

**FACULDADE DE TECNOLOGIA CARLOS DRUMMOND DE ANDRADE**  
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**CAROLINA TRINDADE MARQUES**

**FABIO SANTOS GOMES**

**LARISSA AMANCIO**

**PATRÍCIA DE FÁTIMA OLIVEIRA**

**WILLIAM BARBOSA DOS SANTOS**

**DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DA GRADE**  
**DE AULA PARA O CORPO DOCENTE**

**SÃO PAULO / SP**

**2013**

**CAROLINA TRINDADE MARQUES**

**FABIO SANTOS GOMES**

**LARISSA AMANCIO**

**PATRÍCIA DE FÁTIMA OLIVEIRA**

**WILLIAM BARBOSA DOS SANTOS**

**DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DA GRADE  
DE AULA PARA O CORPO DOCENTE**

Trabalho apresentado como exigência parcial  
para a obtenção da formação ao Curso de  
Ciência da Computação da Faculdade de  
Tecnologia Carlos Drummond de Andrade.

Professora Orientadora: Lúcia Contente Mos

**SÃO PAULO – SP**

**2013**

**CAROLINA TRINDADE MARQUES**

**FABIO SANTOS GOMES**

**LARISSA AMANCIO**

**PATRÍCIA DE FÁTIMA OLIVEIRA**

**WILLIAM BARBOSA DOS SANTOS**

**DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DA GRADE  
DE AULA PARA O CORPO DOCENTE**

Trabalho apresentado em 12 de Dezembro de 2013 como exigência parcial para a obtenção da formação ao Curso de Ciência da Computação da Faculdade de Tecnologia Carlos Drummond de Andrade.

Aprovado em 12 de Dezembro de 2013

---

Professora Lucia Contente Mos  
Faculdade de Tecnologia Carlos Drummond de Andrade  
Orientadora

---

Professor convidado  
Faculdade de Tecnologia Carlos Drummond de Andrade

---

Professor convidado  
Faculdade de Tecnologia Carlos Drummond de Andrade

---

Nota

**SÃO PAULO – SP**

**2013**

## **DEDICATÓRIA**

Carolina:

Dedico este trabalho aos meus pais, Ernando e Selma por terem sempre feito o melhor para mim, às minhas irmãs, Camila e Beatriz, e a toda minha família, em especial à minha tia Alzira, por durante grande parte da minha infância ter ajudado meus pais na minha criação. Dedico também a Deus, pois Ele é a razão de minha vida e sem Ele, nada sou.

Fabio:

Dedico este trabalho aos meus familiares Maria Stela, Antônio Carlos e Fabiane Gomes e a Daniele Cavalcante que nos ajudou indiretamente e diretamente na elaboração e conclusão deste trabalho.

Larissa:

Dedico a todas as pessoas que me ajudaram e continuam ajudando para a construção do meu futuro. André Kato, Maria Inês, a todos os meus amigos da faculdade Carlos Drummond de Andrade, ETEC Aprígio Gonzaga e a Empresa Umanni.

Patrícia:

Dedico esse trabalho aos meus filhos Izabela, Vinicius, Victor e Yasmin que sempre foram minha motivação para que eu nunca desistisse e ao meu marido que foi a primeira pessoa a me dizer: "Você só não consegue se não quiser".

William:

Dedico este trabalho aos grandes mestres que passaram pela minha vida. Nilce Elena Gonzalez, que despertou minha paixão pelo conhecimento. E Vilma Cardoso dos Santos que me apresentou ao mundo da programação pela primeira vez.

## **AGRADECIMENTOS**

Carolina:

Agradeço aos meus pais, Ernando e Selma, pelo apoio de sempre, por terem sempre feito o melhor, por terem me dado excelente educação e me ensinarem que na vida, não se consegue nada sem trabalhar e lutar por isso. Agradeço também à equipe da Federação Metodista de Jovens da 3ª Região, pela paciência, apoio e pelas orações nessa fase.

Fabio:

Agradeço a Deus pelo dom da vida, e pela alegria de viver. Aos meus familiares mãe, pai, e irmãs(ãos) pelo incentivo, carinho e paciência que tiveram comigo e por compreenderem que minha falta de tempo era por um objetivo maior. Agradeço de maneira muito especial a professora Lúcia Contente Mós por ser nossa orientadora por ter aberto mão de seu tempo em nosso favor, e a todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Larissa:

Agradeço a minha mãe, Maria Inês por ter me auxiliado e me apoiado em todo esse tempo. Agradeço também ao André Felipe que também tanto me ajudou e teve muita paciência e perseverança. Também preciso agradecer a Silvia Maria pelas preocupações e a equipe da empresa Umanni, onde mais aprendi e conheci muitos amigos.

Patrícia:

Agradeço a Deus por ter me dado força para superar todos os obstáculos que tive em meu caminho durante todo o período acadêmico. Aos meus filhos e meu esposo por me substituírem nos afazeres de casa, dando-me a oportunidade de concluir meus estudos. Ao amigo espiritual Dr. Lameira de Andrade que sempre me confortou nos meus momentos difíceis.

William:

Agradeço à minha mãe, Maria Luiza, que durante todo o percurso me incentivou a continuar em frente. A um de meus grandes amigos, Emanuel Oliveira, que me manteve animado nos momentos de fraqueza. Devo também um agradecimento pessoal às instituições que contribuíram para meu desenvolvimento pessoal e profissional: "Cenlep – Centro Nosso Lar de Educação Profissional", "Etec Zona Leste" e "Faculdade Carlos Drummond de Andrade". E às empresas pelas quais passei até então, onde aprendi sobremaneira e tive contato com excelentes profissionais: "totalCAD" e "Função Informática". Um agradecimento extra à empresa francesa "Ubisoft", desenvolvedora do jogo "*Prince of Persia: The Sands Of Time*", responsável por despertar em mim o interesse por todo o mundo tecnológico, e posterior ingresso em cursos das instituições já comentadas.

*“Cada pessoa que passa em nossa vida, passa sozinha, é porque cada pessoa é única e nenhuma substitui a outra. Cada pessoa que passa em nossa vida passa sozinha e não nos deixa só porque deixa um pouco de si e leva um pouco de nós. Essa é a mais bela responsabilidade da vida e a prova de que as pessoas não se encontram por acaso.”*

*Charles Chaplin*

AMANCIO, Larissa; GOMES, Fabio Santos; MARQUES, Carolina Trindade; OLIVEIRA, Patrícia de Fátima; SANTOS, William Barbosa dos; Desenvolvimento de um sistema para gerenciamento da grade de aula para o corpo docente. Trabalho de Conclusão do Curso de Bacharel em Ciência da Computação, Faculdade Carlos Drummond de Andrade, São Paulo, 2013, Orientadora: MOS, Lucia Contente.

## **RESUMO**

É um problema muito comum em faculdades e escolas fazer a alocação dos professores de acordo com as necessidades da instituição, do próprio professor e das turmas para criar uma grade horária. A partir disso, este trabalho acadêmico foi desenvolvido em cima deste mesmo problema a fim de saná-lo o máximo possível com uso de algoritmos genéticos. A forma tradicional de solução deste problema é manual sendo assim demanda de tempo, esforço e muitas vezes pode gerar um resultado fora do esperado. A solução proposta aqui é realizada em uma plataforma Windows e inspirados na teoria da seleção natural proposta por Darwin, denominada como algoritmos genéticos.

**PALAVRAS CHAVES:** Algoritmos genéticos. Alocação de professores. Grade Horária.



AMANCIO, Larissa; GOMES, Fabio Santos, MARQUES, Carolina Trindade, OLIVEIRA, Patrícia de Fátima, SANTOS, William Barbosa dos; Development of a System for Managing Grid Class for College. Course Completion Work for Computer Science Bachelor Course, Carlos Drummond de Andrade College, São Paulo, 2013 Advisor: MOS, Lucia Contente.

