MINERVASOLUTIONS

Documento de Arquitetura

Data: 13/10/2018	**Objetivo Estratégico:** Gerenciamento dos dados administrativos do Instituto Ada Lovelace.
ID: 1004	**Nome do Projeto:** sigAda - Sistema de Gerenciamento do Ada
CC: Paulo Pires	**Cliente:** Instituto Ada Lovelace
Patrocinador: UFRJ	**Gerente de Projeto:** Rafael Cardoso

1. Histórico de Versão

Data	Versão	Descrição	Autor(es)
13/10/2018	1.0.0	Abertura do Documento	Eduardo, Rafael e Tainá
14/10/2018	1.1.0	Preenchimento de alguns campos que faltaram	Eduardo, Rafael e Tainá
15/10/2018	1.2.0	Correção do diagrama e caso de uso e da descrição do caso de uso Inserção de links nas imagens	Eduardo, Rafael e Tainá
06/11/2018	1.3.0	Correção do diagrama de caso de classe	Eduardo
15/11/2018	1.4.0	Correção do tópico “Metas e Restrições” Reescrita do subtópico “Descrição dos casos de uso”: ao invés de descrevê-los novamente foi mencionado o documento correspondendo	Tainá
15/11/2018	1.4.1	Correção da coluna “Versão” do histórico de versões Correção da formatação	Tainá, Eduardo, Rafael e Ricardo

		Correção dos dois primeiros subtópicos do tópico “Qualidade”	
16/11/2018	1.5.0	Atualização do diagrama de caso de uso, de classe e do link dos mesmos.	Eduardo
16/11/2018	1.5.1	Atualização do diagrama de pacotes	Tainá
17/11/2018	1.6.0	Atualização do diagrama de classes e o link referente ao mesmo Atualização do tópico “Metas e restrições” Adição do padrão de codificação em “Visão de Implementação”	Tainá

2. Introdução

2.1 Finalidade

Este documento tem como objetivo apresentar de maneira clara e sucinta como se dá a arquitetura do sistema sigAda, de maneira que auxilie no entendimento de como o sistema funciona.

2.2 Escopo

Este documento diz respeito à arquitetura de software do sistema sigAda, que é um sistema de gerenciamento administrativo do Instituto Ada Lovelace de Ensino, desenvolvido pela Minerva Solutions.

2.3 Visão Geral

Os próximos itens tratam de como é organizado a arquitetura do sistema, sendo dividido em:

- Representação da arquitetura: neste tópico é descrito qual a arquitetura utilizada pelo sistema atualmente, bem como ela é representada.
- Visões de casos de uso, lógica e de dados: trata das visões citadas, discutindo respectivamente sobre o modelo de casos de uso, abstrações de partes fundamentais do sistema como objetos ou as classes que representam esses objetos e como os dados são armazenados de maneira persistente.
- Tamanho e desempenho: trata das restrições de desempenho e tamanho do sistema.
- Qualidade: aborda de que maneira as decisões de arquitetura descritas ao longo do documento colabora com a segurança, confiabilidade e usabilidade.

2.4 Referências

- Documento de Especificação de Caso de Uso (ID - 1002)
- Diagrama de Atividade (ID - 1006)
- Diagrama de Sequência (ID - 1007)
- Modelo de Banco de Dados

3. Representação da Arquitetura

A arquitetura utilizada para este sistema será o padrão MVC (Model-View-Controller). Além disso, terá uma camada DAO e um banco de dados local. Como cada camada será implementada e sua interação lógica é melhor explicada no item “Visão Lógica”.

4. Metas e Restrições da Arquitetura

O sistema tem como metas e restrições os itens abaixo:

- O sistema funcionará no sistema operacional Windows 10 e Linux Ubuntu 18.04.1 LTS.
- O sistema deve ser implementado utilizando os ambientes de desenvolvimento integrado Eclipse e/ou Netbeans.
- A linguagem de programação que deve ser utilizada é Java.
- O banco de dados do sistema a ser utilizado é o MySQL.
- As interfaces gráficas farão uso de um design responsivo.
- O manuseamento do sistema não necessita de conexão com a internet.
- O sistema deve estar disponível não só em PCs remotos mas também naqueles oferecidos pelo IALE.
- O sistema deve ser capaz de lidar com tráfego mediano de usuários.
- O acesso ao sistema é feito estritamente através do login e senha, sendo estes intransferíveis.

