **Descrição do sistema**

O projeto desenvolvido aborda a implementação de um *framework* para dar suporte para sistemas de ensino, de forma que seja possível realizar ações e comandos comuns nessa temática.

O objetivo foi criar classes abstratas que possuíssem uma alto grau de generalização, com um conjunto de atributos e métodos, abstratos ou não, que aparecem sempre nesse contexto, pois trata-se de um *framework*. Além disso, foram pensadas questões que poderiam ser implementadas por entidades específicas, como universidades e escolas de ciclo básico.

Neste projeto, além do *framework*, foi implementado uma situação específica de seu uso, uma aplicação. O exemplo é um sistema de uma universidade, de forma enxuta, apenas para demonstrar funcionalidades que julgamos importantes neste *framework*.

Framework

No *framework* não existem pacotes, pois todas as classes são públicas, uma vez que todas têm alto potencial de serem estendidas.

As classes implementadas no *framework* são:

- **Grade**
  Classe responsável por armazenar as disciplinas de um aluno, e também a qual curso essa grade está relacionada. Também deve gerenciar o status das disciplinas, como aprovado ou reprovado.

- **FachadaGrade**
  Classe pública a ser estendida na aplicação contendo um método estático que retorna uma interface do tipo IGrade.

- **Aluno**
  Representa um aluno e suas informações, contém um identificador único, nome, e-mail e *Grade*. Além disso, armazena uma lista de todas as turmas que ele participa ou já participou, juntamente com a nota final que recebeu em cada dessas turmas. Para cada turma existe uma instância da classe *NotaDisciplina* e armazena a nota, chamada de classe de associação.

- **FachadaAluno**
  Classe pública a ser estendida na aplicação contendo um método estático que retorna uma interface do tipo IAluno.

- **Professor**
  Mantém os dados de um professor, como identificador, nome, e-mail e sua área de formação. Contém uma lista das turmas que ministra/ministrou.

- **Disciplina**
  Armazena as informações básicas de uma disciplina, possui um identificador, nome, credito (horas) e uma ementa.

- **FachadaDisciplina**
  Classe pública a ser estendida na aplicação contendo um método estático que retorna uma interface do tipo *IDisciplina*.

- **Turma**
  Classe para manter os dados de uma turma, como seus alunos, a qual disciplina está atrelada, quando começou/terminou, os lugares onde será ministrada a aula e claro um identificador único.

- **NotaDisciplina**
  Criada unicamente para separar a responsabilidade de guardar uma nota para cada disciplina cursada por um aluno.

- **Curso**
  Armazena as informações de um curso.

- **Oferta**
  Classe que contém uma lista de turmas, uma vez que toda turma precisa estar atrelada a uma oferta, como por exemplo uma oferta de uma mesma disciplina para turmas diferentes, turma A, B, C, etc.

- **SalaHorario**
  Tem a responsabilidade de guardar os dados de uma sala juntamente com um horário ao qual será ministrada uma turma.

- **EntidadeResponsavel**
  Representa uma entidade que tem por objetivo controlar um ou mais cursos, além de professores, de forma independente. Possui uma sigla, nome, área, uma lista de professores próprios, em destaque também um professor responsável e os cursos administrados.

- **FachadaEntidadeResponsavel**
  Classe pública a ser estendida na aplicação contendo um método estático que retorna uma interface do tipo *IEntidadeResponsavel*.

- **Escola**
  Classe utilizada para armazenar de forma global as informações de alunos, professores e turmas. No sistema deve ter somente uma instância dessa classe, por isso utiliza o padrão singleton.

Interfaces implementadas de classes que têm visibilidade de pacote:

- **IAluno**
  Métodos públicos que outras classes podem usar quando manipulam uma interface de *Aluno*.

- **IDisciplina**
  Viabiliza alguns métodos para que seja possível manipular interfaces do tipo disciplina.

- **IGrade**
  Agrupamento dos métodos públicos de uma interface do tipo *IGrade*.

- **IEntidadeResponsavel**
  Interface do tipo *EntidadeResponsavel* e métodos que podem ser utilizados por outras classes.

Obrigações do engenheiro de aplicação:

Para a implementação de uma aplicação que utiliza o *framework*, são necessárias algumas diretrizes.

Todas as classes, exceto a classe *Escola*, precisam ser estendidas, uma vez que todas são abstratas, com exceção das classes de fachada. Duas classes possuem métodos abstratos que devem ser implementados na aplicação, as classes *Aluno* e *Grade*.