## **ABSTRACT**

Is a very common problem in universities and schools to make the teacher's allocation according to the needs of the institution, teachers and classes, to create a timetable. From this, this academic work was developed over this problem, to fix it the most possible, using genetic algorithms. The traditional way to solve this problem is manual, being a time demand, effort and many times can generate a result out of the expected. This proposed solution is performed in a Windows platform, inspired in the natural selection theory, proposed by Darwin, nominated as Genetic Algorithms.

**KEY WORDS:** Genetic Algorithms. Teacher's Allocation. Timetable.

## LISTA DE FIGURAS

Figura 1 - Relação entre DNA e RNA .....	26
Figura 2 - Processo completo de <i>crossover</i> .....	27
Figura 3 - Dois cromossomos realizando o <i>crossover</i> .....	28
Figura 4 - Exemplo de ator.....	37
Figura 5 - Exemplo de caso de uso .....	37
Figura 6 - Exemplo de Diagrama de Caso de Uso .....	38
Figura 7 - Exemplo de Classe .....	39
Figura 8 - Exemplo de Diagrama de Classes .....	39
Figura 9 - Exemplo de Objeto .....	40
Figura 10 - Exemplo de Diagrama de Objetos .....	40
Figura 11 - Exemplo de Diagrama de Sequência.....	41
Figura 12 - Modelo de Entidade Relacionamento.....	50
Figura 13 - Diagrama de Caso de Uso .....	59
Figura 14 - Diagrama GA.....	66
Figura 15 - Diagrama DAO.....	67
Figura 16 - Diagrama de sequência atualizarProfessor .....	68
Figura 17 - Diagrama de sequência atualizaCurso .....	69
Figura 18- Diagrama de sequência DelCurso.....	70
Figura 19 - Diagrama de sequência InserirCurso .....	71
Figura 20 - Diagrama de sequência DelDisciplina.....	72
Figura 21 - Diagrama de sequência DelTurma .....	73
Figura 22 - Diagrama de sequência DelProfessor .....	74
Figura 23 -- Diagrama de sequência InserirProfessor .....	75
Figura 24 - Diagrama de sequência DelTurno .....	76
Figura 25 - Diagrama de sequência SalvaTurno .....	77
Figura 26 - Diagrama de sequência AddDisciplina.....	78
Figura 27 - Diagrama de sequência AddTurma .....	79
Figura 28 - Diagrama de sequência Gerar Grade .....	80
Figura 29 - Diagrama de navegabilidade .....	81
Figura 30 - Tela de cadastro de cursos.....	86
Figura 31 - Tela de cadastro de turnos .....	87
Figura 32 - Tela de cadastro de Professores.....	88
Figura 33 - Tela de exibição e geração das grades.....	89

## **LISTA DE TABELAS**

Tabela 1 - Exemplo de método de roleta.....	36
Tabela 2 - Tabela com horário escolar de uma turma com cinco dias e quatro períodos.....	44
Tabela 3 - Disposição de horário, professores e turmas.....	45
Tabela 4 - Disposição de salas, turmas e horários.....	45

## **LISTA DE SIGLAS**

3D – Três dimensões

DNA – Ácido Desoxirribonucleico

EUA – Estados Unidos da América

GA – Genetic Algorithms, em português, algoritmo genético

IEC – Computação evolucionária interativa

RNA – Ácido Ribonucleico

UML – Unified Modeling Language, em português, Linguagem de modelagem unificada

## SUMÁRIO

<b>1. INTRODUÇÃO .....</b>	<b>16</b>
<b>1.1. Objetivo .....</b>	<b>16</b>
<b>1.2. Justificativas .....</b>	<b>16</b>
<b>1.3. Problematização .....</b>	<b>17</b>
<b>1.4. Metodologia.....</b>	<b>17</b>
<b>1.5. Limitação de estudo .....</b>	<b>18</b>
<b>1.6. Estrutura do trabalho .....</b>	<b>18</b>
<b>2. FUNDAMENTAÇÃO TEÓRICA .....</b>	<b>19</b>
<b>2.1. Algoritmos.....</b>	<b>19</b>
2.1.1. Tipos de informações-----	20
2.1.2. Tipos de dados-----	21
2.1.3. Tipo de inteiros-----	21
2.1.4. Tipos Reais -----	21
2.1.5. Tipos caracteres-----	21
2.1.6. Tipos lógicos -----	22
2.1.7. O uso de variáveis-----	22
2.1.8. Uso de constantes-----	23
2.1.9. Português estruturado-----	23
<b>2.2. Teoria da evolução .....</b>	<b>24</b>
<b>2.3. Genética Básica.....</b>	<b>25</b>
<b>2.4. História do algoritmo genético .....</b>	<b>29</b>
2.4.1. Algoritmo genético-----	30
2.4.1.1. Mutação.....	31
2.4.1.2. Seleção .....	32
2.4.1.3. Método de torneio.....	32
<b>2.5. Método de roleta.....</b>	<b>35</b>
<b>2.6. UML .....</b>	<b>36</b>

2.6.1.	Diagrama de Caso de Uso	37
2.6.2.	Diagrama de Classes	38
2.6.3.	Diagrama de Objetos	40
2.6.4.	Diagrama de Sequência	41
3.	CONTEXTUALIZAÇÃO DO NEGÓCIO	42
3.1.	Descrição do negócio	42
3.2.	Descrição do cenário atual	42
3.3.	Descrição do problema	43
3.4.	O sistema	47
4.	DESENVOLVIMENTO	48
4.1.	Requisitos	48
4.1.1.	Requisitos Funcionais	48
4.3.2	Requisitos não funcionais	48
4.2.	Modelo de Entidade e Relacionamento	50
4.3.	Dicionário de Dados	51
4.4.	Hardware e Software	56
4.5.	Estrutura de dados do sistema	56
4.6.	Geração da população inicial no sistema	56
4.7.	Utilização de <i>crossover</i> no sistema	57
4.8.	Diagrama de caso de uso	57
4.8.1.	Descrições dos Casos de Uso	60
4.8.1.1.	CDU 01 - Efetuar Login	60
4.8.1.2.	CDU 02 – Cadastrar Turno	60
4.8.1.3.	CDU 03 – Cadastrar Curso	61
4.8.1.4.	CDU 04 – Cadastrar Turma	61
4.8.1.5.	CDU 05 – Cadastrar Disciplina	62
4.8.1.6.	CDU 06 – Cadastrar Professor	63
4.8.1.7.	CDU 07 – Alterar Cadastro de Professor	63
4.8.1.8.	CDU 08 – Alterar Cadastro de Curso	64
4.8.1.9.	CDU 09 - Gerar Grade	64
4.8.1.10.	CDU 10 – Visualizar Grade	65
4.9.	Diagrama de Classe do sistema	66

<b>4.10. Diagrama de Sequência do sistema.....</b>	<b>68</b>
<b>4.11. Diagrama de Navegabilidade .....</b>	<b>81</b>
<b>4.12. Softwares Utilizados.....</b>	<b>82</b>
<b>4.12.1. SQLite-----</b>	<b>82</b>
<b>4.12.2. C#-----</b>	<b>83</b>
<b>4.12.3. .NET-Framework 4.0 -----</b>	<b>83</b>
<b>4.12.4. Requisitos do Sistema: -----</b>	<b>85</b>
<b>4.12.5. Hardware Utilizado:-----</b>	<b>85</b>
<b>4.13. Protótipos de tela do sistema .....</b>	<b>85</b>
<b>4.13.1. Cursos .....</b>	<b>86</b>
<b>4.13.2. Turnos .....</b>	<b>87</b>
<b>4.13.3. Professores .....</b>	<b>88</b>
<b>4.13.4. Grade.....</b>	<b>89</b>
<b>CONSIDERAÇÕES FINAIS .....</b>	<b>90</b>
<b>IMPLEMENTAÇÕES FUTURAS .....</b>	<b>91</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>92</b>
<b>REFERÊNCIAS WEBGRÁFICAS.....</b>	<b>93</b>

## **1. INTRODUÇÃO**

O conceito de otimização define-se como a ação de produzir condições apropriadas para o melhor desenvolvimento, com um mecanismo de análise de decisões complexas, envolvendo seleção de variáveis, com o objetivo de qualificar desempenho e medir a qualidade das decisões.

