

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE**

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

FELIPE EDUARDO CABRAL VIANNA DE SOUZA

MODELAGEM DE UM SISTEMA PARA GERAR HORÁRIOS DE AULAS

**NATAL
2014**

FELIPE EDUARDO CABRAL VIANNA DE SOUZA

MODELAGEM DE UM SISTEMA PARA GERAR HORÁRIOS DE AULAS

Trabalho de Curso submetido ao Centro
Universitário do Rio Grande do Norte
como requisito necessário para a
obtenção do Grau de Bacharel em
Sistemas de Informação. Sob a
orientação do Professor Ytalo Rosendo
do Amaral.

**Natal
2014**

Catálogo na Publicação – Biblioteca do UNI-RN
Setor de Processos Técnicos

Souza, Felipe Eduardo Cabral Vianna de.

Modelagem de um sistema para gerar horários de aulas. / Felipe Eduardo Cabral Vianna de Souza. – Natal, 2014.

41 f.

Orientador: Profº. Ytalo Rosendo do Amaral.

Monografia (Graduação em Sistemas de Informação) – Centro Universitário do Rio Grande do Norte.

1. Sistema – Monografia. 2. Modelagem – Monografia. 3. Horários de Aulas – Monografia. I. Amaral, Ytalo Rosendo do. II. Título.

RN/UNI-RN/BC

CDU 004

horário, sistema, ensino, automaticamente, modelagem.

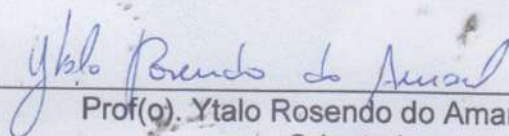
FELIPE EDUARDO CABRAL VIANNA DE SOUZA

MODELAGEM DE UM SISTEMA PARA GERAR HORÁRIOS DE AULAS

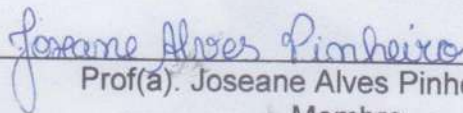
Trabalho de Conclusão de Curso
apresentado ao Centro Universitário do
Rio Grande do Norte, em cumprimento
às exigências legais como requisito
final para obtenção do título de
Bacharel em Sistemas de Informação.

Aprovado (a) em 11/12/14.

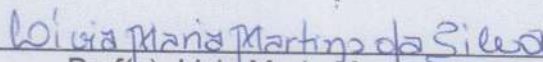
BANCA EXAMINADORA



Prof(o). Ytalo Rosendo do Amaral (MSC)
Orientador



Prof(a). Joseane Alves Pinheiro (MSC)
Membro



Prof(a). Livia Maria Martins da Silva (MSC)
Membro

*Dedico este trabalho a todos
que me ajudaram ao longo
desses 4 anos.*

AGRADECIMENTOS

Agradeço a todos que contribuíram no decorrer desta graduação, em especial:

A Deus, que me deu o dom da vida;

A minha família, especificamente meus pais que me deram todo o suporte ao longo desses anos;

A Anne Caroline que me apareceu no momento certo e me ajudou nas decisões, compreendendo e me incentivando em todas as horas;

Ao orientador Prof. Ytalo Rosendo do Amaral que me ajudou nos últimos instantes e me ajudou a concluir a monografia;

Aos meus colegas que foram companheiros e me ajudaram em vários momentos.

RESUMO

O intuito deste trabalho é fazer a modelagem de um sistema que faça a geração de horários de aulas, onde a finalidade do mesmo é suprir a dificuldade que uma instituição de ensino encontra ao fazer esse tipo de tarefa, fazendo com que este horário seja gerado automaticamente. Esta monografia contém o método de como foi feito este estudo, para se chegar à devida solução para este problema, visando os diversos fatores que precisam ser considerados para que cada horário seja gerado.

Palavras chave: horário, sistema, ensino, automaticamente, modelagem.

ABSTRACT

The purpose of this work is to model a system that makes the generation of class schedules, where the purpose of it is to meet the difficulty that an educational institution is to make this kind of task, making this time is automatically generated. This monograph contains the method as was done this study, to arrive at the proper solution to this problem by targeting the different factors that need to be considered so that each time it is generated.

