UNIP EaD
Projeto Integrado Multidisciplinar VI
Cursos Superiores de Tecnologia

# PROJETO DE UM SISTEMA DE CONTROLE DE MATRÍCULAS DE CURSOS LIVRES.

Polo Sorocaba Campolim
2019

UNIP EaD

Projeto Integrado Multidisciplinar VI

Cursos Superiores de Tecnologia

# PROJETO DE UM SISTEMA DE CONTROLE DE MATRÍCULAS DE CURSOS LIVRES.

Nobuyoshi Ishizuka

Paulo Alves dos Santos Junior

RA(s): 0501648

RA(s): 1868289

Curso: Análise e Desenvolvimento de Sistemas

Polo Sorocaba Campolim

2019

**RESUMO**

A finalidade desse projeto é criar um sistema de Controle de Matrículas de Cursos Livres para realizar o cadastro de alunos, cursos e matrícula dos usuários para cursos de curta duração, utilizando as técnicas e ferramentas de Projetos de Sistemas Orientado a Objetos, compreendendo  e definindo o contexto e o modo de utilização do sistema, projetando a arquitetura do sistema respeitando os requisitos que foram levantados na fase de levantamento de requisitos com o cliente. utilizando o padrão MVC (MODEL-VIEW-CONTROLER), para o desenho da arquitetura de referência e através dos diagramas da UML (Unified-Modeling-Language), identificaremos os principais objetivos do sistema descrevendo os modelos de projeto que serão representados por casos de uso e diagramas de classe de implementação, diagrama de sequência de implementação, diagrama de atividades e  diagrama de distribuição com os requisitos para implementação do sistema que será desenvolvido.

**Palavras-chave:** Sistema de Controle de Cursos Livres. Projeto de Sistemas Orientado a Objetos. UML. M

**ABSTRACT**

The purpose of this project is to create a Free Course Enrollment Control system to register students, courses and user enrollment for short courses using Object Oriented Systems Design techniques and tools, understanding and defining the context and how to use the system, designing the system architecture respecting the requirements that were raised in the requirements gathering phase with the client. Using the MVC (MODEL-VIEW-CONTROLER) standard for reference architecture design and through the Unified-Modeling-Language (UML) diagrams, we will identify the main objectives of the system by describing the design models that will be represented by case studies. use and implementation class diagrams, implementation sequence diagram, activity diagram, and distribution diagram with the requirements for implementing the system to be developed.

**Keywords:** Free Course Control System. Object Oriented Systems Design. UML. MVC.

**SUMÁRIO**

# 1 INTRODUÇÃO

A fase de projeto de software tem por finalidade definir e especificar uma solução a ser implementada, é a fase a ser iniciada logo após o levantamento de requisitos do sistema com o cliente, que definirá as tarefas que se espera que sistema execute e seu correto funcionamento de acordo com as necessidades do usuário. É uma fase de tomada de decisões, tendo em vista que muitas soluções são possíveis nesta fase. Além disso, o projeto é um processo de refinamento. Inicia-se com o projeto de arquitetura do sistema, que visa descrever a estrutura de nível mais alto da aplicação, identificando seus principais elementos ou componentes e como eles se relacionam uns com os outros.

Segundo Pressman (2006), para definir a arquitetura, o projeto se concentra no detalhamento de cada elemento, até atingir o nível de unidades de implementação. O projeto portanto, é a fase do processo de software na qual os requisitos, e as necessidades do negócio e as considerações técnicas se juntam para a criação de um produto ou sistema de software

Este projeto tem a finalidade de elaborar a documentação da fase de projetos de um sistema de controle de matrículas de cursos livres, visando analisar e demonstrar de forma conceitual os artefatos solicitados para este projeto.

# 2 REFERENCIAL TEÓRICO

## 2.1 O Padrão MVC (Model-View-Controller)

O MVC (Model, View, Controler) é largamente utilizado em projetos de desenvolvimento de sistemas devido à arquitetura que possui, que possibilita a divisão do projeto em três camadas muito bem definidas. Sendo elas a camada Model, o Controller e a View. Segundo o site DEVMEDIA o conceito principal do modelo MVC é utilizar uma solução já definida para separar partes distintas do projeto reduzindo suas dependências ao máximo.

A camada Controller ou Controle Interpreta as entradas do mouse ou do teclado enviados pelo usuário do sistema e mapeia essas ações do usuário em comando que serão enviados para a camada Model, sendo responsável pela intermediação entre as camadas Model e View.

A camada Model ou Modelo é a principal camada computacional da arquitetura que contém a lógica da aplicação, que gerência um ou mais elementos de dados. Sendo responsável pelas regras de negócio, representando a informação (dados) dos formulários e as regras SQL para manipular dados do banco de dados, obtendo e convertendo os dados de maneira que tenham conceitos significativos em sua aplicação, assim como processar, validar, associar e outras funções relativas ao tratamento dos dados.

A camada View ou Visão é responsável pela representação (Visão) dos dados presentes no modelo solicitado, fazendo a exibição dos dados, podem ser apresentados em diversos formatos, dependendo do tipo da aplicação, como arquivos HTML, XML vídeos entre outros.

## 2.2 Unified Modeling Language – UML

Segundo o Livro texto de Engenharia de Software I da Unip Interativa a UML foi criada independentemente do processo de software, porém os desenvolvedores podem adotar a UML aos seus projetos e processos, usando-a para registar os resultados de suas decisões de análise e designe.

A finalidade da UML é proporcionar um padrão para especificação e arquitetura de projetos de sistemas, desde os aspectos conceituais até as soluções concretas, tais como classes de objetos componentes de software reusáveis. (JACOBSON; BOOCH; RUMBAUGH, 1999).

É uma linguagem de modelagem que representa um sistema de forma padronizada, com o objetivo de facilitar a compreensão da pré-implementação, a UML permite que desenvolvedores visualizem os produtos de seus trabalhos através do uso de diagramas padronizados, que nos permitirão desenhar o Software antes de começarmos a codificar o software. Trata-se de uma linguagem unificada que habilita os profissionais da área da Tecnologia da Informação a modelar e documentar aplicações de Software.

# 3 SISTEMAS DE CONTROLE DE MATRÍCULAS DE CURSOS LIVRES

## 3.1 Requisitos do Sistema de Controle de Matrículas

O Sistema será utilizado por atendentes e alunos matriculados nos cursos disponíveis que são o curso de informática e o curso de artes. Todo o acesso será feito em terminais na escola por meio de login e senha.

Segundo os requisitos acordados com cliente que solicitou o desenvolvimento do sistema o atendente poderá realizar as seguintes ações:

- Manter Cursos: o atendente cadastra os cursos que abrangem duas áreas diferentes: informática e artes. Todos os tipos de cursos possuem código, nome, data de início, data de término, horário, número de vagas e valor. Para os cursos de informática, há também o número do laboratório e o registro dos softwares que serão utilizados e, para o curso de artes, a descrição do material e nome dos livros que serão utilizados. Manter Alunos: o atendente cadastra os alunos, informando: nome, endereço, telefone, e-mail, RG, CPF, login e senha do aluno.
- Cadastrar Matrícula: o atendente cadastra a matrícula do aluno em um ou mais cursos. É gerado um código de matrícula, a data da matrícula, o valor da matrícula, o status de pagamento e o status da matrícula. Após o cadastro da

matrícula, os dados da matrícula (código matrícula) são enviados para o Sistema Financeiro.

- Gerar relatórios de matrículas: o atendente emite relatórios em tela da quantidade de matrículas por curso em um determinado período.

O aluno poderá realizar as seguintes ações:

- Consultar Cursos: o aluno consulta informações dos cursos disponíveis. 25 Serviço Social
- Consultar Matrículas: o aluno consulta matrículas de cursos que já realizou ou está realizando.

Regras:

1. Caso o aluno já tenha realizado outros cursos, terá desconto progressivo. Um curso, desconto de 5%; dois cursos, desconto de 10%; mais de dois cursos, desconto de 15%.

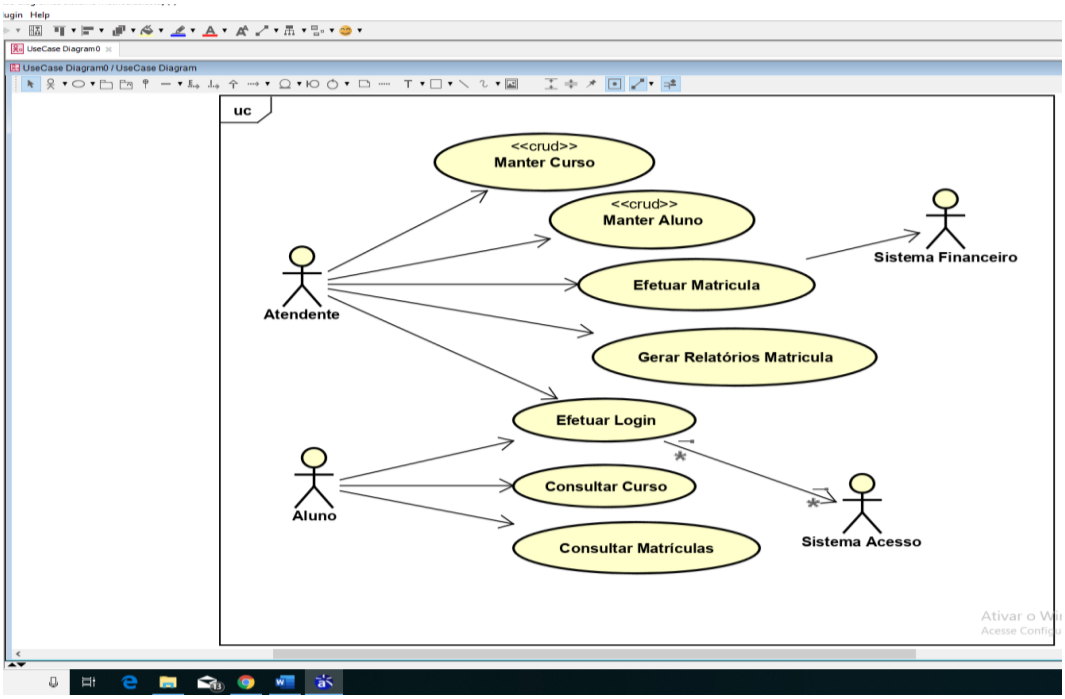
2. Toda parte de cobrança do curso é controlada pelo Sistema Financeiro, o qual somente recebe as informações do cadastro de matrícula.