As interfaces *IAluno*, *IDisciplina*, *IEntidadeResponsavel* e *IGrade*, não precisam ser necessariamente estendidas, a não ser que a aplicação crie novos métodos nas respectivas classes dessas interfaces e queira que eles sejam públicos pela interface. Caso sejam criadas essas interfaces, então elas devem estender suas respectivas interfaces do *framework* e serem implementadas por suas classes concretas correspondentes.

As classes *FachadaDisciplina*, *FachadaEntidadeResponsavel* e *FachadaGrade* devem ser obrigatoriamente estendidas, pois são a ponte para que as demais classes possam manipular classes com visibilidade de pacote através de suas interfaces. Devem possuir métodos que retornem interfaces de suas respectivas classes, na realidade simulam/implementam o banco de dados. A classe *FachadaAluno* foi implementada por uma maior comodidade para manipular os dados.

Na aplicação é recomendado que as classes que estendem as classes *Grade*, *Disciplina*, *NotaDisciplina*, *Oferta* e *EntidadeResponsavel* possuam visibilidade de pacote, para um maior controle e proteção dos dados.

Aplicação

- **GradeUniversitaria**
  Estende a classe *Grade*, possui um histórico, onde armazena uma nota para cada disciplina.

- **FachadaGradeUniversitaria**
  Classe que possui um método estático que simula um banco de dados, retornando uma interface do tipo *IGradeUniversitaria*.

- **AlunoUniversitario**
  Estende a classe *Aluno*, possui alguns atributos novos, CPF, RG, data de nascimento, IRA e endereço, além de métodos para manipular esses atributos. Implementa também métodos abstratos da classe pai *Aluno*.

- **FachadaAlunoUniversitario**
  Classe que possui um método estático que simula um banco de dados, retornando uma interface do tipo *IAlunoUniversitario*.

- **ProfessorUniversitario**
  Estende a classe *Professor*, incluindo alguns atributos, como salário base, CPF, RG, bônus e um grupo de pesquisa.

- **Matéria**
  Estende a classe *Disciplina*, sem novos atributos ou métodos.

- **FachadaMatéria**
  Classe que possui um método estático que simula um banco de dados, retornando uma interface do tipo *IMateria*.

- **TurmaUniversidade**

  Estende a classe *Turma*, possui uma lista com os professores que ministram aquela turma. Além disso, possui alguns métodos para manipular a lista de professores.

- **NotaDisciplinaUniversitaria**

  Apenas estende a classe *NotaDisciplina*, sem novos atributos ou métodos.

- **CursoUniversitario**

  Estende a classe *Curso*, adicionando um coordenador e um vice coordenador do curso e um projeto pedagógico.

- **OfertaUniversitaria**

  Apenas estende a classe *Oferta*, sem novos atributos ou métodos.

- **LugarHorario**

  Estende a classe *SalaHorario* e adiciona um AT - prédio de aula teórica.

- **Departamento**

  Estende a classe *EntidadeResponsavel*, sem novos atributos ou métodos.