Keywords: time, system, educational, automatically, modeling.

LISTA DE SIGLAS

BD	Banco de Datos
OMG	Object Management Group
UML	Unified Modeling Language

LISTA DE FIGURAS

Figura 1 - Exemplo de relação entre ator e caso de uso.....	20
Figura 2 - Exemplo de diagrama de casos de uso.....	22
Figura 3 - Exemplo de diagrama de casos de uso.....	23
Figura 4 - Exemplo de classe contendo atributos e métodos de um cliente.....	26

LISTA DE TABELAS

Tabela 01 - Tabela de requisitos funcionais.....	29
Tabela 02 - Tabela com descrição dos casos de uso.....	30

SUMÁRIO

1.	INTRODUÇÃO.....	13
1.1	OBJETIVO GERAL.....	14
1.2	OBJETIVOS ESPECÍFICOS.....	14
2.	REFERENCIAL TEÓRICO.....	15
2.1	ANÁLISE DE SISTEMAS.....	15
2.2	UNIFIED MODELING LANGUAGE (UML).....	16
2.3	DIAGRAMA DE CASOS DE USO.....	18
2.3.1	OBJETIVO.....	18
2.3.2	NOTAÇÃO.....	18
2.4	DIAGRAMA DE ENTIDADE E RELACIONAMENTO.....	24
2.5	DIAGRAMA DE CLASSES.....	24
2.6	ASTAH.....	27
3.	METODOLOGIA.....	28
4.	RESULTADO.....	29
4.1	ANÁLISE DE REQUISITOS.....	29
4.2	CASOS DE USO.....	30
4.2.1	ESPECIFICAÇÕES DOS CASOS DE USO.....	30
5.	CONCLUSÃO.....	38
	REFERÊNCIAS.....	39
	ANEXO.....	41
	ANEXO 1.....	41
	ANEXO 2.....	42

1.1 OBJETIVO GERAL

Desenvolver a modelagem de um software para trabalhar com geração de horários de aula.

1.2 OBJETIVOS ESPECÍFICOS

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- ✓ Estudo e análise de trabalhos sobre geração de horários de aula
- ✓ Conversa com coordenadores de curso sobre quais requisitos devem ter um sistema de horários de aula
- ✓ Criar o diagrama de classe, no qual irá representar o modelo de domínio desse sistema.
- ✓ Criar os casos de usos do sistema, onde o mesmo irá identificar as ações que o ator poderá fazer dentro do sistema.

1 INTRODUÇÃO

As instituições educacionais estão em crescimento contínuo, tendo que aumentar suas estruturas, aumentar a quantidade de professores e funcionários, devido à quantidade de pessoas que estão tendo acesso a essas instituições.

Aumentando o número de pessoas, aumentando a procura, isso faz com que, consequentemente, as instituições de ensino abram mais turmas e com isso irão surgindo novas vagas todos os anos, o que obviamente fará com que aumente o número de alunos nas instituições.

Então, sabe-se que gerar horário de uma turma não é algo simples a se fazer, devido à complexidade da situação, pela dificuldade de organizar os horários justamente por que existem vários fatores que devem ser levados em consideração, como: restrições de horário de professores e do curso. E gerar um horário de várias turmas, de vários cursos fica ainda mais complicado. Qual seria a melhor forma de automatizar essa geração de horários em uma instituição de ensino?

2 REFERENCIAL TEÓRICO

2.1 ANÁLISE DE SISTEMAS

Por Análise de Sistemas entende-se a atividade inicial do processo de desenvolvimento de sistemas em que se determina e especifica o que um sistema deve fazer, bem como as circunstâncias sob as quais deve operar, envolvendo geralmente um esforço colaborativo entre analistas de sistemas e utilizadores, no qual os analistas procuram obter a partir dos utilizadores, num processo gradual e cumulativo, o maior conhecimento possível acerca do domínio do discurso do sistema e respectivo ambiente (ROCHA, 2012).