3. Um aluno matriculado pode realizar vários cursos.

4. Devem ser exibidas mensagens de advertência para todas as ações malsucedidas.

Na figura-1 abaixo representaremos por um diagrama de caso de uso o papel de cada autor no sistema de controle de matrículas onde a atendente é responsável por cadastrar cursos, alunos, efetuar matrículas, gerar relatórios. O Aluno poderá consultar informações sobre o curso e matrícula. Podemos observar que para utilizar o sistema o atendente e o aluno devem efetuar o login que acessa outro ator externo, o sistema de acesso que através da validação do login e senha liberará o acesso ao usuário e por fim a atendente para efetuar a matrícula de um aluno deverá acessar a função Efetuar Matricula que enviará a o sistema financeiro os dados e código da matrícula.

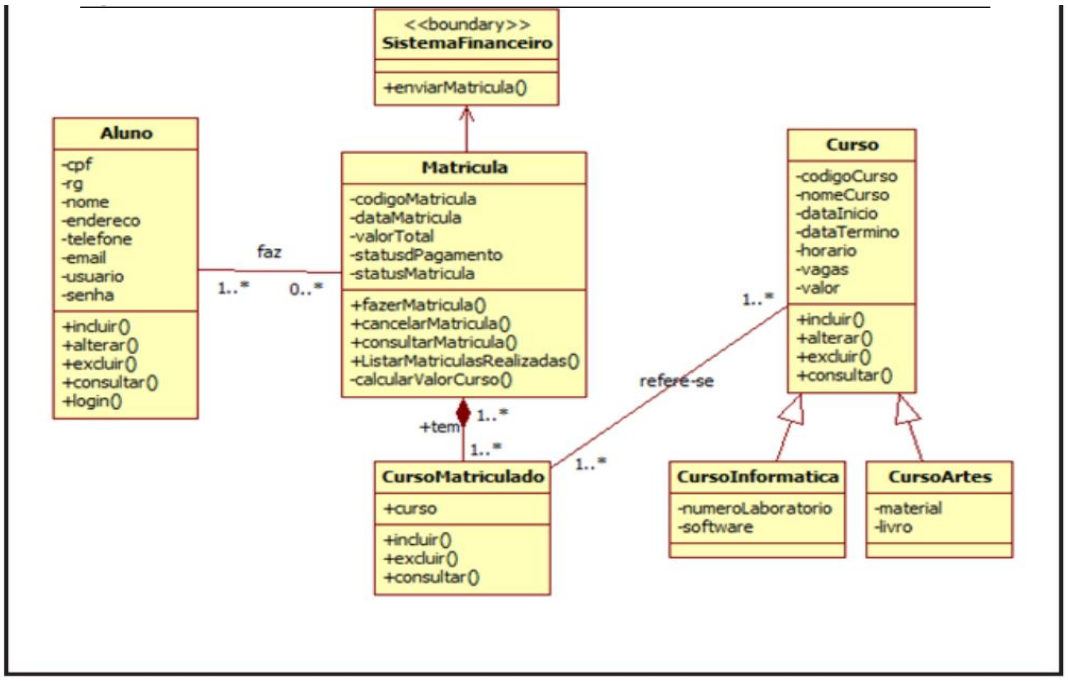
Figura 1 - Caso de Uso do Sistema de Controle de Matrículas




Fonte: O próprio Autor

Com base nos requisitos dos atributos e métodos do sistema foi elaborado um diagrama de classes especificando o modelo de análise que o projeto deverá seguir.

*Figura 2-Diagrama de Classes de análise*

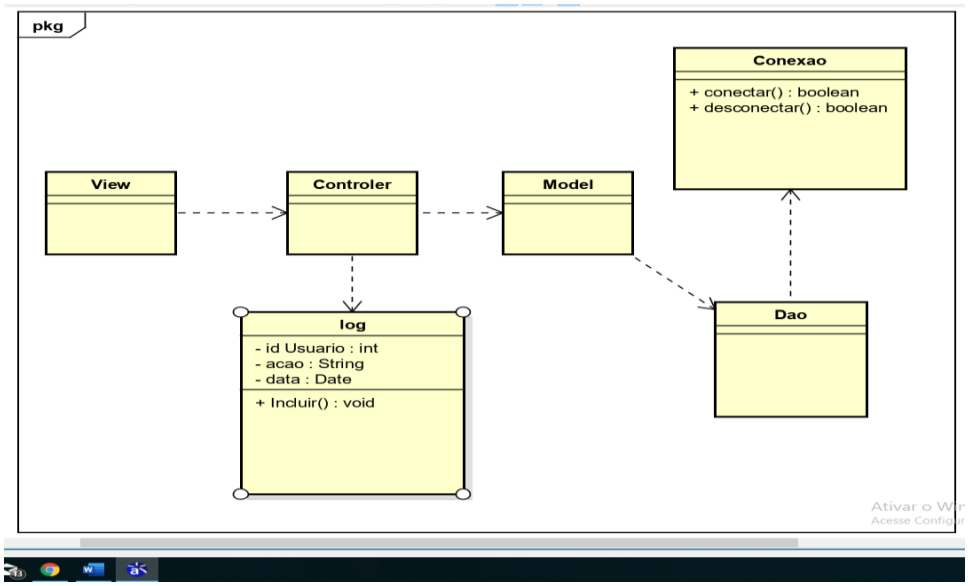



Fonte: O próprio Autor

## 3.2 Arquitetura MVC

Na figura-3, abaixo representamos o desenho da arquitetura MVC do sistema de controle de Matriculas, Neste tipo de arquitetura de Software a camada View (Visão) interage com a camada Controller (Controle), esta camada faz o acesso a camada modelo e insere um registro no log, a camada Model interage com a DAO, e estabelece a conexão com o banco de dados do sistema em questão.

Figure 3 - Desenho da arquitetura MVC




Fonte: O próprio Autor

## 3.3 Modelo Entidade Relacionamento (MER) e Diagrama Entidade Relacionamento (DER)

Segundo o livro texto de Administração de Banco de Dados escrito pelos Autores Pinto e Santos (2012, p.36), O MER foi desenvolvido pelo Prof. Peter Chen para a representação das estruturas de dados de uma maneira mais próxima do mundo real dos negócios. Seus aspectos estruturais formalizam o modo como os dados estão organizados no modelo relacional de banco de dados.

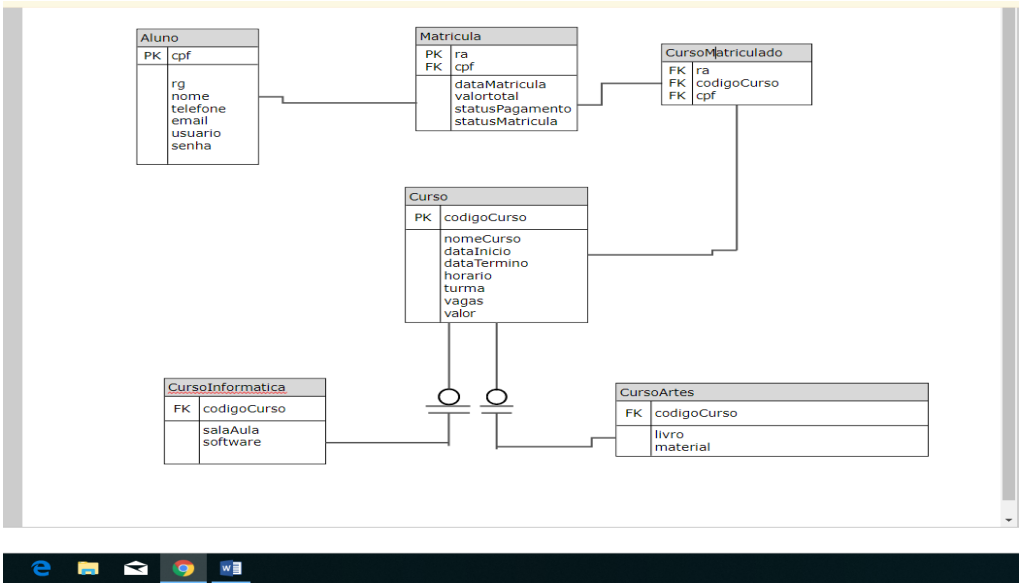
O Modelo Entidade Relacionamento é constituído pelos seguintes objetos:

- Entidades;
- Atributos;
- Relacionamentos.

O DER (Diagrama Entidade relacionamento), é a representação gráfica de um MER (Modelo Entidade-Relacionamento), sendo utilizado para a facilitação da comunicação entre os integrantes da equipe por oferecer uma linguagem utilizada por analistas, que são responsáveis por levantar os requisitos do sistema, e os desenvolvedores, responsáveis por implementar aquilo que foi modelado.