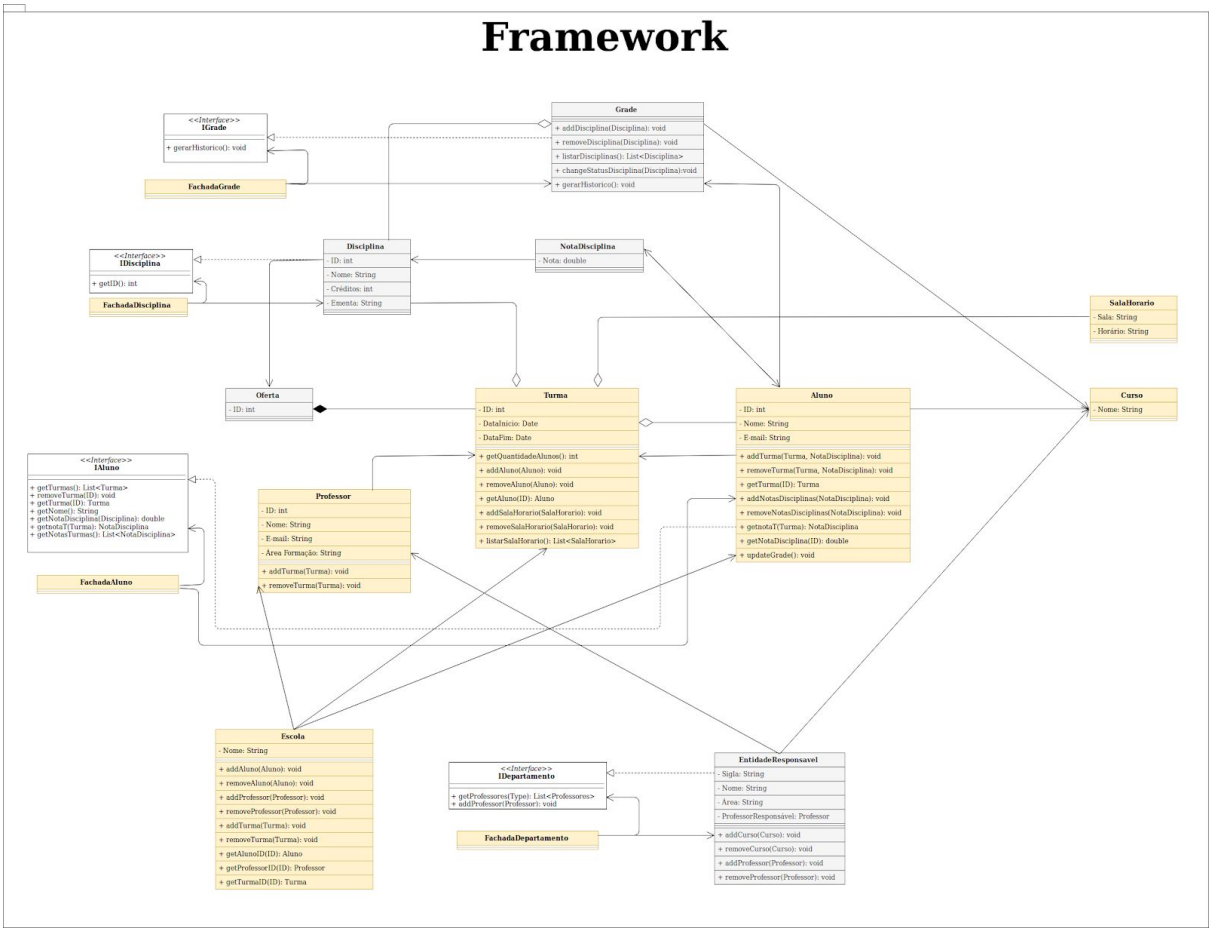
- **FachadaDepartamento**

  Classe que possui um método estático que simula um banco de dados, retornando uma interface do tipo *IDepartamento*.