Os Algoritmos Genéticos são aplicados como uma técnica de procura e vem ganhando popularidade com inúmeros trabalhos de pesquisadores em aplicações diversas. São inspirados no princípio Darwiniano da evolução das espécies e na genética. São algoritmos probabilísticos que fornecem um mecanismo de busca paralela e adaptativo baseado no princípio de sobrevivência dos mais aptos e na reprodução.

### **1.1. Objetivo**

O objetivo deste projeto, esta em auxiliar os professores na organização da grade de disciplinas, além de gerenciar uma base de dados ali contida toda a informação. Foi observado que o quadro de aulas dos professores no sistema fornecido pela faculdade apresenta uma demora considerável na postagem das informações de disciplinas que serão lecionadas por cada membro do corpo docente.

Auxiliar na organização de aulas de forma rápida e eficaz, onde o sistema se encarregará de distribuir as disciplinas conforme cada informação disposta no sistema.

### **1.2. Justificativas**

Atualmente o cenário observado em universidades e escolas é de muitos dias para que seja feita a grade definitiva de horários de aulas. Com isso, gera-se um estresse nas primeiras semanas de aulas, que acabam por serem confusas para os alunos, que muitas vezes não sabem quais são as aulas que terão, até que seja apresentada a grade oficial.



Com a utilização do sistema, o trabalho manual que leva semanas para ser concluído, será executado em poucos minutos e terá um resultado preciso e correto.

O sistema será capaz de gerar automaticamente a grade disciplinar utilizando dados de entrada e fará as combinações, de acordo com o que foi inserido previamente.

### **1.3. Problematização**

A cada início de ano letivo, ou ainda, início de cada semestre é muito comum o erro na criação da grade de horários para professores e turmas, seja em uma universidade ou na escola comum. Muito embora esses não sejam os únicos lugares a usar grade de horários, pois existem instituições que usam o mesmo recurso e por vezes enfrentam o mesmo problema, como é o caso dos hospitais, fábricas, indústrias e o próprio comércio, que precisam manter os horários corretos das jornadas de trabalho em cada departamento.

No presente trabalho o foco está na grade de horário de professores, onde o problema apresentado está justamente em colocar um professor para lecionar com várias turmas em dias e horários diferentes, havendo ainda questões como preferências e restrições. Por exemplo: Não é possível um professor lecionar em uma turma em determinado dia e horário e ao mesmo tempo com outra turma no mesmo dia e horário. Ou, o professor só pode lecionar determinado dia e horário.

O erro na elaboração de uma grade correta pode causar a mudança de salas e horários no meio do período letivo. Portanto, vê-se como ideal a criação de um sistema que ordene a distribuição destas turmas de forma eficaz.

### **1.4. Metodologia**

Para que este trabalho pudesse ser realizado utilizou-se de alguns métodos de pesquisa bibliográfica, sendo constituído principalmente por livros, artigos, apostilas e materiais que se encontram disponíveis na *internet*.

### **1.5. Limitação de estudo**

Como uma das principais limitações de estudos, há a falta de acesso ao sistema pelo professor como usuário, podendo ele realizar as alterações do próprio cadastro.

O sistema também não realiza alocação de salas para as turmas, não possuindo nenhum tipo de gerenciamento sobre salas, quantidade de alunos por turma e laboratórios específicos.

Não é possível atualizar os nomes dos professores, pela forma como foi arquitetado a interação com o usuário é necessário deletar o professor e cadastrá-lo novamente.

### **1.6. Estrutura do trabalho**

O seguinte trabalho foi estruturado em seis capítulos:

O Capítulo I consiste na Introdução, sendo que a mesma apresenta subcapítulos que são a Problematização, Objetivo, Justificativa, Metodologia e a Estrutura do Trabalho.

O Capítulo II apresenta a Fundamentação Teórica, que tem por finalidade apresentar todas as definições e pesquisas que foram utilizadas para a realização deste trabalho.

O Capítulo III consiste na Contextualização do Negócio, nele serão ilustrados o Negócio, o Cenário Atual, a Problemática do Cenário Atual e o Cenário Proposto.

O Capítulo IV consiste no Desenvolvimento do Projeto, onde será explicado como o mesmo foi realizado.

O Capítulo V apresenta as Considerações Finais deste trabalho.

## 2. FUNDAMENTAÇÃO TEÓRICA

### 2.1. Algoritmos

Neste capítulo, entende-se o que são algoritmos? Qual a função de um algoritmo dentro do programa de computador e suas funcionalidades? Estas informações ajudarão a entender os próximos contextos.

Segundo, CORMEN (2002). Informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. Portanto um algoritmo é uma sequência de passos computacionais que transformam a entrada na saída.

Mas Segundo, SALVENTTI, Douglas e BARBOSA, Lisbete (1998, p. 5 e 6) Um algoritmo, intuitivamente, é uma sequência finita de instruções ou operações básicas (operações definidas sem ambiguidade e executáveis em tempo finito dispondo-se apenas de lápis e papel) cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância. A ordenação da sequência de instrução do algoritmo apoia-se na estratégia estabelecida durante a análise do problema. O desenvolvimento do algoritmo não pode perder de vista os tipos de dados considerados e a sua representação.

Também pode visualizar um algoritmo como uma ferramenta para resolver um "problema computacional" bem especificado. O enunciado do problema especifica em termos, o relacionamento entre entrada e a saída desejada. O algoritmo descreve um problema computacional específico para alcançar esse relacionamento da entrada com a saída.

Por exemplo, poderia ser necessário ordenar uma sequência de números em ordem decrescente. Veja como definir formalmente o problema de ordenação:

Entrada: Uma sequência de  $n$  números  $[a_1, a_2, \dots, a_n]$ .

Saída: Uma permutação (reordenação)  $[a'_1, a'_2, \dots, a'_n]$  da sequência de entrada, tal que  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ .

Dada uma sequência de entrada como  $[31, 41, 59, 26, 42, 58]$ , um algoritmo de ordenação retorna como saída a sequência  $[26, 31, 41, 41, 58, 59]$ . Em uma sequência de entrada como esta, é chamada uma instância do problema de ordenação. Em geral, uma instância de

um problema consiste na entrada (que satisfaz a quaisquer restrições impostas no enunciado do problema) necessária para se calcular uma solução para o problema.

CORMEN (2002). A ordenação é uma operação fundamental para a ciência da computação (muitos programas a utilizam como uma etapa intermediária) e, como resultado, um grande número de bons algoritmos de ordem tem sido desenvolvidos.

Um algoritmo é dito “correto” se, pra cada instância de entrada, ele parar com a saída correta. Dito que um algoritmo correto “resolve” o problema computacional dado. Um algoritmo incorreto pode não parar em alguma instância de entrada, ou então pode parar como outra resposta que não a desejada. Um algoritmo pode ser especificado em linguagem comum, como um programa de computador, ou mesmo como um projeto de *hardware*. O único requisito é que a especificação deve fornecer uma descrição precisa do procedimento computacional a ser seguido.