Na figura-4 abaixo foi representado um Diagrama Entidade-Relacionamento.

Figura 4 – Diagrama Entidade-Relacionamento




Fonte: O próprio Autor

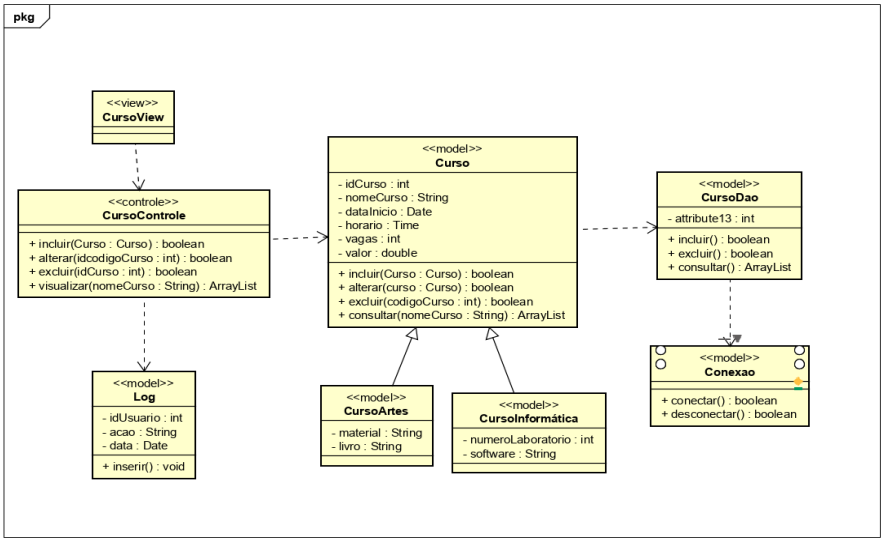
## 4 DIAGRAMAS DE CLASSE DE IMPLEMENTAÇÃO

Os diagramas de implementação modelam a arquitetura física de um sistema, mostrando os relacionamentos entre os componentes de software de hardware no sistema e a distribuição física no sistema.

Apresentaremos a seguir todos os diagramas de classe de implementação que serão elaborados com base no diagrama de casos de uso que foi desenvolvido na fase de análise de projeto e o desenho da arquitetura MVC que foi apresentado na figura-3 deste projeto.

## 4.1 Manter Curso

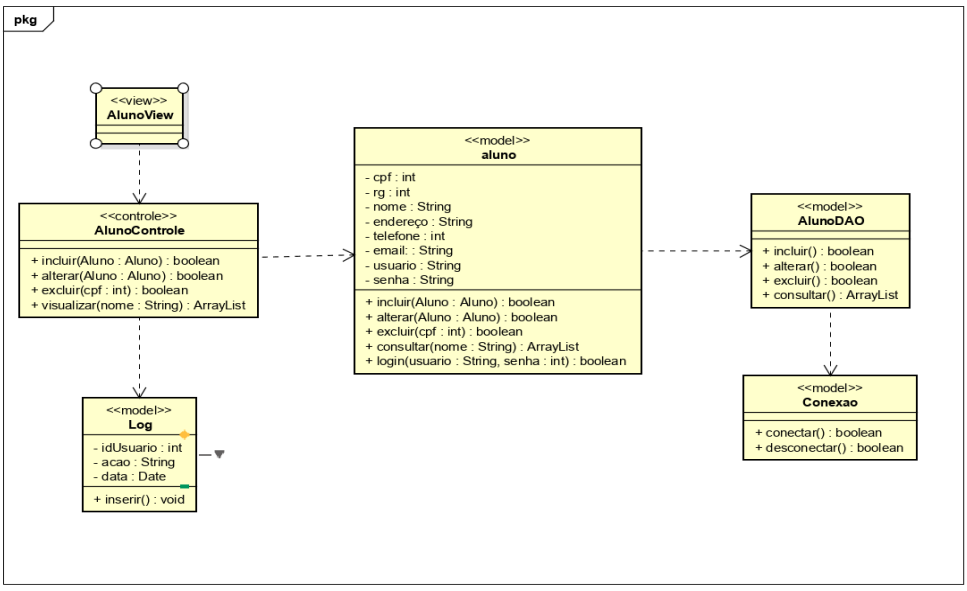
Figura -5 Diagrama de classe de implementação Manter Curso




Fonte: O próprio Autor

## 4.2 Manter aluno

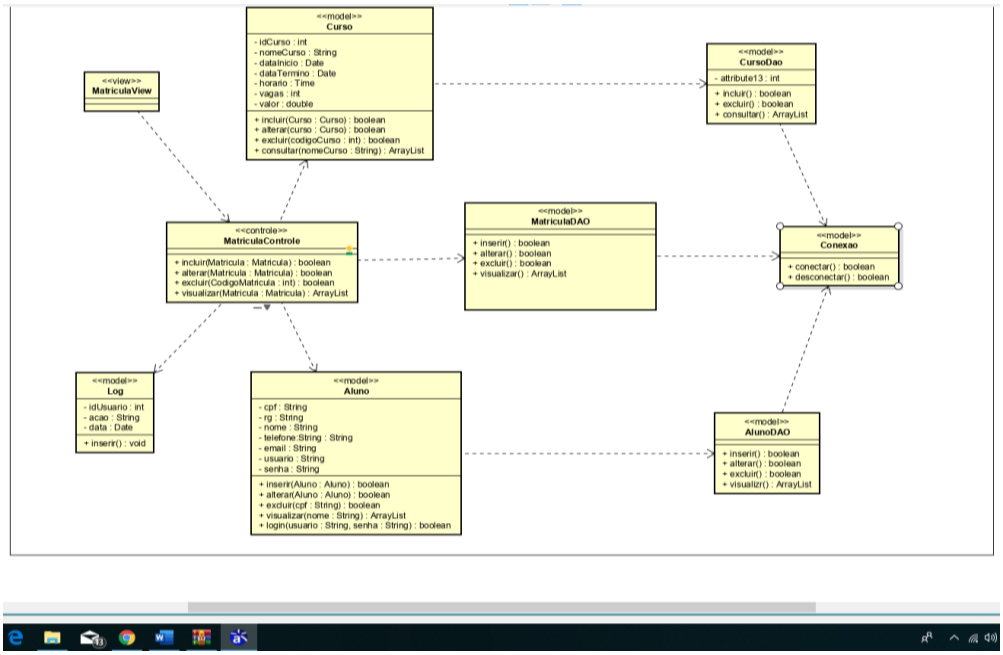
Figura 6 -Diagrama de classe de implementação Manter Aluno




Fonte: O próprio Autor

## 4.3 Efetuar Matrícula

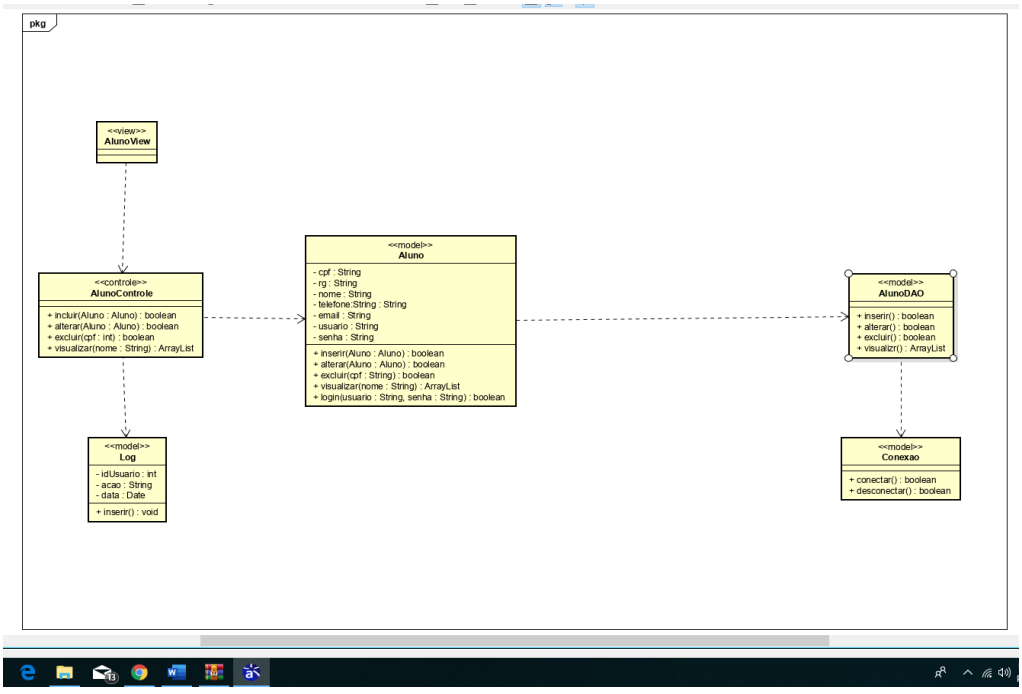
Figura 7- Diagrama de classe de Implementação Efetuar Matrícula