5. Visão de Casos de Uso

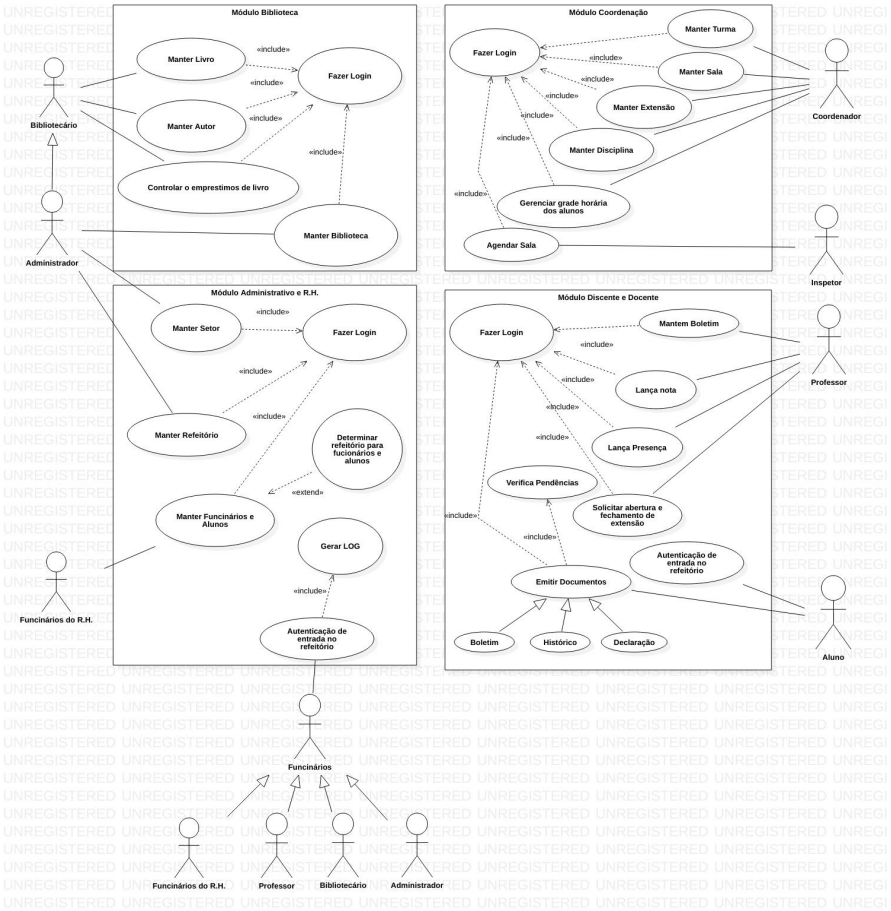
A seguir serão listados e descritos alguns casos de uso que representam funcionalidades centrais/importantes do sistema, bem como os atores que realizam tais casos de uso e sua representação em formato UML.

5.1 Atores

Abaixo é descrito os atores que se relacionam com os casos de uso selecionados, bem como uma breve descrição de sua interação com o sistema:

- **Administrador:** É o funcionário que mantém os dados da biblioteca, setor, refeitório e dos espaços oferecidos pela escola (sala de música, ginásio, etc). Além disso, ele possui um login e senha.
- **Bibliotecário:** É o funcionário que tem acesso e mantém dados dos livros e autores armazenados na biblioteca, possuindo um login e senha para entrar no sigAda. Além disso, é ele que faz o controle do empréstimo de livros na escola.
- **Funcionário do R.H.:** É o funcionário que mantém dos dados dos alunos e funcionários da escola. Também possui um login e senha.
- **Coordenador:** É o funcionário que mantém os dados da turma, sala, extensão e das disciplinas. Ele também é responsável por organizar e gerenciar a grade horária de cada aluno da escola. Possui um login e senha.
- **Inspetor:** É o funcionário responsável por agendar os espaços para os alunos e professores que o solicitarem.
- **Professor:** É o funcionário que mantém o boletim dos alunos para os quais ele leciona. Além disso, também é responsável por lançar a nota e a presença dos alunos no sistema.
- **Aluno:** É a pessoa cadastrada no sistema possuindo o título de aluno, gerando uma matrícula única para cada um. Um aluno pode emitir documentos e participa do processo de autenticação de sua entrada no refeitório designado a ele pelo sistema.

5.2 Diagrama de Casos de Uso



Para visualizar melhor a imagem, clique aqui.

5.3 Descrição dos Casos de Uso

As descrições dos casos de uso podem ser encontradas no Documento de Especificação dos Casos de Uso, cujo link está disponível no início deste documento.