Análise de sistemas é a atividade que tem como finalidade a realização de estudos de processos a fim de encontrar o melhor caminho racional para que a informação possa ser processada. Os analistas de sistemas estudam os diversos sistemas existentes entre hardwares (equipamentos), softwares (programas) e o usuário final (GUILHERMINO).

Os seus comportamentos e aplicações são desenvolvidos a partir de soluções que serão padronizadas e transcritas da forma que o computador possa executar.

Os profissionais da área geram softwares (programas), que são executados em hardwares (equipamentos) operados por usuários (indivíduos), preparados e treinados em procedimentos operacionais padronizados, dotados de conhecimentos do software e hardware para seu trabalho. A partir de então a análise de sistemas é uma profissão cujas responsabilidades concentram-se na análise do sistema e na administração de sistemas computacionais. Cabe a este profissional parte da organização, implantação e manutenção de aplicativos e redes de computadores, ou seja, o analista de sistemas é o responsável pelo levantamento de informações sobre uma empresa a fim de utilizá-las no desenvolvimento de um sistema para a mesma ou para o levantamento de uma necessidade específica do cliente para desenvolver este programa específico com base nas informações colhidas.

A Análise de Sistemas consiste nos métodos e técnicas de investigação e especificação da solução de problemas, a partir dos requisitos levantados, para criação e implementação de software em algum meio que o suporte (ESCOLA ALCIDES MAYA).

O principal objetivo da análise de sistemas é realizar um mapeamento prévio do comportamento requerido para os elementos de modelagem no sistema a serem implementados posteriormente nas fases de construção. Durante as etapas iniciais de um projeto, é comum realizarmos um refinamento nos detalhes e na precisão do “desenho do sistema” a fim de conseguir classes de análise que possam evoluir antes de serem detalhadas durante as atividades de especificação e implementação.

2.2 UNIFIED MODELING LANGUAGE (UML)

A Unified Modelling Language (UML) é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de 'software' orientados por objetos. Não é um método de desenvolvimento, o que significa que não lhe diz o que fazer primeiro ou o que fazer depois ou como desenhar o seu sistema, mas ajuda-o a visualizar o seu desenho e a comunicar com os outros. A UML é controlada pelo Object Management Group (OMG) e é a norma da indústria para descrever graficamente o 'software'.

Mariana Martinez (2014) fala que a UML fornece a tecnologia necessária para apoiar a prática de engenharia de software orientada a objetos, que poderá ser a linguagem de modelagem padrão para modelar sistemas concorrentes e distribuídos. Utiliza-se de um conjunto de técnicas de notação gráfica para criar modelos visuais de software de sistemas intensivos, combinando as melhores técnicas de modelagem de dados, negócios, objetos e componentes. É uma linguagem de modelagem única, comum e amplamente utilizável.

A UML é feita para o desenho de software orientado por objetos e tem uma utilização limitada para outros paradigmas de programação.

É composta por muitos elementos de modelo que representam as diferentes partes de um sistema de software. Os elementos UML são usados para criar diagramas, que representam uma determinada parte, ou um ponto de vista do sistema.

Dentre os diagramas existentes, abaixo temos alguns exemplos (KDE DOCUMENTATION, 2014):

- **Diagrama de Caso de Uso:** mostra atores (pessoas ou outros usuários do sistema), casos de uso (os cenários onde eles usam o sistema), e seus relacionamentos.
- **Diagrama de Classe:** mostra classes e os relacionamentos entre elas.
- **Diagrama de Sequência:** mostra objetos e uma sequência das chamadas do método feitas para outros objetos.
- **Diagrama de Colaboração:** mostra objetos e seus relacionamentos, colocando ênfase nos objetos que participam na troca de mensagens.
- **Diagrama de Estado:** mostra estados, mudanças de estado e eventos num objeto ou uma parte do sistema.
- **Diagrama de Atividade:** mostra atividades e as mudanças de uma atividade para outra com os eventos ocorridos em alguma parte do sistema.
- **Diagrama de Componente:** mostra os componentes de programação de alto nível.
- **Diagrama de Distribuição:** mostra as instâncias dos componentes e seus relacionamentos.
- **Diagramas de Entidade-Associação:** mostram os dados e as relações e as restrições entre os dados.