Fonte: O próprio Autor

## 4.4 Gerar Relatório da Matrícula

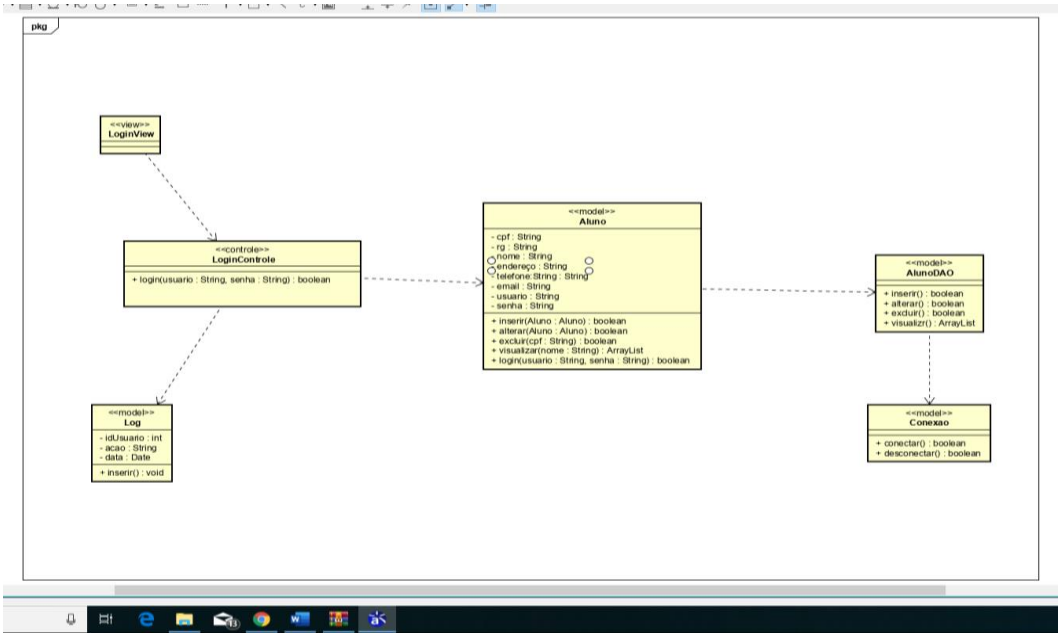
Figura 8 – Diagrama de classe de implementação Gerar Relatório Matrícula




Fonte: O próprio Autor

## 4.5 Efetuar Login

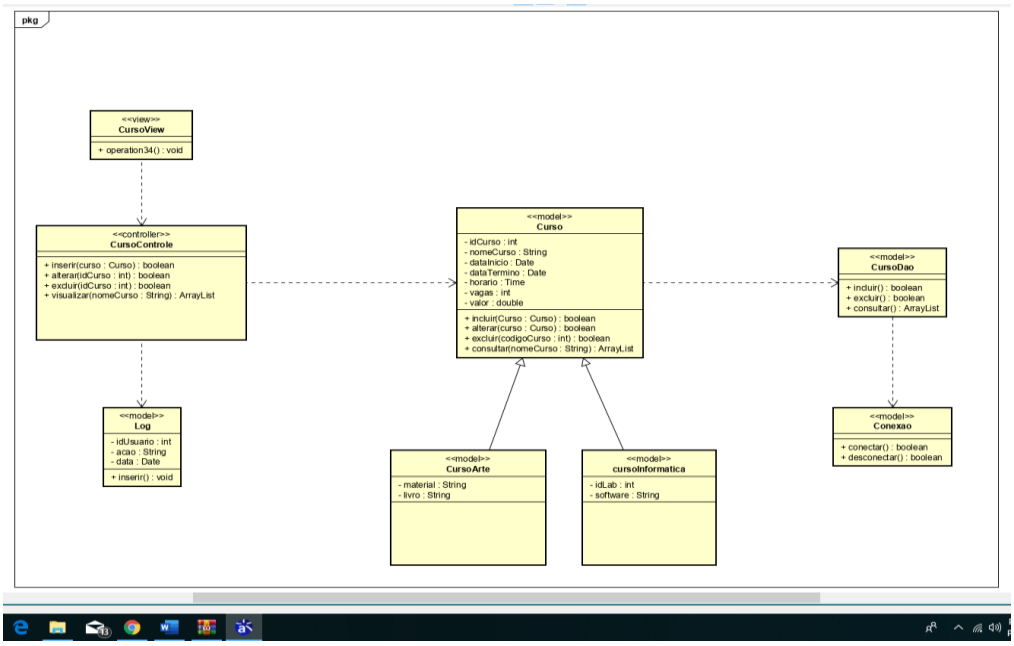
Figura 9 - Diagrama de classe de implementação Efetuar Login




Fonte: O próprio Autor

## 4.6 Consultar Curso

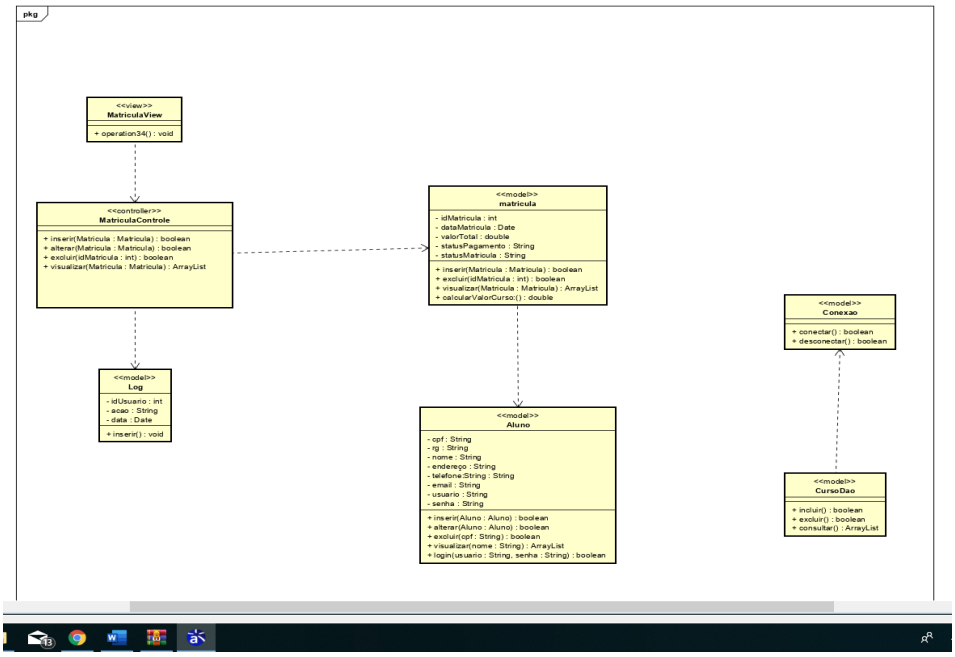
Figura 10- Diagramas de classe de implementação Consultar Curso




Fonte: O próprio Autor

## 4.7 Consultar Matrícula

Figura 11 – Diagrama de Classe de Implementação Consultar Matrícula




Fonte: O próprio Auto

## 5 DIAGRAMAS DE SEQUÊNCIA DE IMPLEMENTAÇÃO

O diagrama de sequência busca determinar a sequência de eventos e troca de mensagens entre vários objetos em um determinando contexto (caso de uso, operação etc.), demonstrando quais operações devem ser disparadas entre os objetos envolvidos e em qual ordem para a realização desse contexto.

Utilizado para modelar os aspectos dinâmicos do sistema, tais como:

- Interação entre objetos dentro de um determinado processo (Caso de uso);
- As mensagens são trocadas entre esses objetos;
- Os métodos que são chamados entre esses objetos;
- A ordem em que ocorrem os eventos.

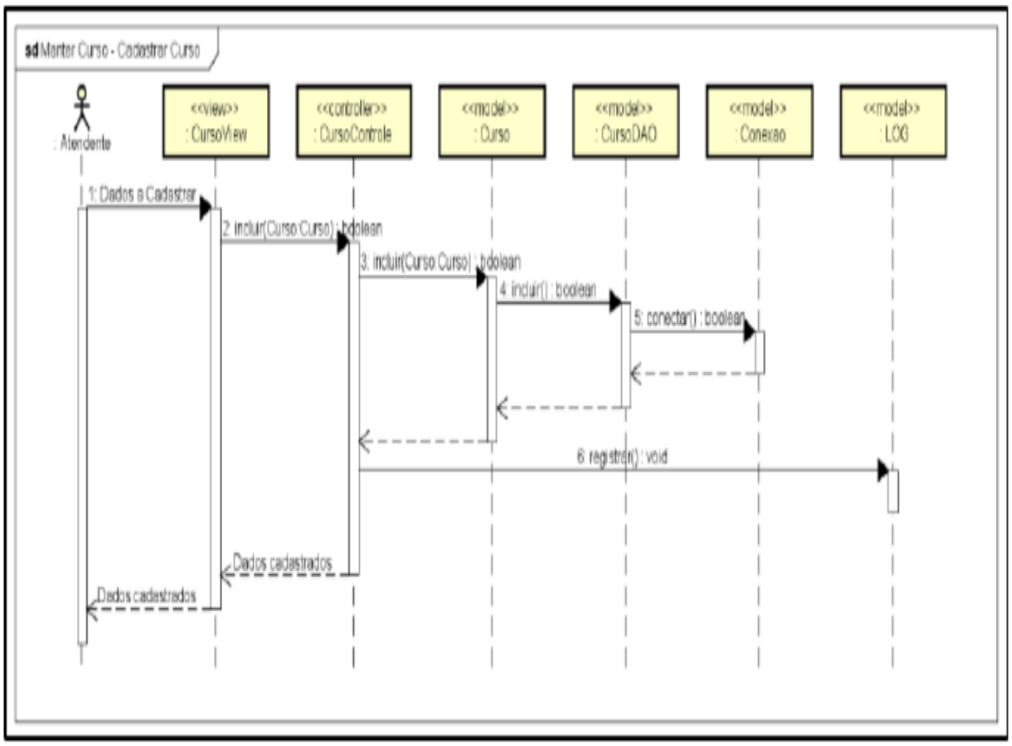
No diagrama de sequência a representação do tempo de vida de um objeto (linha de vida ou lifeline) é realizada por linhas verticais, essas linhas são preenchidas por barras verticais que indicam exatamente quando exatamente o objeto passou a existir e quando esse objeto deixou de existir é acionado um “X” a parte inferior da

linha de vida. As linhas horizontais representam as mensagens trocadas entre os objetos, acompanhadas com um rótulo contendo o nome da mensagem, e opcionalmente, os parâmetros, linhas horizontais tracejadas representam os retornos das mensagens.