2. Diagrama de classes

Framework

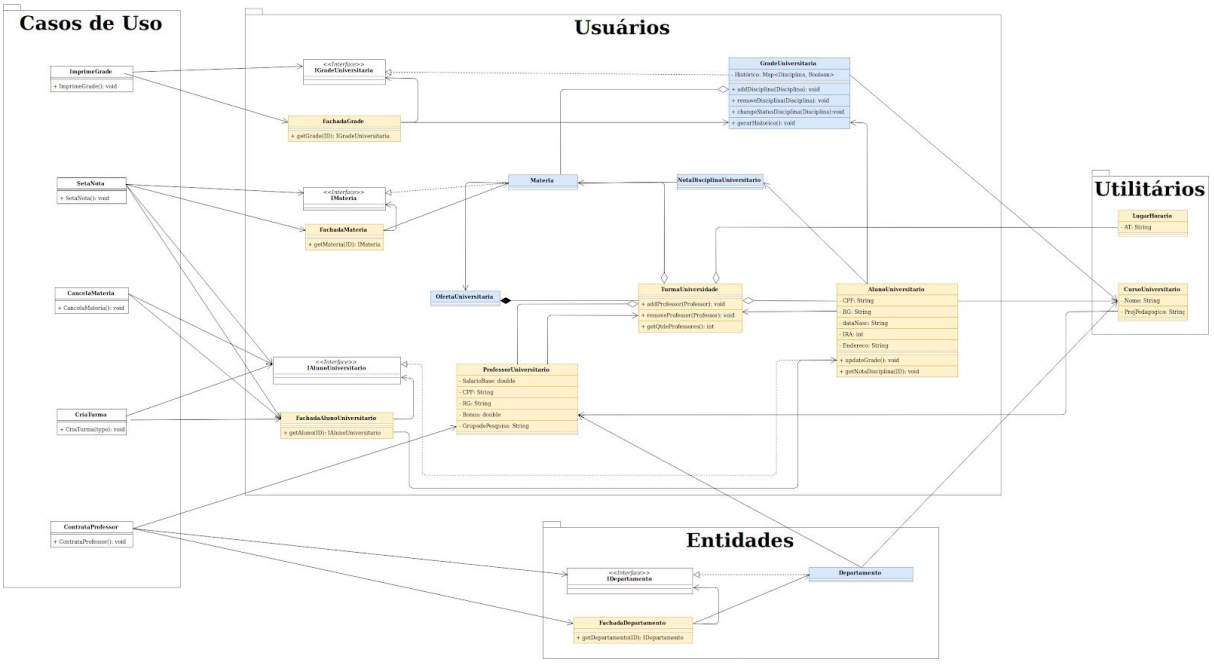
- **Diagrama de classes completo**



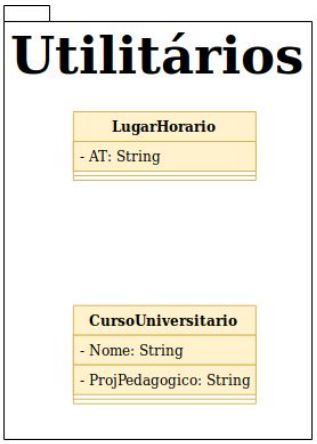
Observação: É recomendado que as classes em cinza tenham visibilidade de pacote na aplicação a ser desenvolvida.