6. Visão Lógica

6.1 Visão geral

Como o sistema utilizará uma arquitetura de padrão MVC, a visão lógica será dividida nos pacotes View, Controller, Model, e o pacote DAO (Data Access Object):

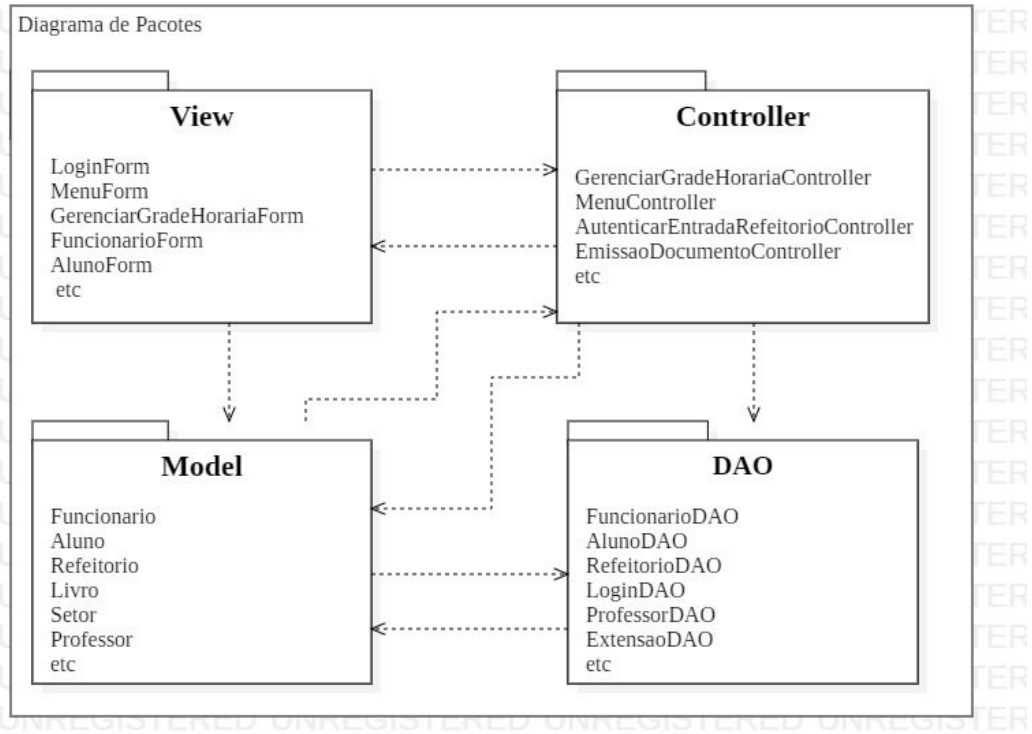
- **View:** Esse pacote terá classe(s) e/ou interface(s) responsável(is) por receber/apresentar os dados para o usuário, além disso, é esse pacote que terá a função de enviar os eventos gerados pelo usuário para o controlador, e devolver para o usuário a resposta do sistema, ou seja, responsável pela interação do sistema com o usuário. Algumas das classes que estarão neste pacote são as telas para login, cadastro de pessoas, etc.

- **Controller:** Neste pacote está(rão) a(s) classe(s) e/ou interface(s) responsável(is) por fazer o gerenciamento da comunicação entre a View, Model e a DAO. A Controller é quem irá receber os eventos enviados pela View, preparar as informações, para então enviá-las para Model, onde o evento será realizado de fato ou um acesso ao banco de dados será feito.

- **Model:** Neste pacote está(rão) a(s) classe(s) que processa(m) as informações enviadas pela Controller, ou seja, é onde são feitos os eventos solicitados pelo usuário. Além disso, caso seja necessário um acesso ao banco de dados, é também nela que tais solicitações ao BD são feitas. Algumas das classes que estarão neste pacotes são as que representam as entidades que compõem o sistema, como Aluno, Professor, Disciplina, etc.

- **DAO:** Este pacote terá a(s) classe(s) responsável(is) por qualquer acesso aos dados armazenados no banco de dados, onde cada classe do domínio tem uma correspondência no pacote DAO.

6.2 Diagrama de pacotes



7. Visão de Processos

7.1 Diagramas de Atividade

Os diagramas de atividade podem ser encontrados no referente à tais diagramas, cujo link para ele está nas referências.