A seguir iremos apresentar os diagramas de sequência de implementação que foram desenvolvidos baseado no diagrama de classes e o diagrama de caso de uso que foram apresentados na fase de análise de projeto. Será elaborado um diagrama de sequência de implementação especifico par cada caso de uso que foram desenhados na fase de análise de projeto, utilizando a arquitetura MVC tendo como base os requisitos apresentados no diagrama de caso de uso na figura-1, para os casos de uso do tipo “clud” (incluir, alterar, consultar e excluir) realizaremos um diagrama de sequência para cada uma dessas atividades.

## 5.1 Diagrama de Sequência de Implementação – Manter Curso (Cadastrar Aluno)

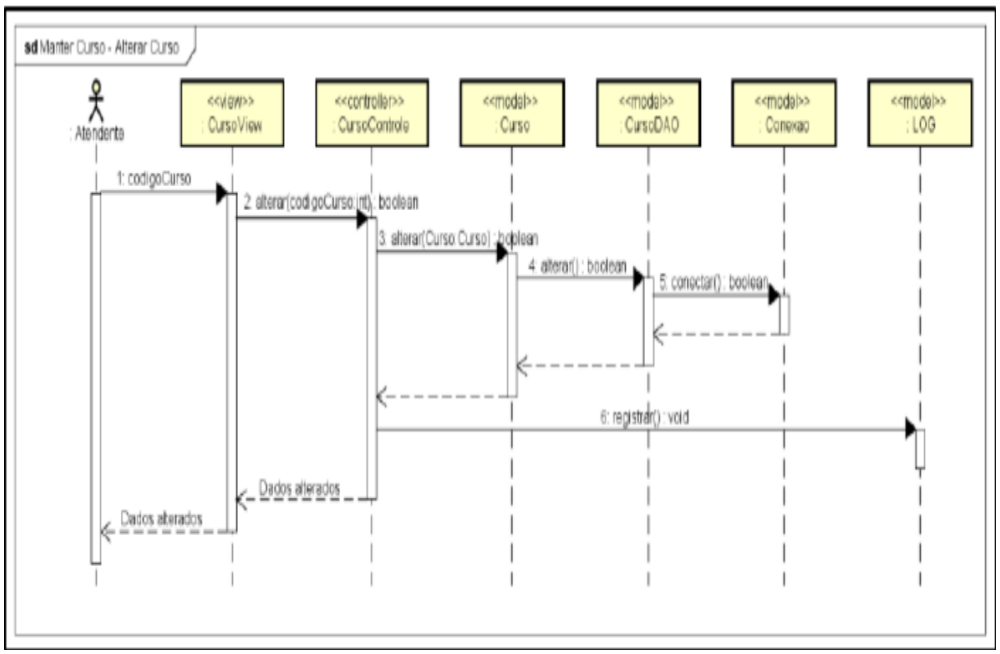
Figura 12 – Diagrama de Sequência de Implementação - Cadastar Alun




Fonte: O próprio Autor

## 5.2 Diagrama de Sequência de Implementação – Manter Curso (Alterar Curso).

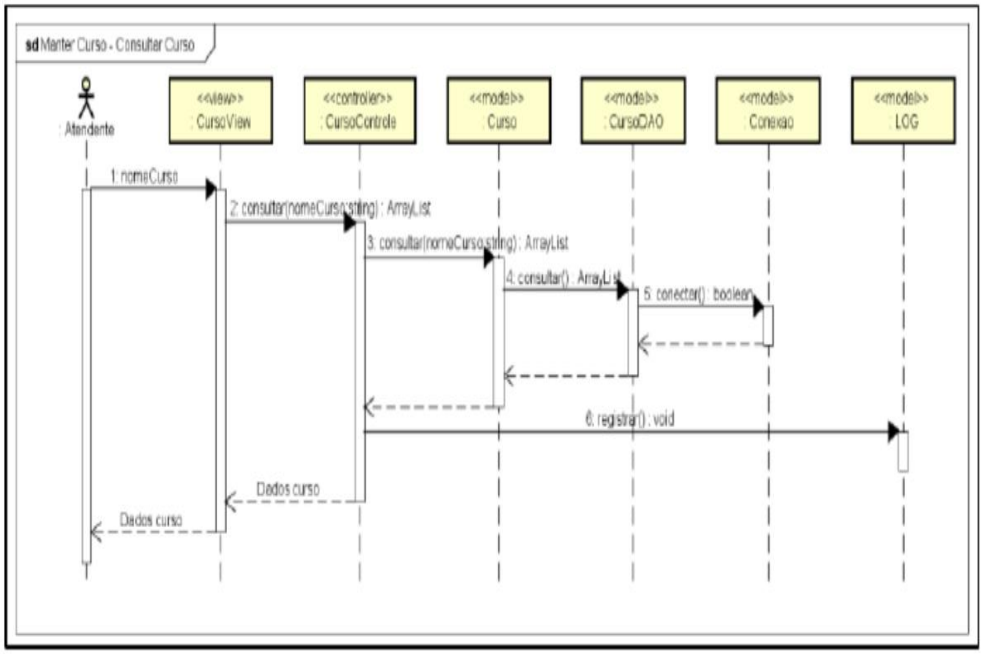
Figura 13 – Diagrama de Sequência Implementação - Alterar Aluno




Fonte: O próprio Autor

## 5.3 Diagrama de Sequência de Implementação – Manter Curso (Excluir Curso)

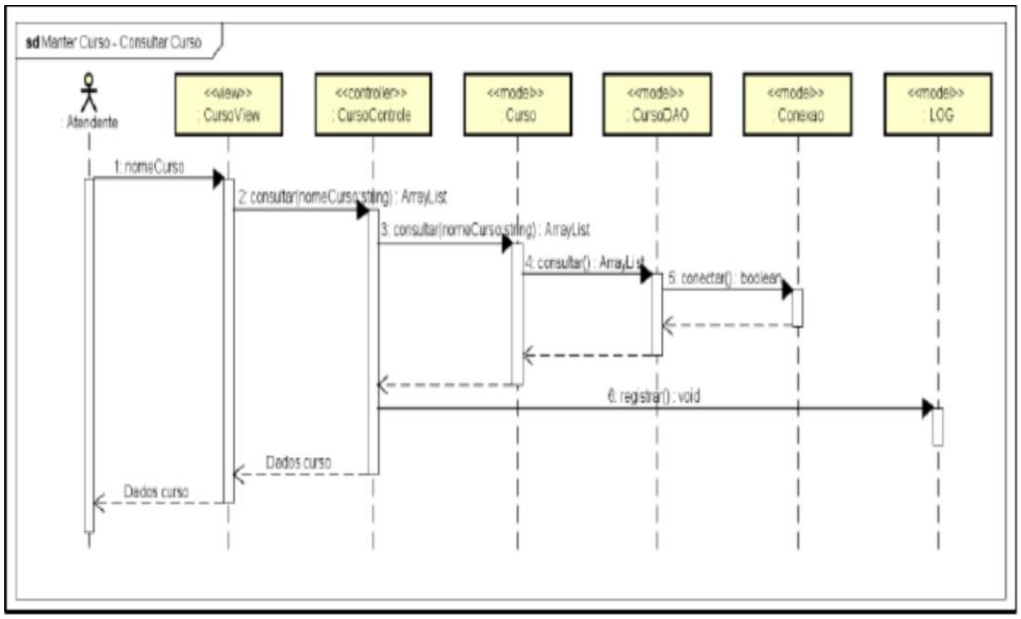
Figura 14 – Diagrama de Sequência de Implementação – Excluir Aluno




Fonte: O próprio Autor

## 5.4 Diagrama de Sequência de Implementação – Manter Curso (Consultar Curso).

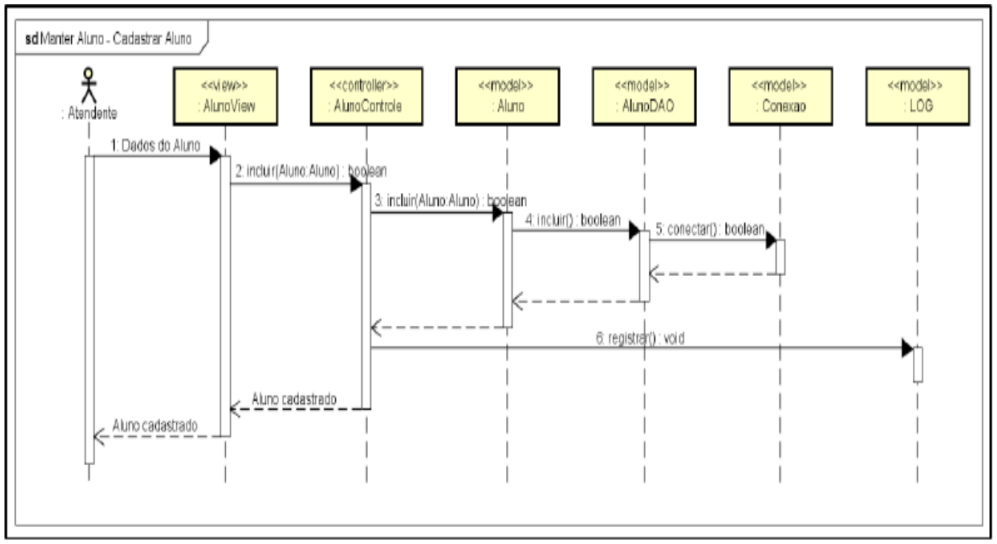
Figura 15 – Diagrama de Sequência de Implementação – Consultar Curso