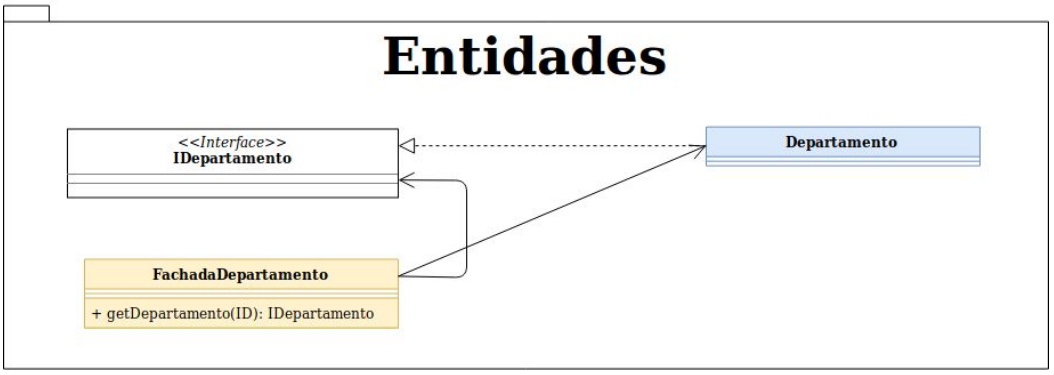
Aplicação

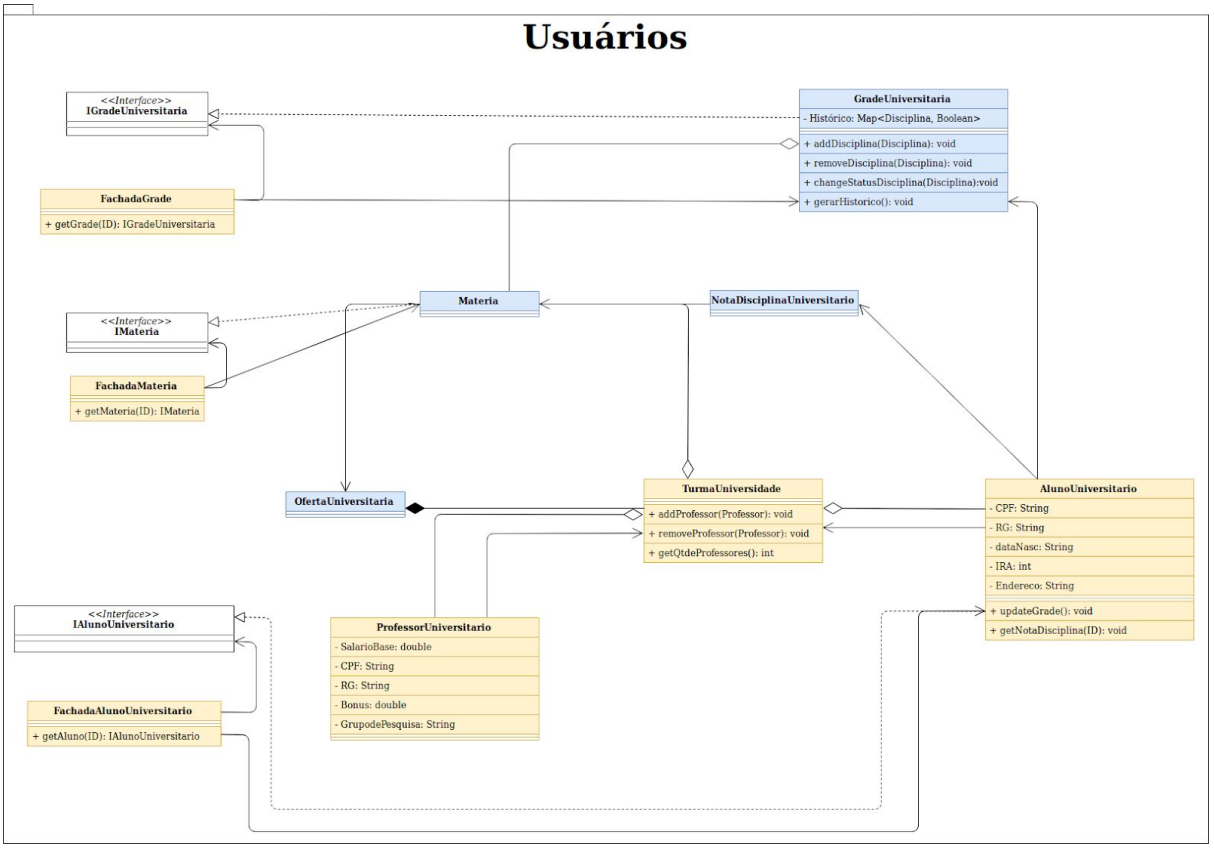
- **Diagrama de classes completo**



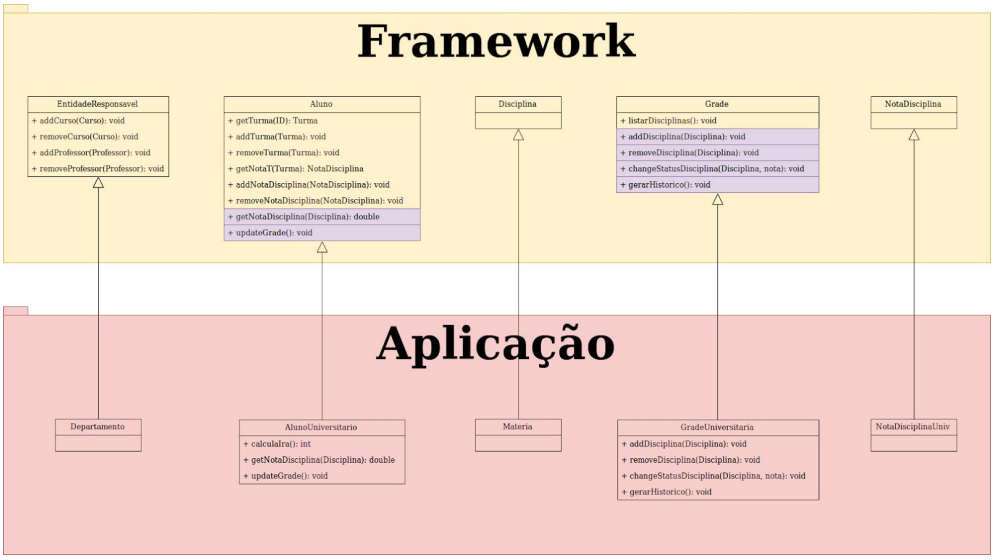
- **Diagrama de cada pacote, exceto do casos de uso**