7.2 Diagramas de Sequência

Os diagramas de sequência podem ser encontrados no referente à tais diagramas, cujo link para ele está nas referências.

8. Visão de Implementação

8.1 Visão geral

O sistema pode ser dividido em 4 camadas: View, Controller, Model e a camada DAO (*Data Access Object*). As três primeiras são as típicas do padrão arquitetural MVC: a camada View é responsável por toda a interação usuário-sistema, ou seja, a interface gráfica do sistema; já a Controller tem como função estabelecer a comunicação entre as camadas View, Model e DAO, sendo ela quem recebe as solicitações de ações da View, reformulando-as e as

enviando para a Model; por fim, a camada Model é responsável pela realização das ações enviadas pela Controlle e, além disso, é ela que faz os pedidos de acesso ao banco de dados. Por último, a camada DAO será aquela responsável pela conexão e acesso de fato ao banco de dados local utilizado nesse sistema, cujo esquema está representado pelo diagrama Entidade-Relacionamento no item 7.3.

8.2 Padrão de codificação

O padrão de codificação utilizado será parcialmente aquele que já é disponibilizado e recomendado pela própria linguagem Java.

Organização geral dos arquivos

1. Declarações de pacotes
2. Instruções de importação
3. Declaração de classes/interfaces
4. Variáveis estáticas
5. Variáveis constantes
6. Construtores
7. Declaração de métodos
8. Variáveis

- **Declaração de pacotes**
    - Deve seguir o seguinte padrão: com.ms.sigada.<nomedopacote>.
    - <nomedopacote> deve estar com todas as suas letras em minúsculo e não pode conter ‘_’ ou ‘$’.

- **Declaração de classes/ interfaces**
    - Devem ser utilizados substantivos para nomear as classes/interfaces, onde a letra inicial de cada palavra utilizada deve estar em maiúsculo.
    - A chave de abertura “{” deve aparecer na mesma linha da declaração da classe.
    - Evitar abreviações e acrônimos.
    - Ex.: Funcionario.java, AutenticacaoEntrada.java

- **Variáveis constantes**
    - Todos os caracteres utilizados devem estar em maiúsculo e caso possua mais de uma palavra, devem ser separadas por ‘_’.
    - Deve-se rotular tais variáveis como estática e final.

- Ex.: MAX_WIDTH

- **Declaração de métodos**

  - Excluindo a primeira palavra, as outras devem possuir sua primeira letra em maiúsculo.
  - Evitar colocar como primeira palavra algo diferente de verbos.
  - Não pode haver espaços entre o nome do método e o primeiro parêntesis.
  - A chave de abertura "{" deve aparecer na mesma linha da declaração do método.
  - Ex.: setNome, alteraCampo

- **Variáveis**

  - Seus nomes devem representar com clareza suas funções e devem possuir um tamanho tão pequeno quanto possível.
  - Excluindo a primeira palavra, as outras devem possuir sua primeira letra em maiúsculo, caso exista.
  - Não podem possuir '_' ou '$'.
  - Variáveis com somente um caractere devem ser evitadas, salvo aquelas que são temporárias.
  - Algumas sugestões para variáveis inteiras são: "i", "j", "k", "m".
  - Ex.: nome, dataDeNascimento, diaDaSemana

Codificação das camadas

- **View**
  - As classes que estão na camada View possuem o seguinte padrão de nomenclatura: <nome>Form.java, onde cada palavra de <nome> deve ter sua primeira letra em maiúsculo.
  - Não pode ser utilizado caracteres especiais em <nome>.

- **Controller**
  - As classes que estão na camada Controller possuem o seguinte padrão de nomenclatura: <nome>Controller.java, onde cada palavra de <nome> deve ter sua primeira letra em maiúsculo.
  - Não pode ser utilizado caracteres especiais em <nome>.