Fonte: O próprio Autor

## 5.5 Diagramas de Sequência de Implementação – Manter Aluno (Cadastrar Aluno)

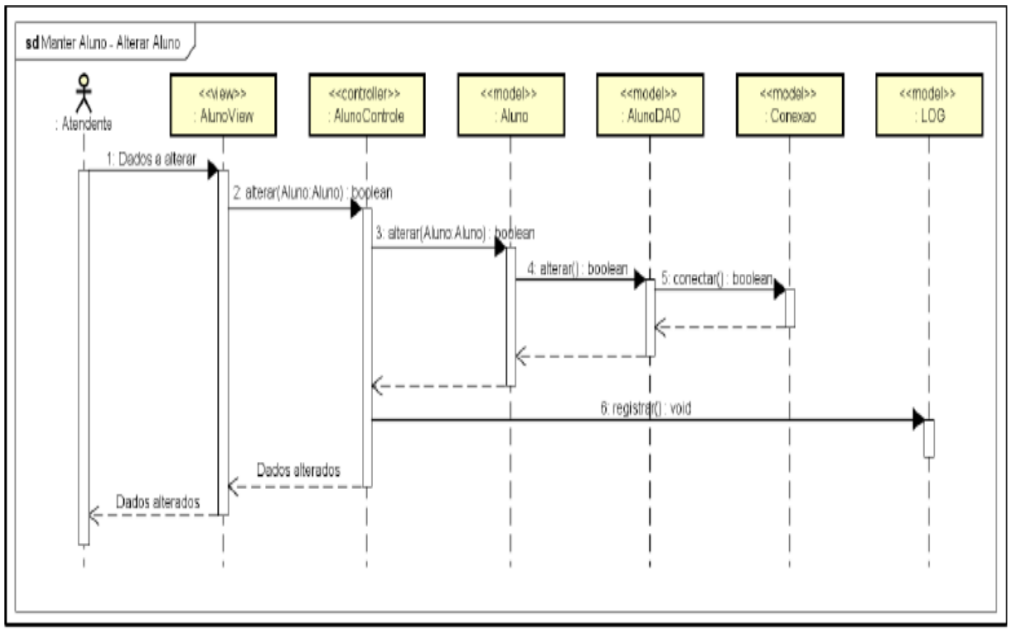
Figura 14 – Diagrama de Sequência de Implementação – Cadastrar Aluno




Fonte: O próprio Autor

## 5.6 Diagramas de Sequência de Implementação – Manter Aluno (Alterar Aluno)

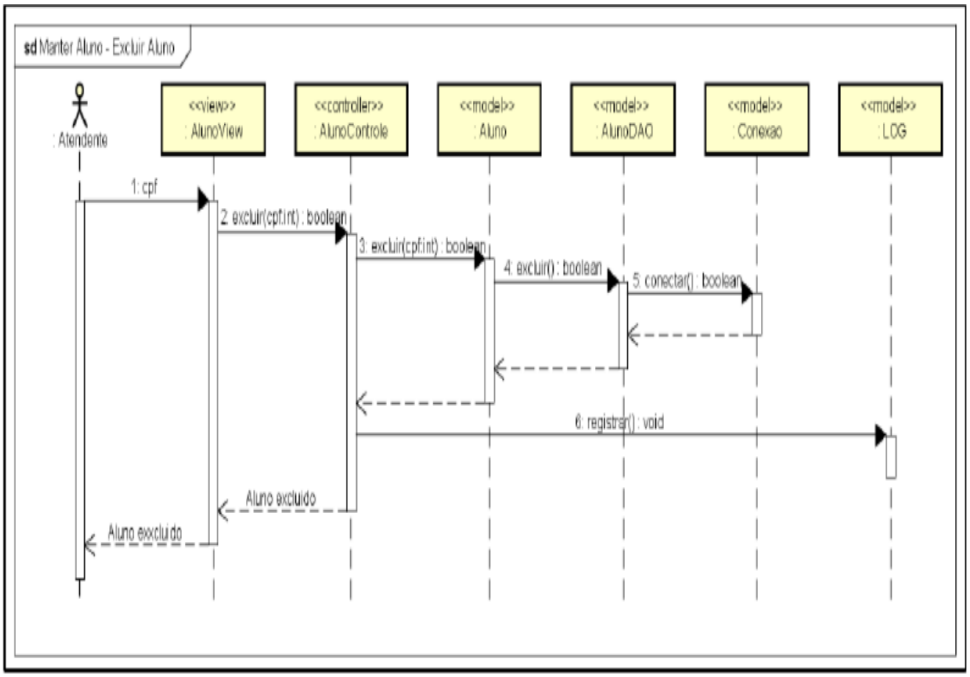
Figura 14 – Diagrama de Sequência de Implementação – Alterar Aluno




Fonte: O próprio Autor

## 5.7 Diagramas de Sequência de Implementação – Manter Aluno (Excluir Aluno)

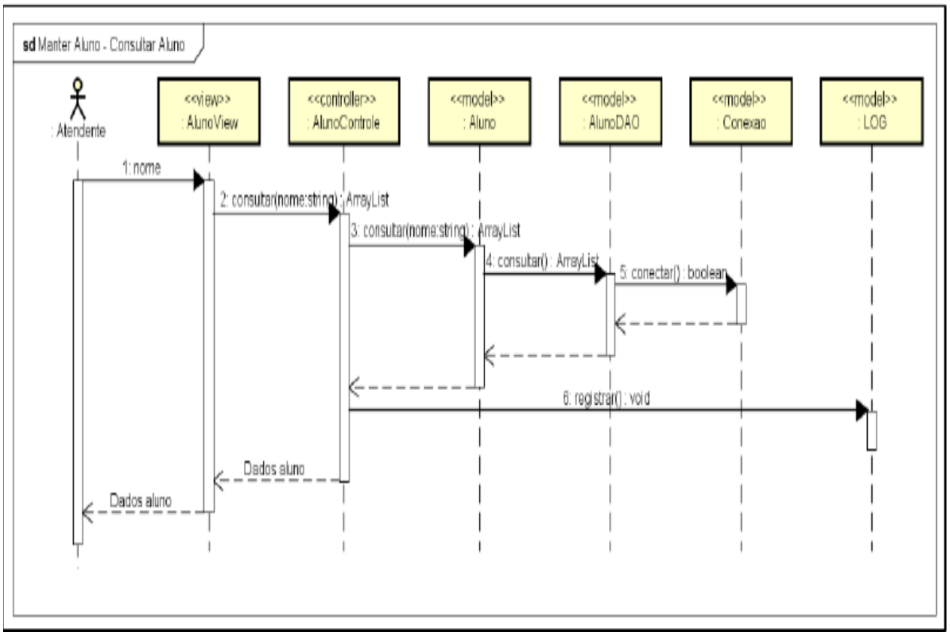
Figura 15 – Diagrama de Sequência de Implementação – Excluir Aluno




Fonte: O próprio Autor

## 5.8 Diagramas de Sequência de Implementação – Manter Aluno (Consultar Aluno)

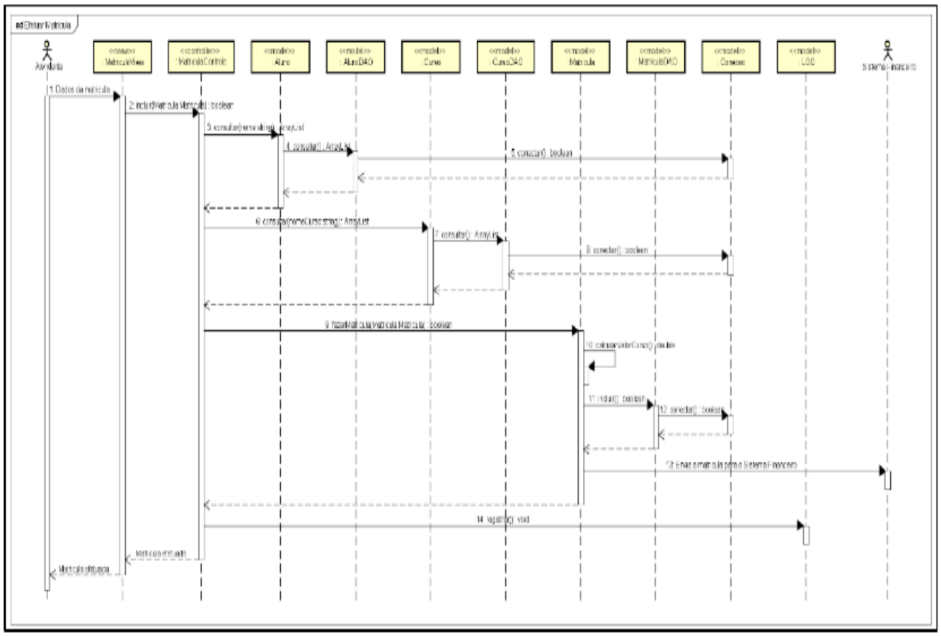
Figura 16 – Diagrama de Sequência de Implementação – Consultar Aluno