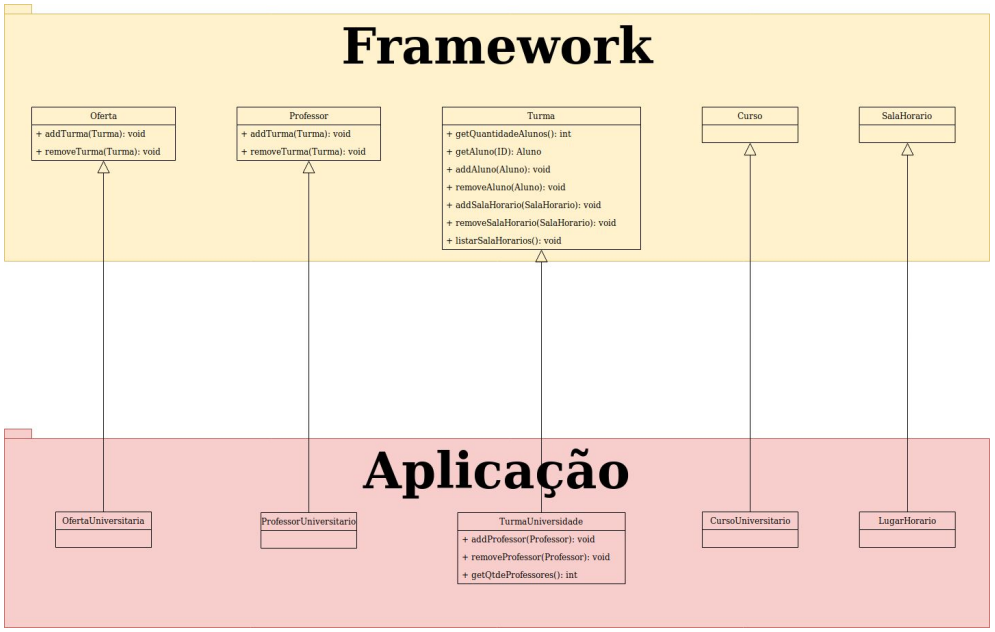


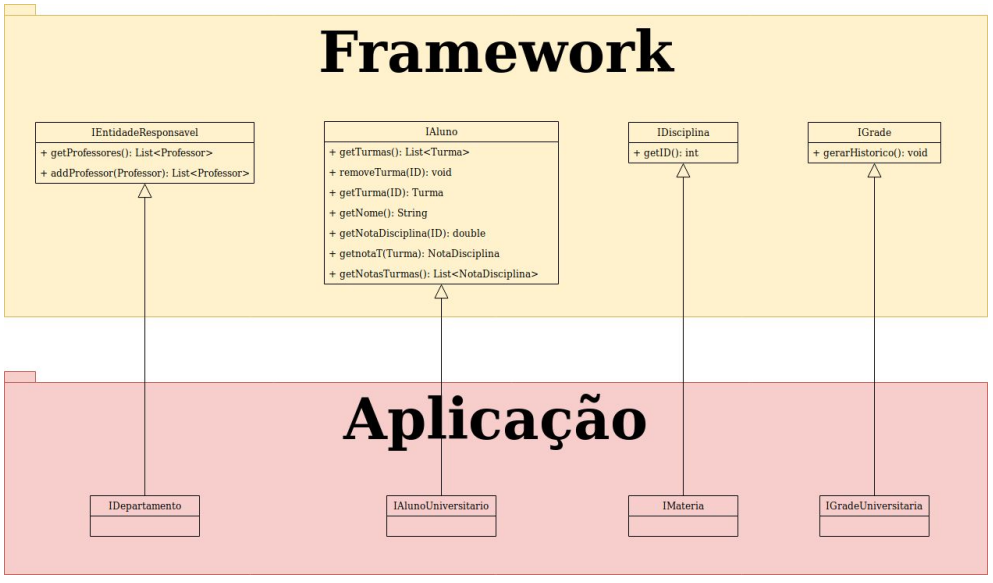


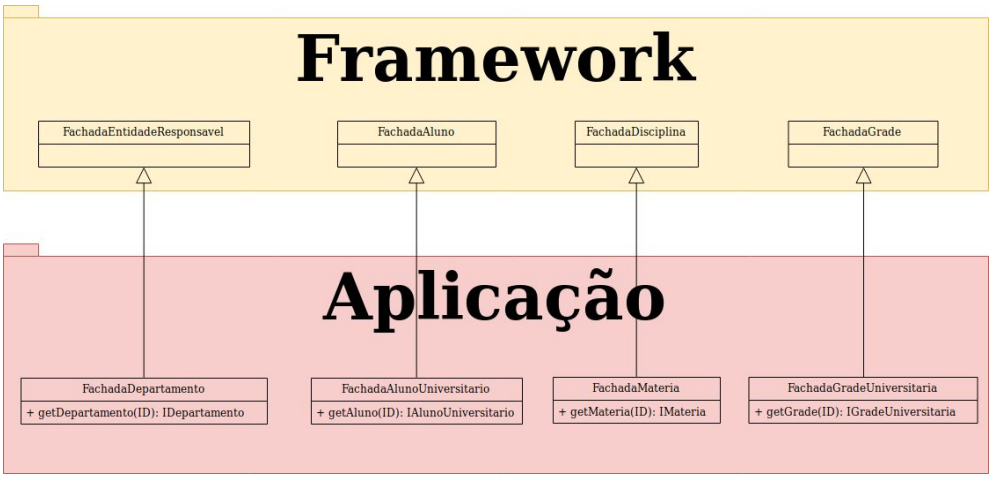


- **Diagrama das heranças**









***Observação:** Os métodos destacados em roxo são abstratos, por isso aparecem tanto no *framework* quanto na aplicação.