2.3 DIAGRAMA DE CASOS DE USO

2.3.1 Objetivo

O Diagrama de *Casos de Uso* tem o objetivo de auxiliar a comunicação entre os analistas e o cliente.

Um diagrama de Caso de Uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

Segundo SAMPAIO (2007) "O cliente deve ver no diagrama de Casos de Uso as principais funcionalidades de seu sistema".

2.3.2 Notação

O diagrama de Caso de Uso é representado por:

- Atores;
- Casos de uso;
- Relacionamentos entre estes elementos.

Estes relacionamentos podem ser:

- Associações entre atores e casos de uso;
- Generalizações entre os atores;
- Generalizações, *extends* e *includes* entre os casos de uso.

Em maiores detalhes:

- **Atores:** Um ator é representado por um boneco e um rótulo com o nome do ator. Um ator é um usuário do sistema, que pode ser um usuário humano ou um outro sistema computacional.

- **Caso de uso:** Um *caso de uso* é representado por uma elipse e um rótulo com o nome do *caso de uso*. Um *caso de uso* define uma grande função do sistema. A implicação é que uma função pode ser estruturada em outras funções e, portanto, um *caso de uso* pode ser estruturado.
- **Relacionamentos:**

1- Relacionamento entre atores:

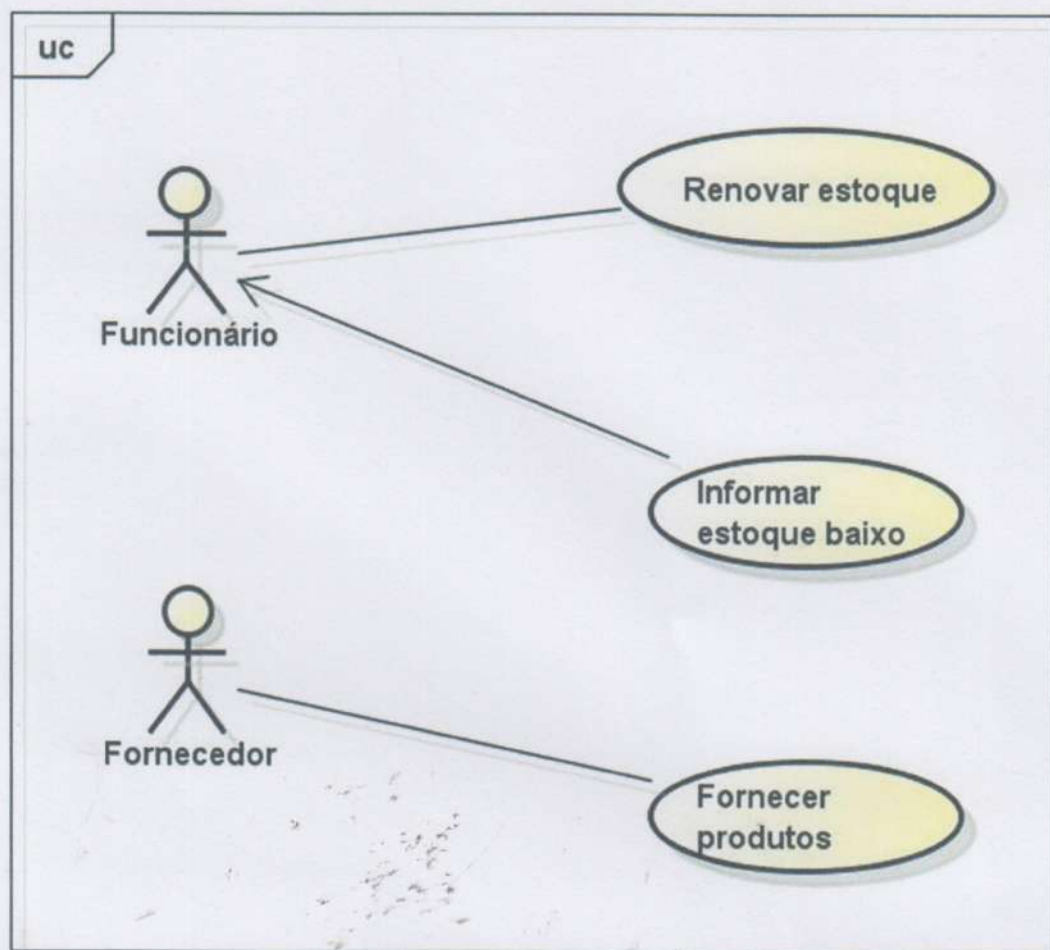
Relacionamento de Associação - é uma ligação entre classes, portanto uma ligação entre objetos e suas respectivas classes.