A ordenação não é de modo algum o único problema para o qual foram desenvolvidos algoritmos. As aplicações práticas de algoritmos são onipresentes e incluem os exemplos a seguir:

- No Projeto Genoma Humano, no armazenamento das informações coletadas e armazenadas e bancos de dados;
- Na manipulação do alto volume de dado contido na *internet*;
- Na capacidade de manter privativas as informações do comércio eletrônico;
- Nas indústrias;
- Entre outros;

### **2.1.1. Tipos de informações**

Segundo MANZANO e OLIVEIRA, (2005). Antes de iniciar o estado de programação é necessário considerar que um computador nada mais é do que uma ferramenta utilizada para solucionar problema que envolva a manipulação de informações, sendo que essas informações classificam-se genericamente em dois tipos básicos: “dados de instrução”.

### **2.1.2. Tipos de dados**

Segundo MANZANO e OLIVEIRA, (2005). Os dados são representados pelas informações a serem tratadas (processos) por um conjunto. Essas informações estão caracterizando por três tipos de dados, a saber: dados numéricos (inteiros e reais), dados caracteres e dados lógicos.

### **2.1.3. Tipo de inteiros**

Segundo MANZANO e OLIVEIRA, (2005). São caracterizados como tipos inteiros os dados numéricos positivos ou negativos, excluindo-se destes qualquer número fracionário. Como exemplo deste tipo de dados têm os valores: 35, 0 -56 entre outros.

### **2.1.4. Tipos Reais**

São caracterizados como tipos reais os dados numéricos positivos, negativos e números fracionários. Como exemplo deste tipo de dado têm-se os valores: 35, 0 -56, 1.2, -45.897, entre outros.

### **2.1.5. Tipos caracteres**

Segundo MANZANO e OLIVEIRA, (2005). São caracterizadas como tipos caracteres as sequências contendo letras, números e símbolos especiais. Uma sequência de caracteres deve ser iniciado entre aspas (“”). Este tipo de dados é também conhecido como: alfanumérico, *string*, literal ou cadeia. “Como exemplo “deste tipo de dado, têm-se os valores: “Programação”, “Rua Alfa, 52 Ap. 1”, “Fone: 574-9988”, ”“, “7”, entre outros.

### **2.1.6. Tipos lógicos**

São caracterizados como tipos lógicos os dados com valores verdadeiro e falso, sendo que este tipo de dados poderá representar apenas um dos valores. Ele é chamado por alguns de tipo booleano, devido à contribuição do filósofo e matemático inglês George *Boole* na área da lógica matemática. Para facilitar a citação de um dado do tipo lógico, fica aqui declarado que estes deverão ser apresentados e delimitados pelo caractere ponto (.). Como exemplo deste tipo de dados têm-se os valores: Falso, F,N (para o valor lógico: falso) e Verdadeiro, V e S (para o valor lógico: verdadeiro).

### **2.1.7. O uso de variáveis**

Segundo MANZANO e OLIVEIRA, (2005). Tem-se como definição de variáveis tudo aquilo que é sujeito a variações que é incerto, instável ou inconstante. E quando se fala de computadores, temos que ter em mente que o volume de informações a serem tratadas são grandes e diversificadas. Desta forma, os dados a serem processados serão bastante variáveis.

Todo dados a ser armazenado na memória de um computador deve ser previamente identificado, ou seja, primeiro é necessário saber qual o tipo para depois fazer o seu armazenamento adequado. Estando armazenado o dado desejado, ele poderá ser utilizado e manipulado a qualquer momento.

O nome de uma variável é utilizado para sua identidade e posterior uso dentro de um programa. Sendo assim, é necessário estabelecer algumas regras de utilização das variáveis:

Nomes de uma variável padrão devem ser atribuídos com um ou mais caracteres;

O primeiro caractere do nome de uma variável não poderá ser, em hipótese alguma, um número; sempre deverá ser uma letra;

O nome de uma variável não poderá possuir espaços em branco;

Não poderá ser nome de uma variável em uma palavra reservada a uma instrução de programa;

Não poderão ser utilizados outros caracteres a não ser letras e números;

São nomes válidos de variáveis: NOMEUSUARIO, FONW1, X, DELTA25, Z4, entre outros. São nomes inválidos de variáveis: NOME USUARIO, 1X, FONE#, ESCREVA (considerando que seja esta uma palavra reservada à instrução de uma linguagem, no caso, o nosso “português estruturado”).

#### **2.1.8. Uso de constantes**

Têm-se como definição de constante tudo aquilo que é fixo ou estável. E existirão vários momentos em que este conceito deverá estar em uso.

#### **2.1.9. Português estruturado**

Segundo MANZANO e OLIVEIRA, (2005). Tendo estabelecido os passos anteriores (algoritmo e diagrama de blocos), será efetuada a fase de codificação. Esta fase obedece ao que está definido no diagrama de blocos, pois ele é a representação gráfica da lógica de um programa. Porém, sempre deverá ser relacionado com todas as variáveis que serão utilizadas. Define também o espaço de memória que será necessário para manipular as informações fornecidas durante a execução de um programa.

Desta forma, são utilizadas no exemplo, três variáveis: A, B e X, sendo que deverão ser relacionadas antes de seu uso, estabelecendo-se assim o seu respectivo tipo.

```
programa SOMA_NÚMERO
var
X : inteiro
A : inteiro
B : inteiro
Inicio
    leia A
leia B
```

$X \leftarrow A+B$   
escreva X

## 2.2. Teoria da evolução

Segundo LINDEN, (2012). É importante entender que esta sessão existe, pois os algoritmos genéticos são baseados na teoria da evolução.

Até o século XIX os cientistas mais proeminentes acreditavam em uma, dentre as teorias do criacionismo “(Deus criou o universo)” ou da geração espontânea “(a vida surge de essências presentes no ar)” em torno do século XIX para o século XX, Charles Darwin fez uma longa viagem no navio HMS Beagle, visitando vários locais e a sua grande habilidade de observação permitiu que ele percebesse que animais de uma espécie eram ligeiramente diferentes que seus parentes em outros ecossistemas, sendo mais adaptados às necessidades e oportunidades oferecidas pelo seu ecossistema específico.

A teoria da evolução diz que na natureza todos os indivíduos dentro de um ecossistema competem entre si por recursos limitados, tais como comida e água. Aqueles dentre os indivíduos (animais, vegetais, etc.) de uma mesma espécie que não obtém êxito tendem a ter uma prole menor e esta descendência reduzida, tendo a probabilidade de ter os genes prolongados para as novas gerações num índice menor, este processo é denominado Seleção natural.

A combinação entre as características dos indivíduos que sobrevivem podem produzir um novo indivíduo muito mais bem adequado às características de seu meio ambiente a mesclar estas características. Este processo implica nos descendentes de indivíduos serem a variação dos seus pais.

Segundo LINDEN, (2012). Entretanto, esta evolução natural não é um processo dirigido, com intuito de maximizar algumas características das espécies. Na verdade, não existe nenhuma garantia que estas descendências de pais muito bem adaptados também o sejam.

Pode-se afirmar que a evolução é um processo no qual os seres vivos são alterados por um conjunto de modificações que eles sofrem através dos tempos, podendo ser explicada por



alguns fatores como mutação genética (que será abordado nos capítulos seguintes), recombinação genética, seleção natural e isolamento.