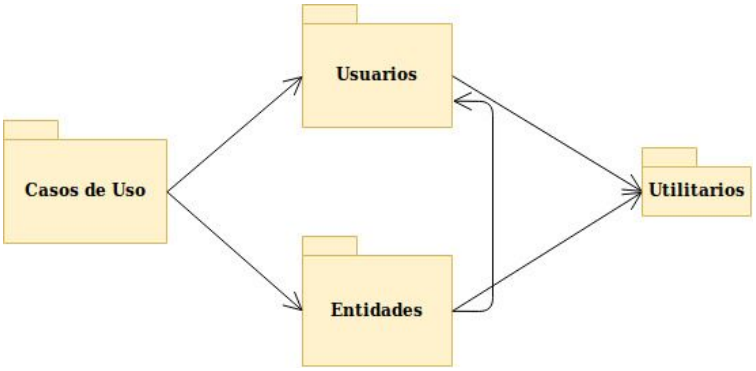
3. **Diagrama de pacotes**

Framework

Como mencionado acima, o *framework* não possui pacotes devido todas as classes serem públicas.

Aplicação

Para a aplicação, o seguinte diagrama ficou estabelecido:

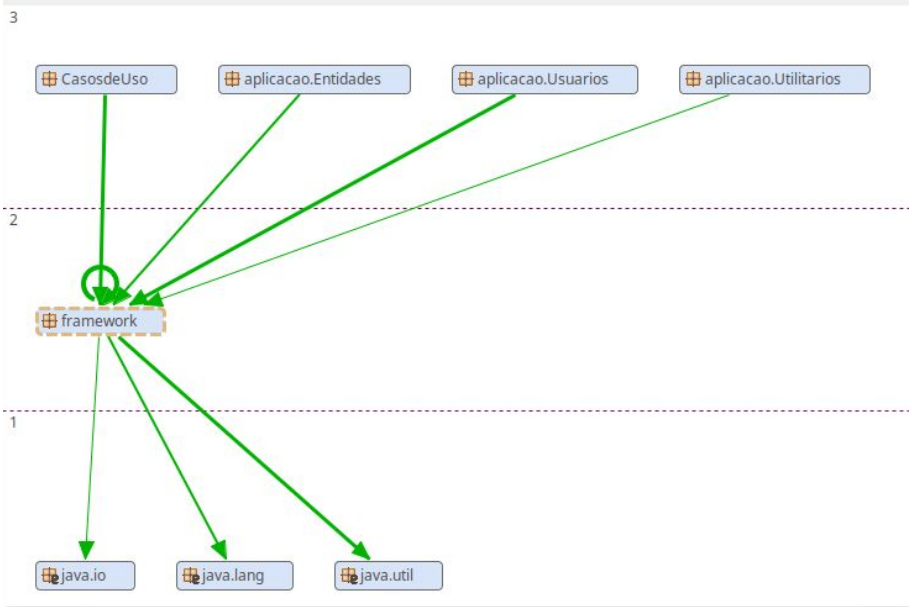


4. **Casos de uso implementados**
   a. **CancelaMateria**: O aluno deseja cancelar sua participação em uma determinada disciplina, encerrando seu vínculo com a turma que participa.
   b. **ContrataProfessor**: Simula a criação de um novo objeto da classe professor e sua inserção em um departamento.
   c. **CriaTurma**: Cenário de criação de uma nova turma.
   d. **ImprimeGrade**: Um dado aluno quer saber as informações da sua grade.

e. **SetaNota**: O professor de uma determinada turma está dando as notas finais para seus alunos.