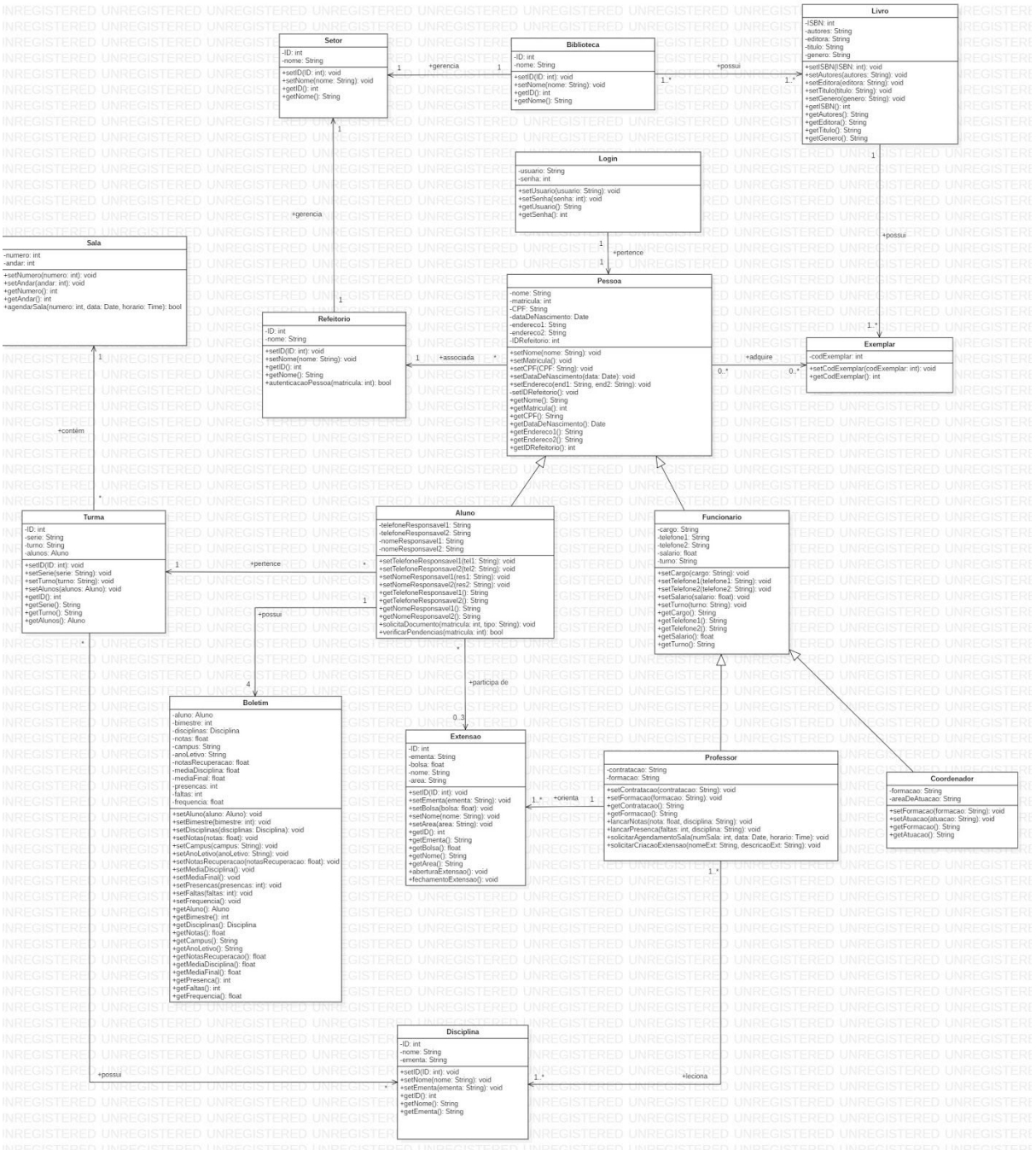
- **Model**
  - As classes da camada Model não possui um padrão além do fato de que toda palavra que forma o nome das classes/interfaces devem ter suas primeiras letras em maiúsculo.

- Não pode ser utilizado caracteres especiais em <nome>.