Fonte: O próprio Autor

## 5.9 Diagramas de Sequência de Implementação – Efetuar Matrícula

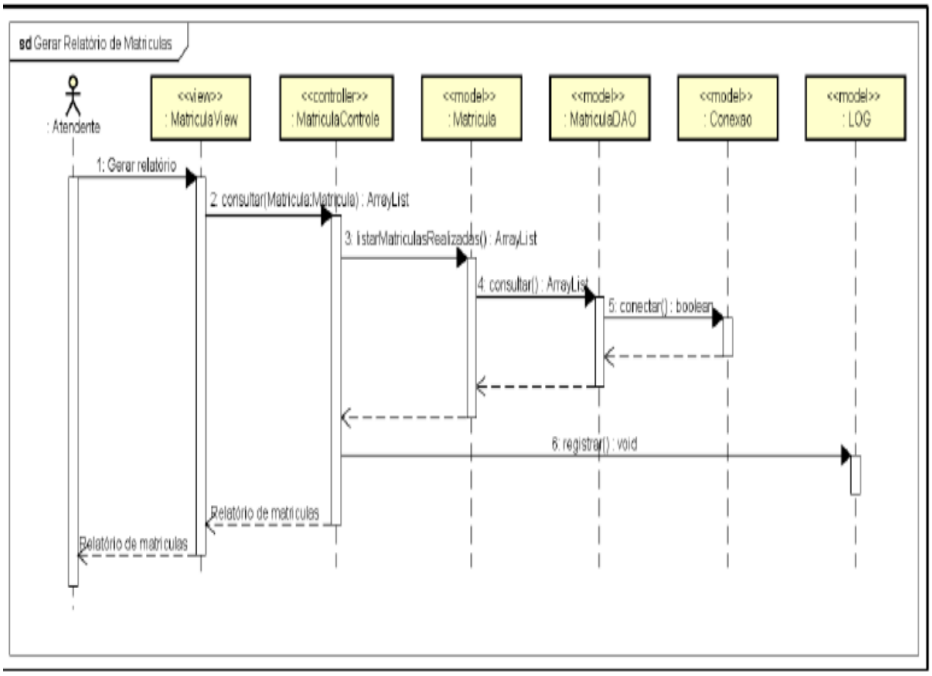
Figura 17 – Diagrama de Sequência de Implementação – Efetuar Matrícula



Fonte: O próprio Autor

## 5.10 Diagramas de Sequência de Implementação – Gerar Relatório de Matrículas

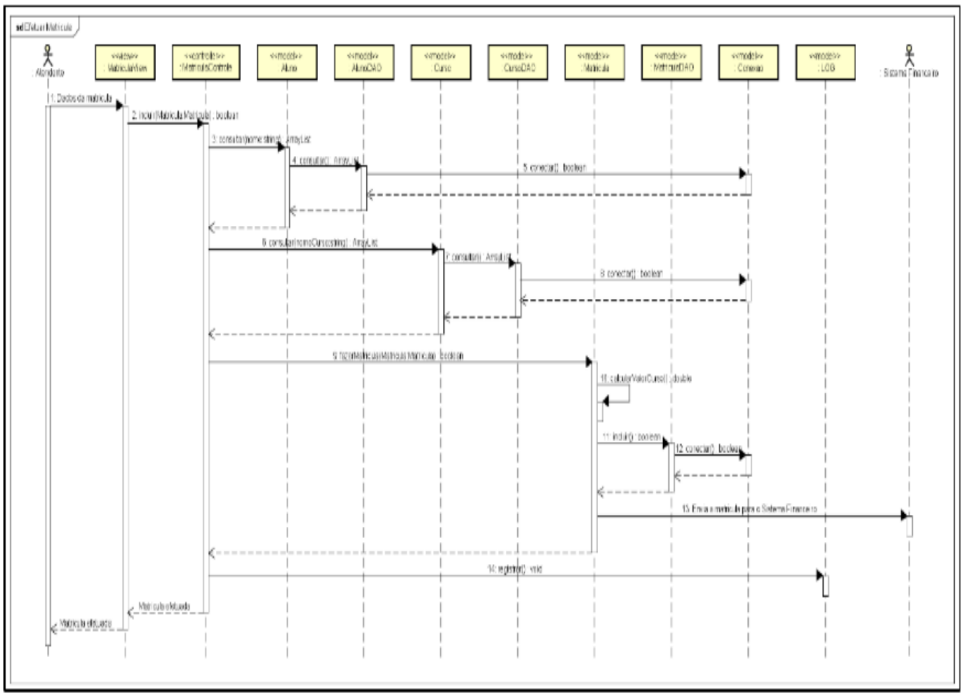
Figura 18 – Diagrama de Sequência de Implementação – Gerar Relatório de Aluno




Fonte: O próprio Autor

## 5.11 Diagramas de Sequência de Implementação – Efetuar Login

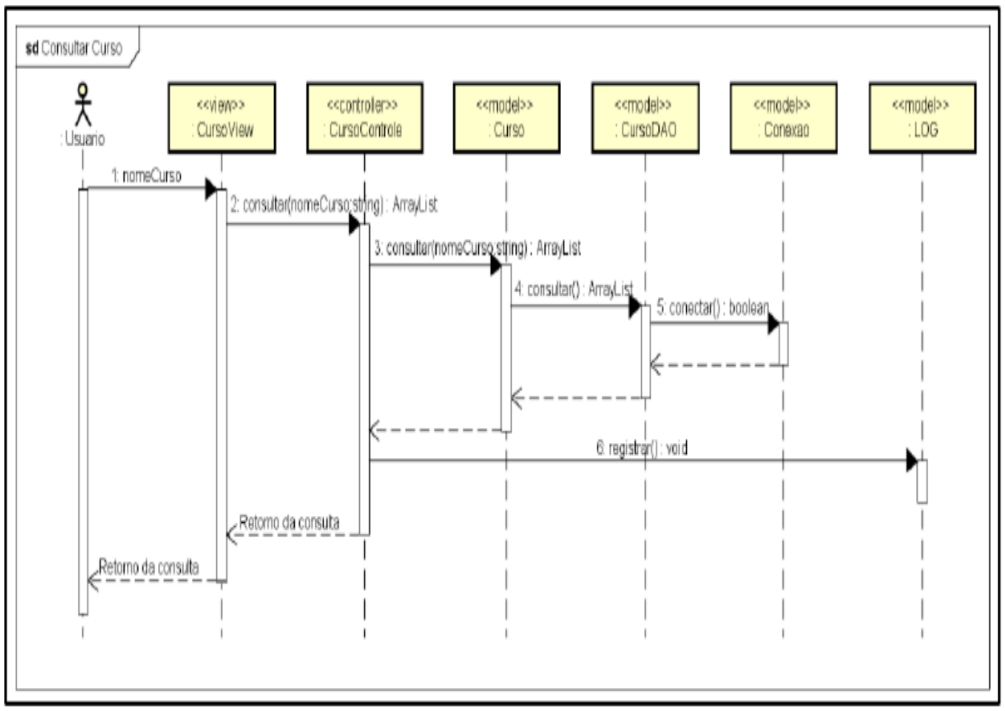
Figura 19– Diagrama de Sequência de Implementação – Efetuar Login



Fonte: O próprio Autor

## 5.12 Diagramas de Sequência de Implementação – Consultar Curso

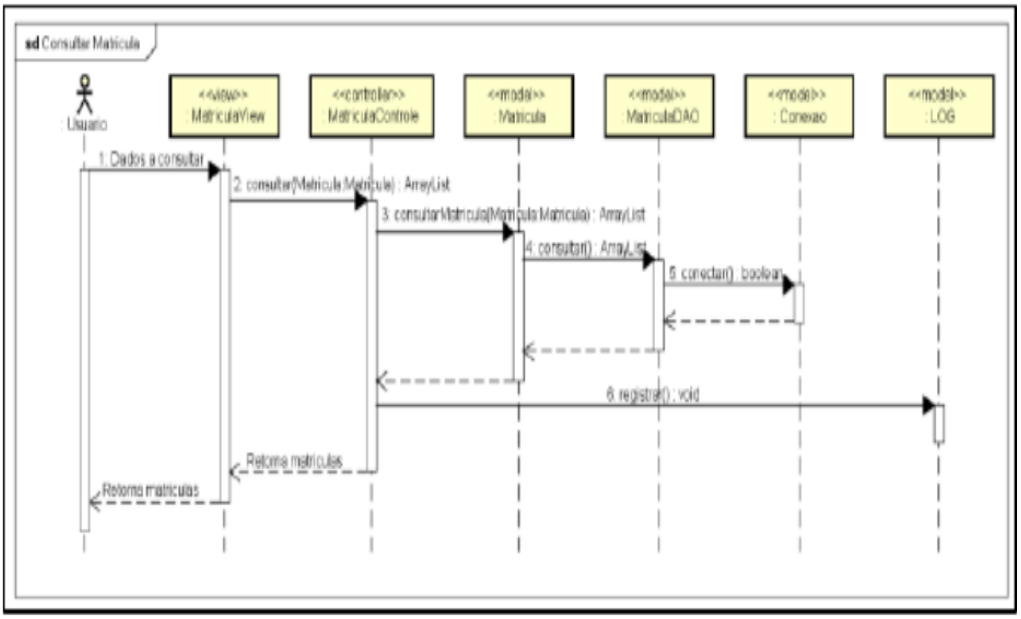
Figura 21 – Diagrama de Sequência de Implementação – Consultar Aluno