Relacionamento de Generalização - é o relacionamento entre um elemento mais geral e um mais específico. O mais específico pode conter apenas informações adicionais.

2- Relacionamento entre atores e casos de uso:

Dependendo da situação, pode indicar duas alternativas a escolher: ator que inicia o caso de uso ou simplesmente a direção dos dados.

A Figura 1 mostra tipos de relacionamentos entre atores e casos de uso.



powered by astah

FIGURA 1 – exemplo de relação entre ator e caso de uso.

Fonte: <http://hslife.com.br/news/7>

3 RELACIONAMENTO ENTRE CASOS DE USO

Include - Uma das formas de interação, um dado caso de uso pode *incluir* outro. Incluir é uma relação direta entre dois casos de usos, implicando que o comportamento do caso de uso incluído é inserido no comportamento do caso de uso inclusor. Esta relação indica uma obrigatoriedade do caso de uso incluir a funcionalidade do caso de uso incluído. Assim, sempre que o primeiro ocorrer, obrigatoriamente o incluído ocorrerá.

A notação é uma seta pontilhada para o caso de uso incluído com o estereótipo <<include>>.

Extends - Outra forma de interação, um caso de uso pode estender outro. Esta relação indica que o comportamento do caso de uso estendido pode ser ou não inserida no caso de uso extensor. Notas ou restrições podem ser associadas ao relacionamento para ilustrar as condições em que este comportamento será executado.

A notação é uma seta pontilhada da extensão para o caso de uso estendido com a etiqueta <<extends>>.

Exemplos de um diagrama de casos de uso:

Exemplo 1- A Figura 2 mostra um exemplo de diagrama de casos de uso.