### 2.3. Genética Básica

Quando Darwin afirmou que a evolução e a seleção natural faziam com que as espécies fossem se adaptando naturalmente ao meio ambiente, ele não sabia quais eram os mecanismos básicos através dos quais esta adaptação acontecia, pois o processo de transmissão de informação genética ainda era desconhecido. No início do século XIX, um padre chamado Gregor Mendel compreendeu que este processo de transmissão de características positivas estava associado à uma unidade básica de informação, o Gene.

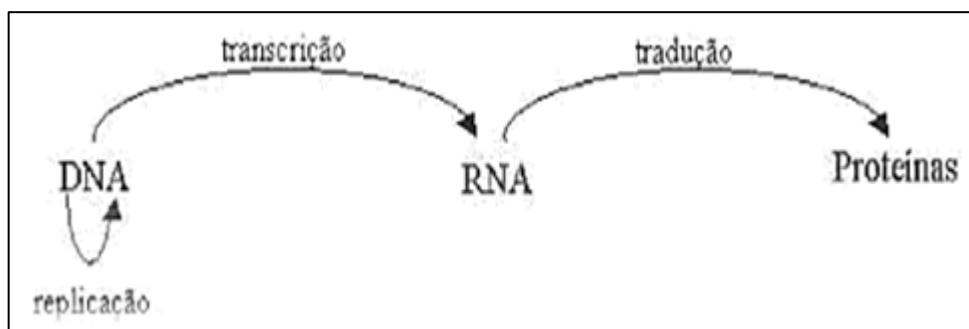
O processo levou à descoberta de como estas características eram fisicamente armazenadas dentro da célula, levou quase um século para ser concluído. No século XIX, o bioquímico suíço Friedtich Mieschner chegou à conclusão de que os núcleos celulares possuem várias substâncias específicas que podiam ser separadas em duas categorias principais: as proteínas e as moléculas ácidas. Substâncias desconhecidas na época que foram denominados “Ácidos Nucléicos”.

Segundo LINDEN, (2012). No século XIX, um químico natural da Rússia, Phoebus A. t. Levene identificou corretamente as riboses como açúcar de um dos dois tipos de ácidos nucleicos, o ácido ribonucleico, e certos componentes de outro ácido nucleico, o ácido desoxirribonucleico (a qual descobriu em 1929), além de identificar corretamente a estrutura básica no DNA da por fosfato-açúcar-base.

Basicamente, todo indivíduo, seja ele animal, vegetal ou mesmo organismos inferiores como vírus e bactérias, é formado por uma ou mais células, e dentro de cada uma delas o organismo possui uma cópia completa do conjunto de um ou mais cromossomos que descrevem o organismo, conjunto denominado genoma.

Os cromossomos vem em pares, sendo que o número de pares ( $n$ ) varia para cada espécie. Os seres humanos possuem 23 pares de cromossomos por células. Já os burros possuem 31 e as carpas 52 pares. Um cromossomo consiste de genes, que são blocos de sequências do DNA, sendo que cada gene é uma região do DNA que tem uma posição específica no cromossomo, chamada *locus*.

Segundo LINDEN, (2012). A relação entre o DNA e as proteínas, é descrita pelo dogma central da biologia. Este afirma que uma sequência de DNA, que contém toda a informação necessária para se autorreplicar, é transcrita em RNA e esta, posteriormente traduzida para a proteína, como é visto na figura 1.



**Figura 1 - Relação entre DNA e RNA**

(FONTE: LINDEN, 2012)

Um conjunto específico de genes no genoma é chamado de genótipo. O genótipo é a base do fenótipo, que por sua vez é a expressão das características físicas e mentais codificadas pelos genes e modificadas pelo ambiente, como cor dos olhos, inteligência etc. Assim pode-se concluir que o DNA do ser humano codifica toda a informação necessária para descrevê-lo, mas esta informação está sob o controle de uma grande rede de regulação genética que, associada às condições ambientais, geram as proteínas nas quantidades certas.

Segundo LINDEN, (2012). Tudo isso se perderia se não puder transmiti-la de geração para geração. Esta transmissão é realizada através da reprodução existente na natureza de formas distintas:

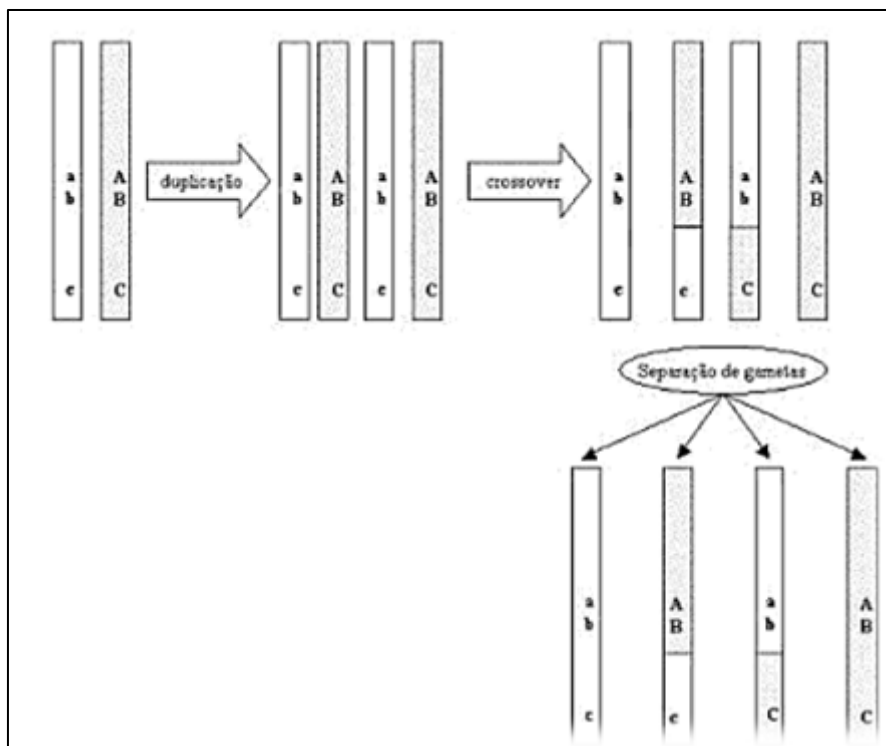
Assexuada: típicas de organismos inferiores, como bactérias;

Sexuada: exige a presença de dois organismos para que ocorra, na maioria dos casos de sexo oposto, para que aja a troca de material genético;

A reprodução sexuada é à base dos GAs.

Nos organismo que utilizam a reprodução sexuada, como os humanos e as moscas cada progenitor fornece um pedaço de material genético chamado gametas. Estes gametas são resultado de um processo denominado *crossing-over* ou *crossover*, demonstrado na figura 2, que permitem que os filhos herdem características de seus pais.

Processo inicia com a duplicação dos cromossomos. Após serem duplicados, os cromossomos realizam o *crossover*, processo no qual um pedaço de cada cromossomo é trocado como seu par. Após este processo, tem-se 4 cromossomos potencialmente diferentes que são separados para os gametas, conforme figura 2.



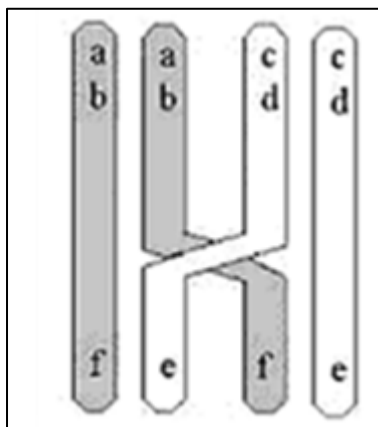
**Figura 2 - Processo completo de *crossover***

(FONTE: LINDEN, 2012)

O processo *crossover* recebe este nome porque fisicamente um cromossomo se cruza sobre o outro para realizar a operação, conforme pode ser visto na figura 2. O cruzamento mostrado troca os cromossomos em apenas um ponto, mas ele foi assim desenhado apenas para facilitar a compreensão. Na realidade, os cromossomos podem ser cruzados em vários pontos e trocar vários genes ao mesmo tempo.