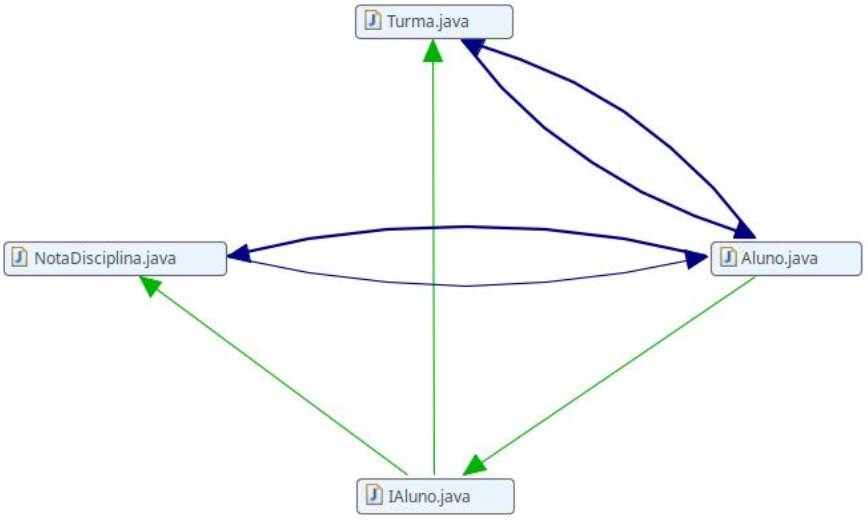
5. Análise com SonarGraph

Dependência de pacotes do framework e aplicação

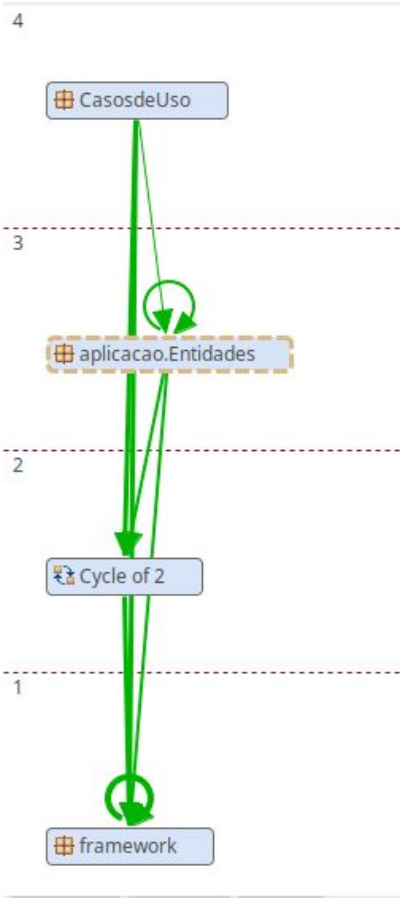


Framework

Diagramas de ciclos presentes no sistema



Aplicação



Nível de manutenibilidade do sistema: *85.49*
Abstração:

- **Framework:** *0.74*
- **Aplicação.Entidades:** *0.33*
- **Aplicação.Usuários:** *0.23*
- **Aplicação.Utilitários:** *0*
- **CasosdeUso:** *0*

Visibilidade:

- **Framework:** *100*
- **Aplicação.Entidades:** *66.67*
- **Aplicação.Usuários:** *69.23*
- **Aplicação.Utilitários:** *100*
- **CasosdeUso:** *100*

Instabilidade:

- **Framework:** *0*
- **Aplicação.Entidades:** *0.75*
- **Aplicação.Usuários:** *0.40*
- **Aplicação.Utilitários:** *0.50*
- **CasosdeUso:** *1.00*

6. Evoluções previstas

Algumas evoluções previstas para o sistema:

1. **Imprimir/Fazer download da grade de um aluno**:

   Como existe uma classe *Grade* que já manipula os dados necessários das disciplinas e notas, uma opção seria criar uma classe responsável por colocar essas informações em um arquivo, .pdf por exemplo, recebendo como entrada uma instância da classe *Grade*.

2. **Dar preferência durante a inscrição em turmas**:

   Implementar um método na classe *Aluno* que fosse responsável por calcular uma métrica, a partir de seus dados, que representasse sua preferência na hora de se inscrever em uma turma.

3. **Laboratórios e grupos de pesquisa**:

   Criar uma classe que representasse um grupo de pesquisa que contivesse um professor responsável e uma lista de alunos.

4. **Tipos diferentes de alunos**:

   Criar classes que herdam da classe *Aluno*, que se especializam no seu tipo, como por exemplo, aluno de graduação, mestrado ou doutorado.

5. **Implementar turmas com mais de uma disciplina**:

   Como no casos de escolas, uma turma pode possuir um ou mais professores de matérias diferentes. Nesse caso seria necessário modificar o atributo singular *Disciplina* na classe *Turma* para uma lista de Disciplina.