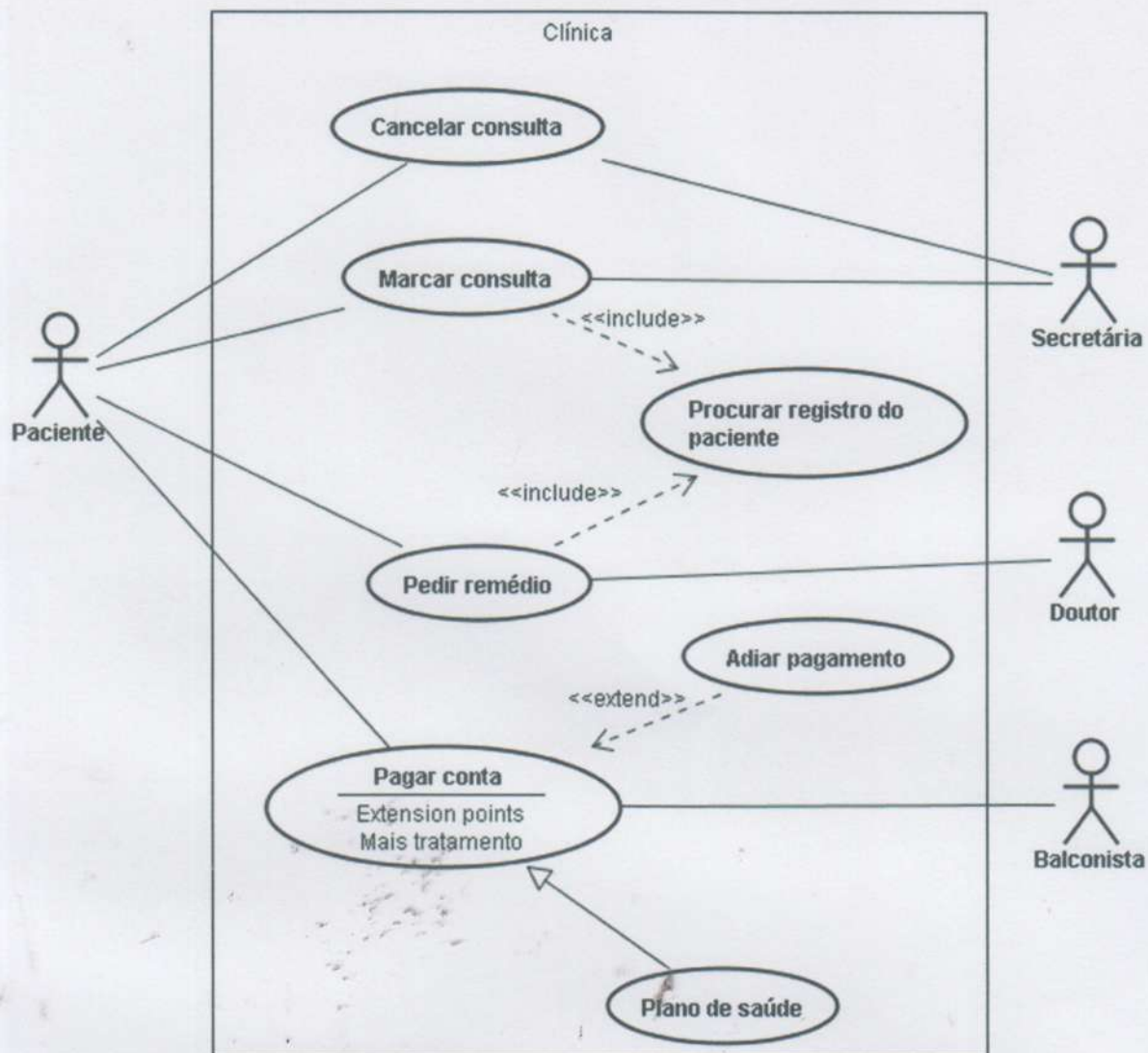


FIGURA 2 – Exemplo de diagrama de casos de uso (Fonte: UFCG)

(Fonte: <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>)

Exemplo 2 - A Figura 3 mostra um exemplo de diagrama de casos de uso.