Segundo LINDEN, (2012). É importante considerar também que o processo de replicação do DNA é exatamente complexo. Pequenos erros podem ocorrer ao longo do tempo, gerando mutação dentro do código genético. Além dos erros, estas mutações podem ser causadas por fatores aleatórios, tais como, a presença de radiação ambiente, causando pequenas mudanças nos genes dos indivíduos. Estas mutações podem ser boas ou ruins ao organismo exposto.

Felizmente existem mecanismos de correção que garantem que a taxa de mutação seja muito baixa. Afinal, as células estão sobre ação de fatores exógenos que as induzem à mutação e erros o tempo inteiro. Conforme figura 3.



**Figura 3 - Dois cromossomos realizando o *crossover***

(FONTE: LINDEN, 2012)

Para entender como o maquinário celular é eficiente para manter o DNA intacto, basta comparar taxa de erros entre os vários serviços existentes no mundo.

- Correio nos EUA: 13 entregas atrasadas a cada 100;
- Bagagem de avião: 1 perda a cada 200;
- Datilografia (120 palavras/minuto): 1 erro a cada 250 caracteres;
- Direção nos EUA: 1 morto a cada  $10^4$  motoristas por ano;
- Replicação do DNA (sem correção): 1 erro a cada  $10^7$  nucleotídeos;
- Replicação do DNA (com correção): 1 erro a cada  $10^9$  nucleotídeos;

Pode-se dizer que os indivíduos com uma melhor adequação de seu fenótipo ao meio ambiente reproduzem mais. Ao reproduzirem mais, têm a maior chance de passar seus genes para a próxima geração. Entretanto, graças aos operadores genéticos, os cromossomos de seus filhos não serão exatamente iguais aos dos pais, devida a “recombinação e mutação”, mas sim uma combinação de seus genes.

Segundo LINDEN, (2012). Desta forma consegue-se ligar a genética com a teoria da evolução. Esta variação causada pela atuação dos operadores de *crossover* e mutação, é aleatória, logo a evolução natural não é direcionada. Entretanto os indivíduos mais bem

sucedidos tendem a procurar parceiros mais atraentes e bem sucedidos. Logo se combinam sempre boas qualidades genéticas, os genes dos bons indivíduos, combinados através do *crossover* e mutação, tendem a gerar indivíduos ainda mais aptos e assim a evolução natural caminha.

## 2.4. História do algoritmo genético

Segundo LINDEN, (2012). A história do algoritmo inicia-se na década de 40, quando os cientistas começam a tentar inspirar-se na natureza para criar o ramo da inteligência artificial. A pesquisa se desenvolveu, mais nos ramos da pesquisa cognitiva e na compreensão dos processos de raciocínio e aprendizado até o final da década de 50, quando começou a busca por modelos de sistemas genéticos que pudessem gerar soluções candidatas para problemas que eram difíceis demais para resolver computacionalmente.

Uma das primeiras tentativas de se associar a evolução natural a problema de otimização foi feita em 1957, quando Box apresentou seu esquema de operação evolucionário. Estes eram métodos de perturbar de forma sistemática duas, três ou várias variáveis de mutação e seleção (Goldberg, 1990).

Segundo LINDEN, (2012). Uma tentativa de usar processos evolutivos para resolver o problema foi feita por I. Rechenberg, na primeira metade da década de 60, quando ele desenvolveu as estratégias evolucionárias, (*evolutinary strategies*) (Rechenberg, 1965). Esta, mantinha uma população de dois indivíduos com cromossomos compostos de números reais em cada instante, sendo que um dos dois indivíduos com cromossomos compostos de números reais em cada instante, sendo que os dois eram filhos do outro, e era gerado através da aplicação exclusiva do operador de mutação. O processo descrito por Rechenberg tinha ampla fundamentação teórica, sendo que a mutação era aplicada à partir de uma distribuição gaussiana dos parâmetros e foi usado com sucesso em vários problemas práticos.

HOLLAND estudou formalmente a evolução das espécies e propôs um modelo heurístico computacional que quando implementado poderia oferecer boas soluções para problemas extremamente difíceis que eram insolúveis computacionalmente até aquela época. Em 1975 Holland publicou seu livro, “*Adaptation in natural and Artificial Systems*”, no qual faz um estudo dos processos evolutivos, em vez de projetar novos algoritmos, como a maioria

pensa. O trabalho de Holland apresenta os algoritmos genéticos como uma metáfora para os processos evolutivos, de forma que ele pudesse estudar a adaptação e a evolução no mundo real.

Segundo LINDEN, (2012). Um fato interessante quanto ao trabalho de Holland e sua influência na área de GA é que ele usou originalmente cromossomos binários, genes eram apenas zero e um. Esta limitação foi abolida por pesquisadores posteriores, mas ainda hoje muitos cientistas insistem em usar apenas a representação binária, mesmo quando há outras que podem ser mostrar mais adequadas para a resolução do problema.

#### **2.4.1. Algoritmo genético**

Segundo LINDEN, (2012). Algoritmos Genéticos (GAs - *Genetic Algorithms*) constituem uma técnica de busca e otimização, altamente paralela, inspirada no princípio Darwiniano de seleção natural e reprodução genética. Os princípios da natureza nos quais os GAs se inspiram são simples. De acordo com a teoria de C. Darwin, o princípio de seleção privilegia os indivíduos mais aptos com maior longevidade e, portanto, com maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chance de perpetuarem seus códigos genéticos nas próximas gerações. Tais códigos genéticos constituem a identidade de cada indivíduo e estão representados nos cromossomos.

Estes princípios são imitados na construção de algoritmos computacionais que buscam uma melhor solução para um determinado problema, através da evolução de populações de soluções codificadas através de cromossomos artificiais.

Em GAs um cromossomo é uma estrutura de dados que representa uma das possíveis soluções do espaço de busca do problema.

Cromossomos são então submetidos a um processo evolucionário que envolve avaliação, seleção, recombinação sexual (crossover) e mutação. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos.

#### 2.4.1.1. Mutação

Tendo um conjunto de pessoas que são selecionadas para reprodução a partir de processo de seleção, e depois compostos os filhos, entra em ação o operador de mutação que funciona da seguinte maneira: tem associada uma probabilidade extremamente baixa (em média de 0,5%) e é sorteado um número entre 0 e 1. Se o número sorteado for menor que a probabilidade predeterminada, então o operador trabalha em cima do gene em questão, mudando o valor randomicamente. Então se repete o processo para todos os genes dos dois filhos.

Segundo LINDEN, (2012). O valor da probabilidade que se obtém, se o operador de mutação será ou não aplicador, é um dos parâmetros do algoritmo genético que somente a experiência consegue determinar.

A principal ideia sobre o valor da probabilidade é que o valor deve ser baixo, se for muito alto, o algoritmo genético terá semelhança com uma técnica que se chama *random walk*, onde a solução é determinada de forma aleatória, sorteando os elementos sem passar pelas informações passadas e atuais.

O algoritmo genético normalmente se comporta de forma estranha se a taxa de mutação for colocada em 100%. Dessa maneira, todos os bits do cromossomo serão invertidos e a qualidade da população se degenerará e dificilmente um algoritmo genético conseguirá resolver este problema.