Fonte: O próprio Autor

## 5.13 Diagramas de Sequência de Implementação – Consultar Matrícula

Figura 22 – Diagrama de Sequência de Implementação – Consultar Matricula




Fonte: O próprio Autor

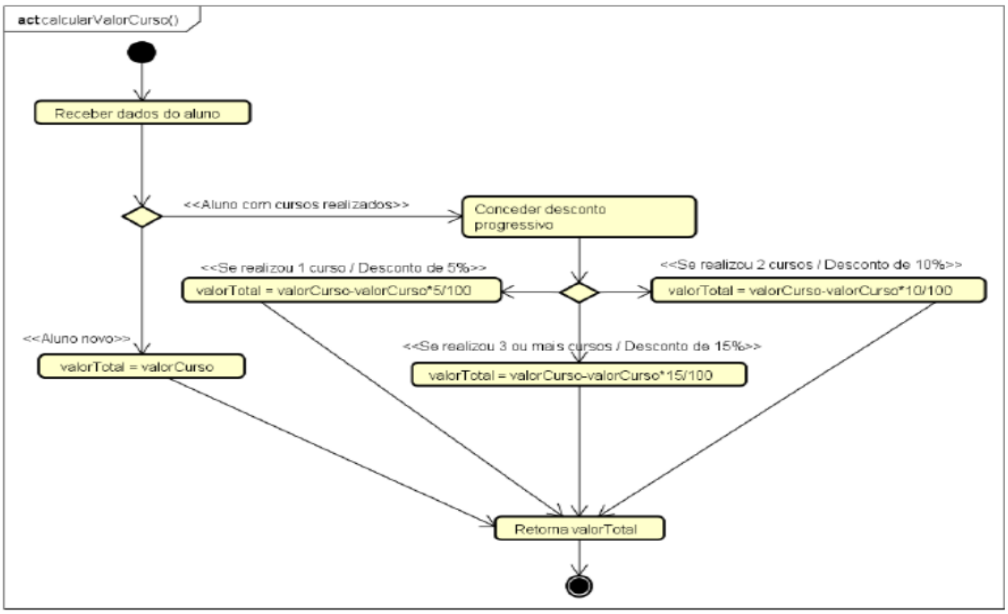
## 6 DIAGRAMA DE ATIVIDADES

O diagrama de atividades fornece uma visualização do comportamento de um sitema descrevendo a sequência de ações em um processo. Os diagramas de atividades são semelhantes a fluxogramas porque mostram o fluxo entre as açôes em uma atividade, porém os diagramas de atividades também podemmostrar fluxos paralelos ou simultâneos e fluxos alternativos.

Utlizaremos o diagrama de atividades para mostrar como as funcionalidades vão realizar requisitos (funções executadas pelas funcionalidades), e a relação de requisitos funcionais com as regras de negócio.

Na figura 23, apresentaremos um diagrama de atividades para o método privado “CalcularValorCurso” da classe matrícula, esse método calcula o valor do curso a ser cobrado do aluno, onde um aluno que já tenha cursado outro curso na instituição têm o direito de 5% de desconto, caso o aluno já tenha efetuado dois cursos ele terá o direito de 10% de desconto e se já tenha cursado três ou mais cursos terá o desconto de 15% de desconto. Através do diagrama de atividades podemos ver o fluxo desse método calcularValorCurso(), onde o sistema recebe os dados do aluno e retornará ao usuário o valor total do curso que o aluno deverá pagar, se o aluno já tiver feito algum curso na instituição terá um desconto de acordo com a quantidade de cursos realizados pelo aluno.

Figura 23 – Diagrama de Atividade do método “calcularValorCurso()”




Fonte: O próprio Autor

## 7 DIAGRAMA DE DISTRIBUIÇÃO

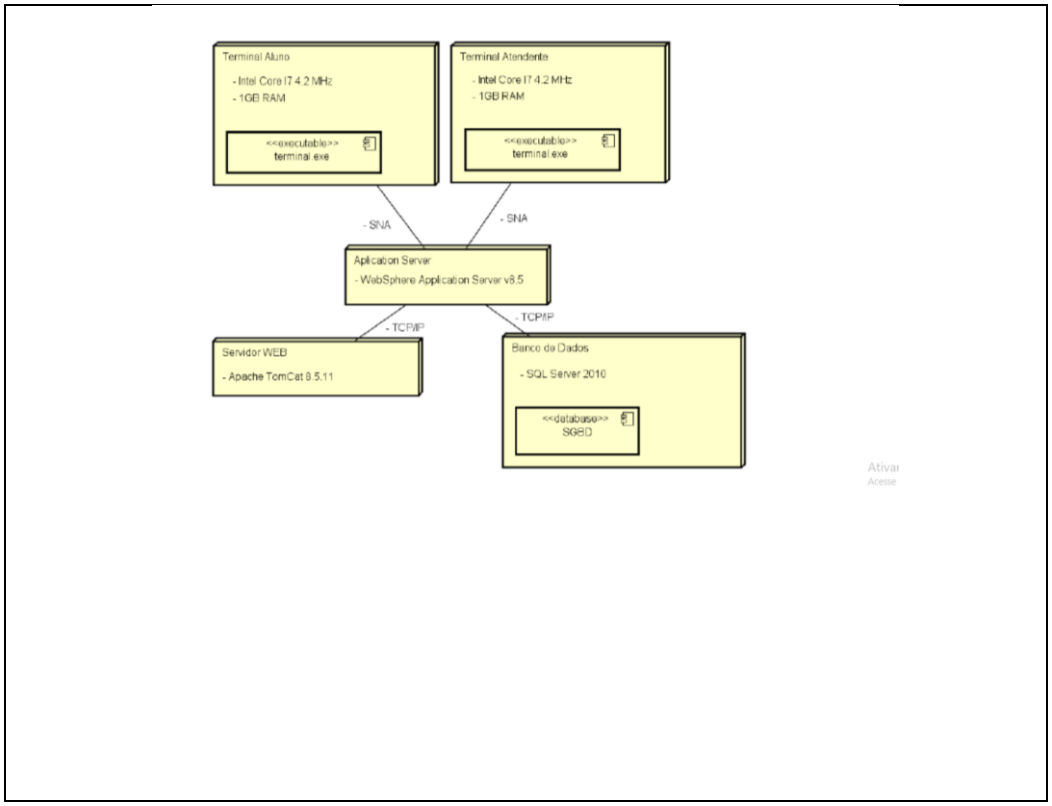
O diagrama de distribuição, ou de implantação, mostra como os componentes são configurados para execução, em “nós” de processamento (LARMAN, 2007).

O diagrama de distribuição tem como elemento básico os nós, que representem os computadores configurados para execução, as associações entre os nós, que são suas ligações e os artefatos, que representam as entidades físicas do mundo real.

Segundo (VERSOLATO,2015) Um “nó” de processamento é um recurso computacional que permite a execução de um sistema de software ou de parte dele, como um componente. Esse “nó” poderá ser um computador, ou outro dispositivo.

Na figura 24, foi elaborado um diagrama de distribuição elaborado para a execução do sistema proposto neste projeto onde podemos ver o relacionamento entre os componentes de software e hardware no sistema.

Figura 24 – Diagrama de Distribuição




Fonte: O próprio Autor

## 8 CONCLUSÃO

Neste PIM-VII abordamos a fase de projetos de software onde através de um diagrama de casos de uso e um diagrama de classe que demonstrava os requisitos e do sistema de controle de matrículas foi desenvolvida uma fase de projeto para a construção de um sistema de matrículas de cursos livres, sendo os cursos de informática e artes.

Foi realizada a fase de projetos deste sistema utilizando o desenho de arquitetura MVC, e o diagrama de classes para representar os principais elementos do projeto. Utilizamos os diagramas da UML para apresentar e documentar o funcionamento deste sistema. Demos ênfase em como desenvolver a fase de projetos de software com qualidade, respeitando os requisitos levantados junto ao usuário que solicitou o sistema na fase de levantamento de requisitos, respeitando as regras de negócio e as considerações técnicas, para a construção da modelagem e arquitetura de um software de qualidade, funcional e que atendam aos requisitos do cliente.

Através do PIM-VII, foi possível colocar em prática todo o conhecimento adquirido neste bimestre, principalmente nas habilidades estudadas na disciplina de “Projeto de Sistema Orientado a Objetos” com ênfase no projeto(designer)

**REFERÊNCIAS BIBLIOGRAFICAS**

VERSOLATTO, Fábio Rossi. **Projeto de Sistemas Orientado a Objetos.** São Paulo: Editora Sol, 2015.

PINTO, Gisele Lopes Batista; DOS SANTOS, Luiz Fernando. **Administração de Banco de Dados.** Editora Sol, 2012.

Costa, Ivanir. **Engenharia de Software-I.** Editora Sol, 2014.

**Manual Unip Interativa PIM VII**. Disponível em: https://ava.ead.unip.br/bbcswebdav/pid-766139-dt-content-rid-2920160_1/institution/2019/GRADE%20EAD/PROJETO%20INTEGRADO%20MULTIDISCIPLINAR%20%20DP/SUP%20TEC%20EM%20AN%C3%81LISE%20E%20DESENVOLVIMENTO%20DE%20SISTEMAS/3019-50%20-%20Projeto%20Integrado%20Multidisciplinar%20VII/Manual.pdf
Acesso em 01 de outubro de 2019.

**Padrão MVC.** Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308#MVC
Acesso em: 04 de outubro de 2019.

**Diagrama de sequência.** Disponível em: https://www.ibm.com/support/knowledgecenter/ptbr/SSCLKU_7.5.5/com.ibm.xtools.sequence.doc/topics/cseqd_v.html
Acesso em: 05 de outubro de 2019.