- **DAO**

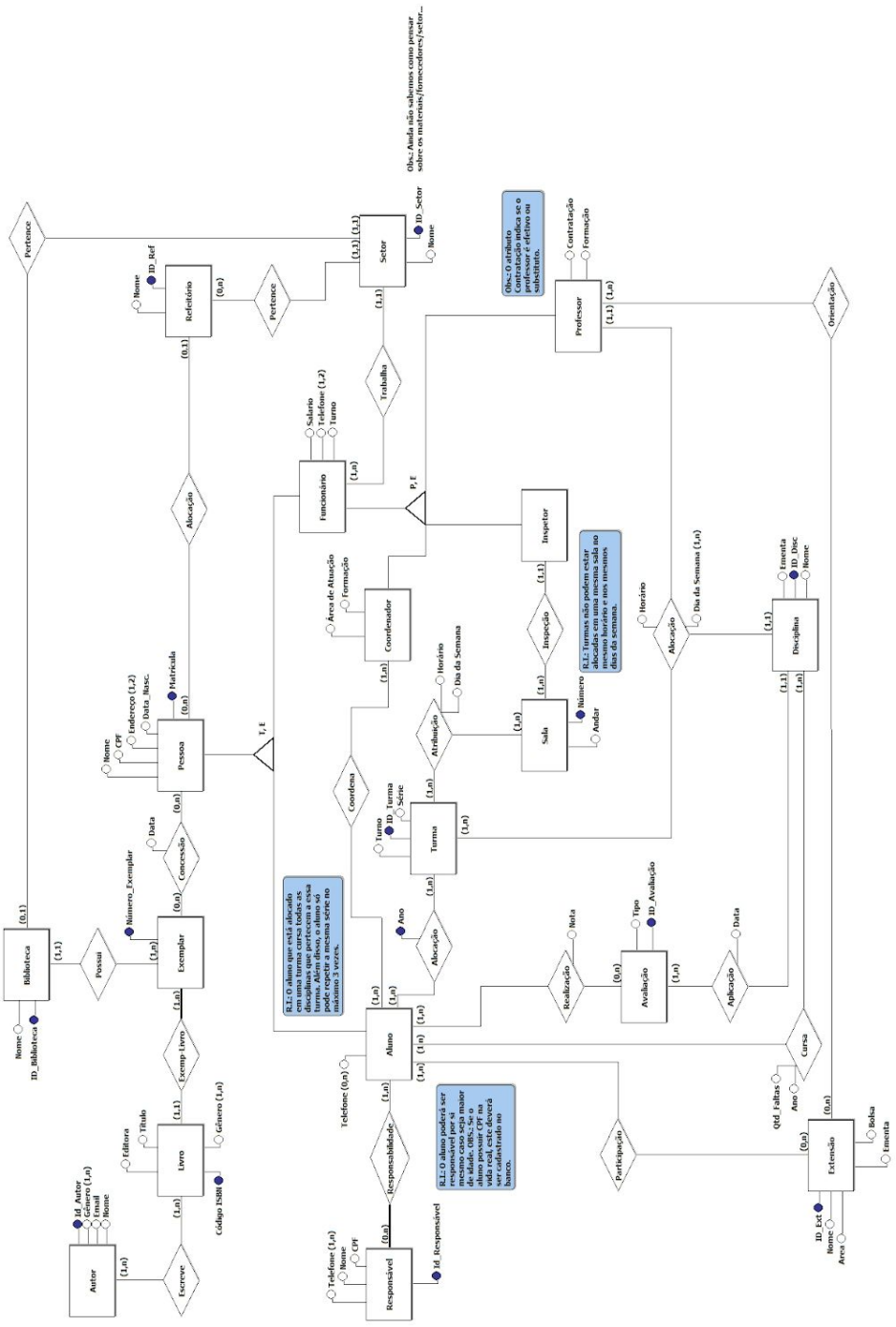
  - As classes que estão na camada DAO possuem o seguinte padrão de nomenclatura: <nome>DAO.java, onde cada palavra de <nome> deve ter sua primeira letra em maiúsculo.
  - Não pode ser utilizado caracteres especiais em <nome>.

8.3 Diagrama de Classes



Para melhor visualização da imagem, clique aqui.

8.4 Diagrama de Entidade-Relacionamento



Para visualizar a imagem melhor, clique [aqui]

12. Tamanho e Desempenho

O sistema deve suportar até 10 usuários utilizando simultaneamente o banco de dados. Além disso, deve ser capaz de realizar as tarefas em menos de 2 minutos. A parte cliente requer menos de 200 MB de espaço em disco e 3 GB de RAM.

13. Qualidade

O sistema conforme descrito nos itens acima, garante confiabilidade, segurança e usabilidade, uma vez que:

- Todos os dados armazenados estarão num banco de dados. Como consequência disso, será utilizado a ferramenta view, disponibilizada pelo BD, de maneira que acessos à certas áreas do banco de dados só possam ser feitas por certos usuários.
- O sistema será projetado de maneira mais intuitivo possível, de maneira que não seja necessário treinamentos para além da leitura do "Manual de Introdução ao sigAda" e que este seja acessado a menor quantidade de vezes possível.
- No próprio sistema haverá disponível uma área de instruções para o usuário que seja simples, claro e direto, a fim de ajudá-lo a utilizar o software.
- Como tanto o sistema, quanto o banco de dados possui uma senha e login, garante-se um nível considerável de privacidade.