Existem outros textos referentes a este assunto que preferem que o operador de mutação não aja de forma aleatória, apenas se este for selecionado altere o valor do gene para qualquer outro valor válido dentro do alfabeto genético. É facilmente perceptível que o processo citado acima acaba em multiplicar a probabilidade do operador de mutação por  $n/(n-1)$  sendo  $n$ , a cardinalidade do nosso alfabeto genético. Por exemplo, no caso de um alfabeto binário, têm-se apenas dois valores (0 e 1) e assim dobra-se a probabilidade do operador de mutação, se for utilizada esta técnica.

#### **2.4.1.2. Seleção**

Segundo LINDEN, (2012). Em alguns trabalhos de algoritmos genéticos o processo de seleção dos pais não é utilizado e os pesquisadores pulam direto para o processo de roleta viciada. Mas a falta da seleção dos pais para a roleta viciada pode influenciar muito o resultado final, pois dependendo do módulo de seleção, é possível acelerar ou retardar a ocorrência de convergência genética, podendo fazer mais ou menos agressivo no aproveitamento das melhores soluções. Sendo assim, é muito válido pesquisar sobre técnicas alternativas ao método da roleta viciada.

Chama-se de pressão seletiva, a força que o método de seleção faz para empurrar os melhores esquemas nas melhores soluções para a próxima geração. Mede-se a intensidade da pressão seletiva ou intensidade de seleção a partir da melhoria obtida na avaliação média dos indivíduos da população atual que pode ser normalizada com um desvio padrão. Quanto menor for este valor, maior a melhoria na atuação do módulo de seleção dos operadores sobre a última população. No final da execução do algoritmo genético, este valor diminui bastante devido a convergência genética.

Segundo LINDEN, (2012). Um grande problema na construção de um algoritmo genético é a escolha dos pais para a próxima geração de indivíduos. Se forem muito restritivos ao usar-se apenas bons pais, pode-se estar jogando fora ótimas oportunidades que só estão presentes em indivíduos considerados ruins. Em contrapartida, se utilizar somente indivíduos considerados ruins para a criação da nova geração, os esquemas que tornam estes ruins, nunca sumirão da população.

#### **2.4.1.3. Método de torneio**

Como o próprio nome já diz, este método seleciona uma série de indivíduos da população e fazer com que estes entrem em uma competição direta pelo direito de ser pai utilizando como objeto para chegar primeiro a sua avaliação.

Neste método tem-se um parâmetro chamado tamanho do torneio que define quantos indivíduos serão selecionados aleatoriamente dentro da população que compete. Aquele



dentre os competidores que conseguir uma boa avaliação, é selecionado para uma aplicação do valor genético.

Segundo LINDEN, (2012). O valor mínimo do tamanho do torneiro, que chamaremos de  $K$ , é igual a 2, se não fosse 2, não haveria competição. Teoricamente não há limite de valores para este parâmetro, mas se for escolhido um número igual o número da população, que se chama de  $N$ , só tem-se um único vencedor sempre, sendo o melhor de todos os indivíduos. E se for escolhido um valor muito alto, os  $N-K$  indivíduos vão sempre predominar, tendo em vista que sempre um destes será o vencedor do torneio.

Os indivíduos são selecionados para este torneio de forma aleatória, sem favorecimento para nenhum indivíduo, como no caso da roleta viciada. A única vantagem que os indivíduos podem ter é que se os melhores forem selecionados, acabarão sendo os vencedores do torneio.

#### Exemplo de aplicações – Setor petrolífero

Segundo LINDEN, (2012). Em todo o mundo o setor petrolífero é um dos mais rentáveis do mundo que oferece diversas oportunidades de pesquisa em todas as suas ramificações, incluindo as áreas que possuem difíceis problemas de otimização, suscetíveis a diversos tipos de aplicações de algoritmos genéticos.

#### Sendo um exemplo: Inversão sísmica

Segundo LINDEN, (2012). No campo da geologia, é de extrema importância o problema da inversão sísmica e consiste em determinar a estrutura dos dados do subsolo a partir de uma pesquisa geológica, tendo como objetivo primário obter um modelo 3d ou uma seção geológica.

Sendo um problema que possui irregularidades em sua função, é totalmente não linear tendo também muitos mínimos e máximos, além de descontinuidades, é um ótimo exemplo para uma aplicação com algoritmos genéticos. Além disso, este problema tem alguns aspectos em especial que são muito interessantes como a sensibilidade da solução obtida a condições iniciais distintas. Estes problemas são sempre resolvidos com o conhecimento dos geólogos, sendo assim é de grande importância levá-los em consideração.

WIJNS 2003. Apresenta um algoritmo genético muito interessante para a solução deste problema em que “a função de avaliação computacional do seu algoritmo é substituída por um analista humano (geólogo) que ordena as soluções de acordo com a sua adequação. A justificativa por trás deste modelo é a dificuldade de se quantificar numericamente, de forma absoluta, a qualidade de uma solução para o problema da inversão, mas a possibilidade racional de distinguir a melhor dentre duas soluções”. (WIJNS 2003 apud LINDEN, 2012)

Segundo LINDEN, (2012). Este caso demonstra capacidades de uma área chamada Computação Evolucionária Interativa (IEC) que, como o próprio nome sugere, tem como base o conhecimento de especialistas para avaliar as soluções que foram propostas reduzindo a quantidade de soluções que podem ser vistas.

No caso do problema de inversão, que depende de parâmetros contínuos, o algoritmo genético utilizado, usa uma representação de um cromossomo real ou mais próximo da realidade possível ao invés de uma representação booleana.

O método de seleção que é utilizado aqui é baseado em uma função de avaliação onde é aplicada uma normalização linear que serve para diminuir os efeitos de superindivíduos na população. Já o operador de mutação escolhido para esta situação seleciona aleatoriamente o valor dentro dos limites selecionados antes para se tornar o substituto do gene escolhido anteriormente.

Segundo LINDEN, (2012). Uma coisa interessante nesta implementação é o fato de que este utiliza uma função de avaliação que vai mudando com o passar do tempo. Nas primeiras gerações, a função de avaliação terá um reflexo de baixa importância e depois que a população se adapta as condições submetidas e a importância é aumentada. Sendo assim, o tempo do algoritmo diminui e obtém-se também um resultado melhor do mesmo.

Segundo LINDEN, (2012). Outro ponto interessante deste algoritmo é o fato de que os resultados obtidos não têm boas melhorias se uma geração (geralmente por volta da centésima), tendo um esforço computacional perdido se continuar com este algoritmo por mais centenas de gerações como foi feito. Mas isso não significa que o algoritmo deve ser parado toda vez que chegar à centésima geração, mas sim que é necessário acrescentar um critério de parada na falta da melhora da função de avaliação.

## 2.5. Método de roleta

O princípio básico do funcionamento dos GAs é que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de seleção é projetada para escolher preferencialmente indivíduos com maiores notas de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. Um método de seleção muito utilizado é o Método da Roleta, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta.

Neste método, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta. Finalmente, a roleta é girada um determinado número de vezes, dependendo do tamanho da população, e são escolhidos, como indivíduos que participarão da próxima geração, aqueles sorteados na roleta. (<http://www.icmc.usp.br/pessoas/andre/research/genetic/>)

Segundo LINDEN, (2012). Quando é montada uma roleta para uma determinada população, somamos todas as avaliações e para cada indivíduo é alocado um pedaço igual à avaliação deste indivíduo dividida pela soma das avaliações de todos os indivíduos. O que aconteceria então se tivesse um ou mais indivíduos com avaliação negativa? A resposta é: a soma total ainda seria  $360^\circ$ , mas a soma dos espaços alocados apenas para os de avaliação positiva excederia  $360^\circ$ .

Em uma situação hipotética, como uma função de avaliação  $f(x)=x$  e o domínio da função  $f$  é o intervalo  $[-20,20]$ . Conforme o exemplificado na tabela 1.