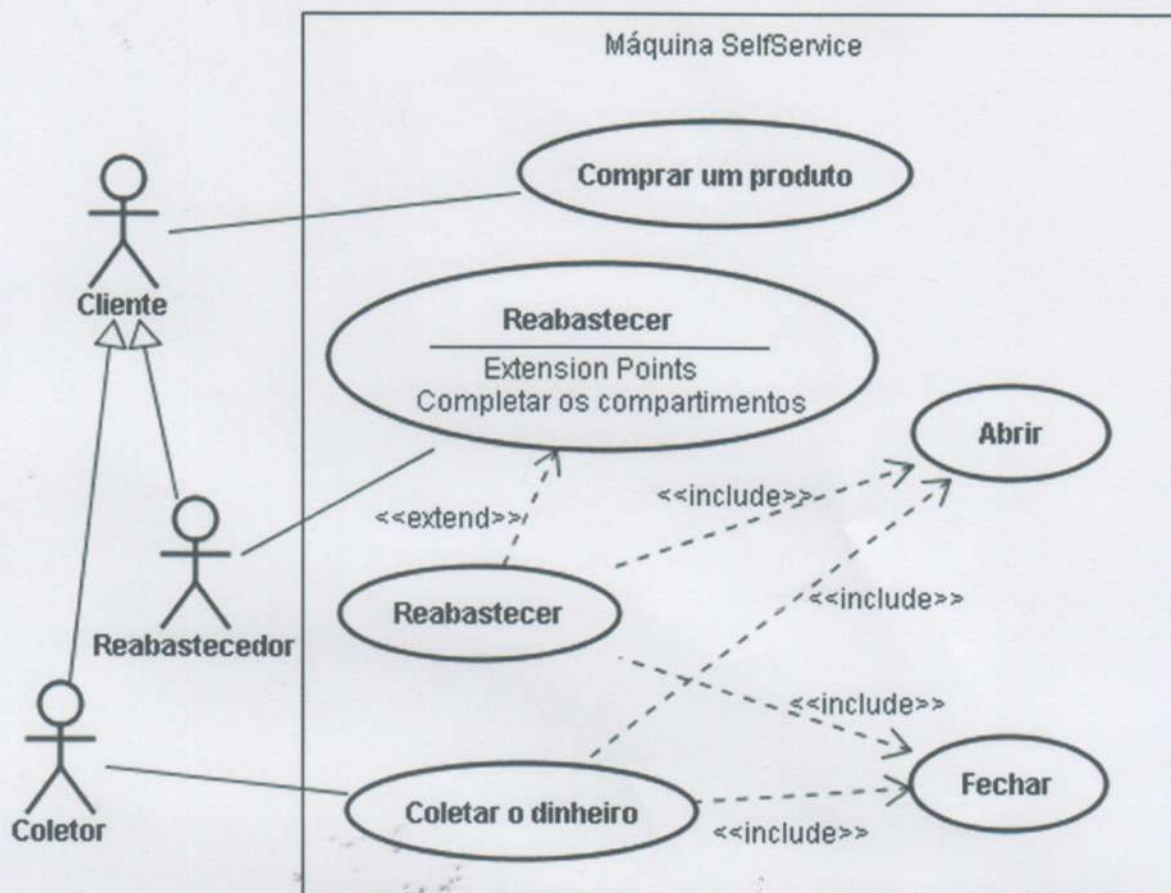


FIGURA 3 – Exemplo de diagrama de casos de uso. (Fonte: UFCG)

(Fonte: <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>)

2.4 DIAGRAMA DE ENTIDADE E RELACIONAMENTO

A ideia fundamental deste modelo é a de conservar como conceitos de base os conceitos genéricos (objetos, associação, propriedade) usados no processo de abstração que vai da observação de uma realidade à sua descrição (UFPR).

Maurício fala que Diagrama de Entidade e Relacionamento é um modelo baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados entidades e nos relacionamentos entre esses objetos (MELO).

- Entidade - Uma entidade é o objeto do mundo real (concreto ou abstrato) que nós queremos representar no computador e que tem uma existência própria. Uma entidade existe independentemente do fato que ela está ligada a outros objetos do BD.
- Associação - Uma associação liga várias entidades onde cada uma delas ocupa um "papel". Se a associação liga dois (ou mais) entidades de um mesmo tipo ela é dita cíclica e neste caso a especificação do papel de cada entidade torna-se indispensável (UFMA).
- Atributo - Um atributo é a propriedade associada à um tipo de entidade, ou à um tipo de associação.

2.5 DIAGRAMA DE CLASSES

Um diagrama de classe UML descreve o objeto e as estruturas usadas pelo seu aplicativo internamente e comunicação com seus usuários de informações. Ele descreve as informações sem referência a qualquer implementação específica. Suas

classes e relacionamentos podem ser implementados de diversas maneiras, como tabelas de banco de dados, etc. (MICROSOFT DEVELOPER NETWORK).

O diagrama de classes é considerado por muitos autores como o mais importante e o mais utilizado diagrama da UML. Seu principal enfoque está em permitir a visualização das classes que irão compor o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do sistema se relacionam, se complementam e transmitem informações entre si. Este diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em definir a estrutura lógica das mesmas. O diagrama de classes serve como base para a construção da maior parte dos demais diagramas da UML (SILVA, 2014).

Basicamente, o diagrama de classes é composto por suas classes e pelas associações existentes entre elas, ou seja, os relacionamentos entre as classes.

Segundo GUEDES (2009), o objetivo do diagrama de classes é mostrar os relacionamentos existentes entre as classes que são abstraídas no projeto, e como esses relacionamentos colaboram para a execução de um processo específico.

Existem três perspectivas que você pode usar quando projetar diagramas de classes (SILVA, 2014):

- **Conceitual:** Se tomarmos a perspectiva conceitual, você projeta um diagrama que representa os conceitos do domínio que está sendo estudado. Estes conceitos serão naturalmente relacionados às classes que irão executá-los. Na verdade, um modelo conceitual deve ser projetado com pouca ou nenhuma preocupação com o software que poderá implementá-lo. Portanto, deve ser considerado independente da linguagem utilizada. Esta perspectiva recebe o nome de perspectiva essencial.

- **Especificação:** Agora estamos examinando o software, mas estamos analisando as suas interfaces, não a sua implementação. O desenvolvimento orientado a objetos dispõe muita ênfase na diferença entre interface e implementação, mas isso é

frequentemente negligenciado na prática porque a noção que temos de classe em uma linguagem orientada a objetos combina interface com implementação.

Implementação: Nesta visão, realmente temos classes e estamos pondo a implementação às claras. Esta é, provavelmente, a perspectiva usada com mais frequência.

A compreensão das diversas perspectivas é crucial tanto para desenhar como para ler diagramas de classes. Infelizmente, as linhas entre as perspectivas não são rígidas, e a maioria dos analistas de sistemas não se preocupa em ter suas perspectivas classificadas quando eles estão desenvolvendo a modelagem de um sistema.

A Figura 4 apresenta um exemplo de diagrama de classe de um modelo de domínio, onde a mesma contém atributos e um método.

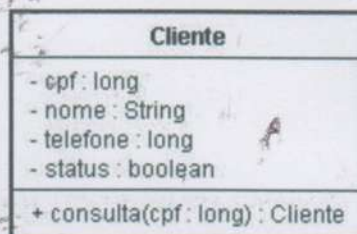


Figura 4 - Exemplo de classe contendo atributos e métodos de um cliente.

2.6 ASTAH

Astah é um software para modelagem UML. É desenvolvido na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui uma máquina virtual Java (ASTAH).

É uma ferramenta gratuita voltada para a modelagem de diagramas UML (Unified Modeling Language). A ferramenta é conhecida por sua praticidade e simplicidade em elaborar diagramas, como por exemplo: diagramas de classe, caso de uso, sequência, atividade, comunicação, máquina de estado, componentes, implantação, estrutura de composição, objetos e pacotes (BRONDANI; AREND; SOUZA).

"O Astah é uma ferramenta para modelagem UML muito fácil de usar e muito útil. "(Elton Minetto, 2012)

3 METODOLOGIA

O tipo de pesquisa utilizada para esse projeto quanto a sua natureza, foi a pesquisa aplicada, pois foi feita uma pesquisa por três sistemas que tivessem as mesmas funcionalidades, no sentido de gerar horários automaticamente. Logo após, houve o entendimento sobre quais requisitos eram usados nesses sistemas, para obter com sucesso o objetivo principal, que é modelar um sistema que faça geração de horários de aulas.

Diante da análise, constatei que seriam necessários alguns dados como: carga horária, disponibilidade e nome dos professores, nome dos cursos, nome das turmas, nome dos turnos, nome das disciplinas, quais disciplinas serão lecionadas por cada professor, quais turma tem em cada curso e em qual turno. Justamente para poder fazer a relação necessária entre esses dados e, a partir daí, conseguir fazer o horário.

4 RESULTADO

4.1 ANÁLISE DE REQUISITOS

MAKESYS (2013) “Saber o que é fundamental, baseado nos problemas que o sistema precisa resolver, pode fazer a diferença entre uma ferramenta de sucesso e outra defasada”.

Após estudos para saber a melhor forma de fazer uma modelagem para esse tipo de sistema, foi feito uma análise de requisitos, os quais são propostos para este sistema.

Requisitos Funcionais descrevem explicitamente as funcionalidades e serviços do sistema (Figueiredo, 2014).

Segue abaixo a Tabela 01 com os requisitos funcionais:

ID	Descrição
RF01	O usuário pode criar, excluir ou modificar Professores.
RF02	O usuário pode criar, excluir ou modificar Cursos.
RF03	O usuário pode criar, excluir ou modificar Turmas.
RF04	O usuário pode criar, excluir ou modificar Turnos.
RF05	O usuário pode criar, excluir ou modificar Disciplinas.
RF06	O usuário pode criar, excluir ou modificar Horários.

TABELA 01- Tabela de requisitos funcionais

Fonte: resultado da pesquisa do próprio autor.