Tabela 1 - Exemplo de método de roleta

<i>Indivíduo</i>	<i>Avaliação</i> $f(x)=x$	<i>Pedaço da</i> <i>roleta %</i>	<i>Pedaço da</i> <i>roleta em (°)</i>
1	1	<b>6,25</b>	<b>22,5</b>
-5	-5	-31,25	-112,5
20	20	<b>125</b>	<b>450</b>
<i>Total</i>	<i>16</i>	<i>100.00</i>	<i>360.00</i>

FONTES: LINDEN, (2012)

## 2.6. UML

UML (*Unified Modeling Language*), Linguagem de modelagem de *softwares* que utiliza objetos e diagramas, para que seja representada visualmente e é usada de forma internacional pela Engenharia de *Software*.

Segundo (GUEDES, 2011). A modelagem de *software* consiste na criação de modelos de *software*, sendo uma abstração do sistema, a fim de determinar o que deve ser incluído, e descrever aspectos estruturais ou comportamentais.

A UML auxilia na identificação de Requisitos, Comportamento, Estrutura Lógica e Física e Processos do sistema que será criado e implantado.

A utilização da UML ocorre antes do desenvolvimento do sistema, gerando diversos documentos e diagramas que auxiliarão na construção do sistema.

Antes de existir a UML, existiam três métodos de modelagem, que eram: Booch, OMT e OOSE, muito populares em meados da década de 1990. Com a união dos três métodos, foi gerada em 1996 a primeira versão da UML e em 2005, foi lançada a versão 2.0 do projeto. É possível consultar a versão oficial da UML através do site: <<http://www.uml.org/>>.

Segundo (GUEDES, 2011). Explica que a UML é composta por uma série de diagramas, que podem dar diversos tipos de visões sobre o sistema que será modelado. Cada tipo de diagrama tem uma determinada informação e objetivo.

### 2.6.1. Diagrama de Caso de Uso

É gerado nas fases de levantamento de requisitos e apresentará o comportamento do sistema. É constituído pelos atores, casos de uso e suas ligações, que podem ser setas variadas, e cada uma com uma função.

A figura 4 mostra o exemplo de um ator:

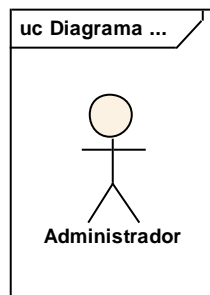


Figura 4 - Exemplo de ator

FONTE: GUEDES, (2011)

A figura 5 mostra o exemplo de um caso de uso:

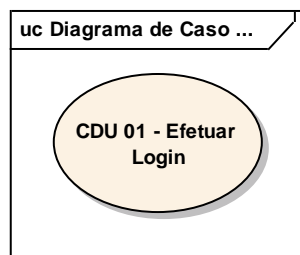


Figura 5 - Exemplo de caso de uso

FONTE: GUEDES, (2011)

Na figura 6, um exemplo de um diagrama de caso de uso de um Sistema de Controle Bancário:



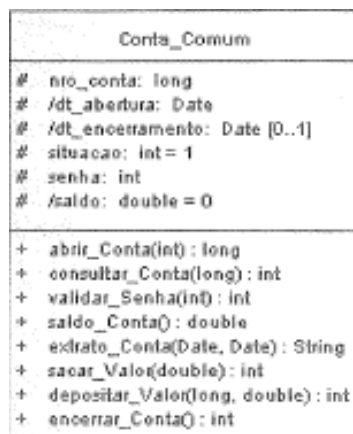


Figura 7 - Exemplo de Classe

FONTE: GUEDES, (2011)

A figura 8 exemplifica um Diagrama de Classes:

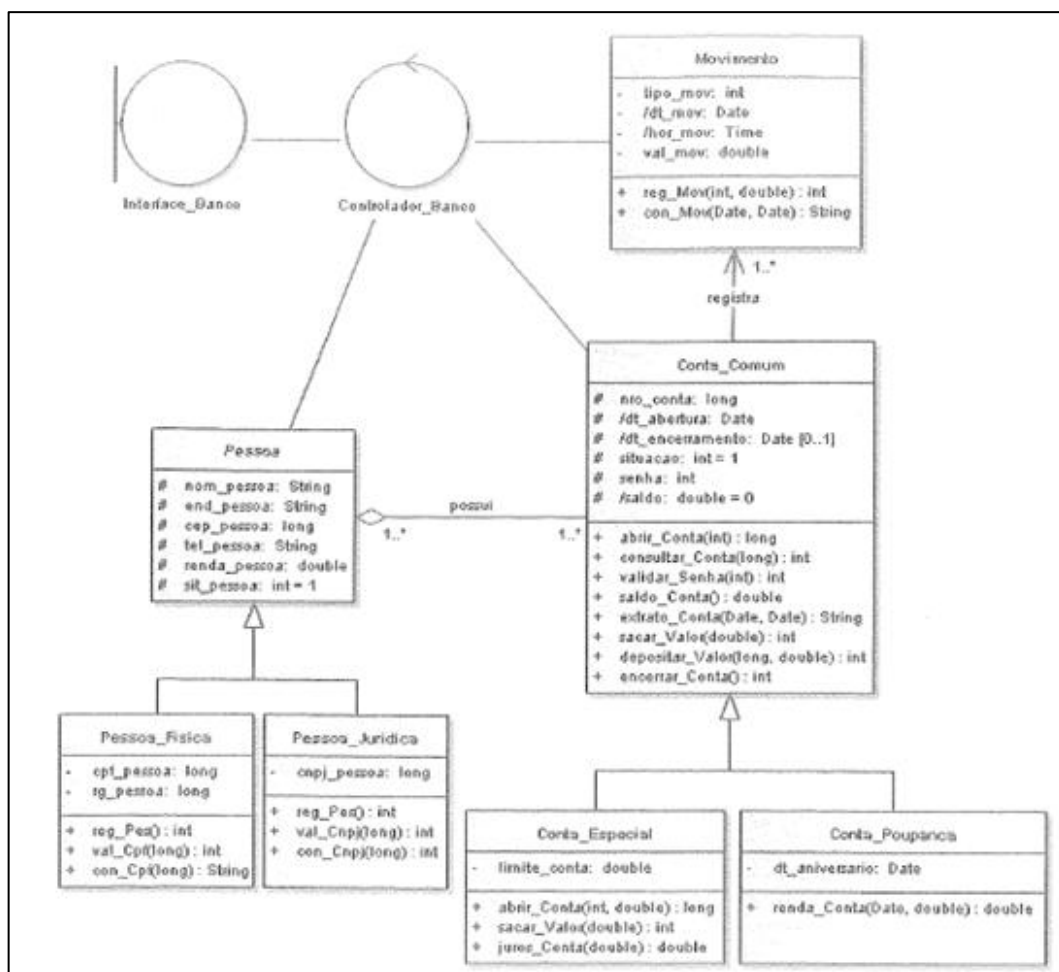


Figura 8 - Exemplo de Diagrama de Classes

FONTE: GUEDES, (2011)

### 2.6.3. Diagrama de Objetos

O Diagrama de Objetos complementa e depende do Diagrama de Classes, pois mostra os valores armazenados dentro de cada objeto da classe.

A figura 9 mostra um exemplo de Objeto:

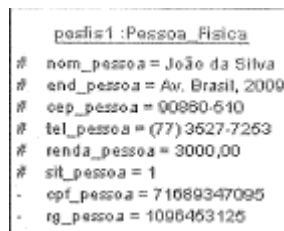


Figura 9 - Exemplo de Objeto

FONTE: GUEDES, (2011)

A figura 10 apresenta um exemplo de Diagrama de Objetos:

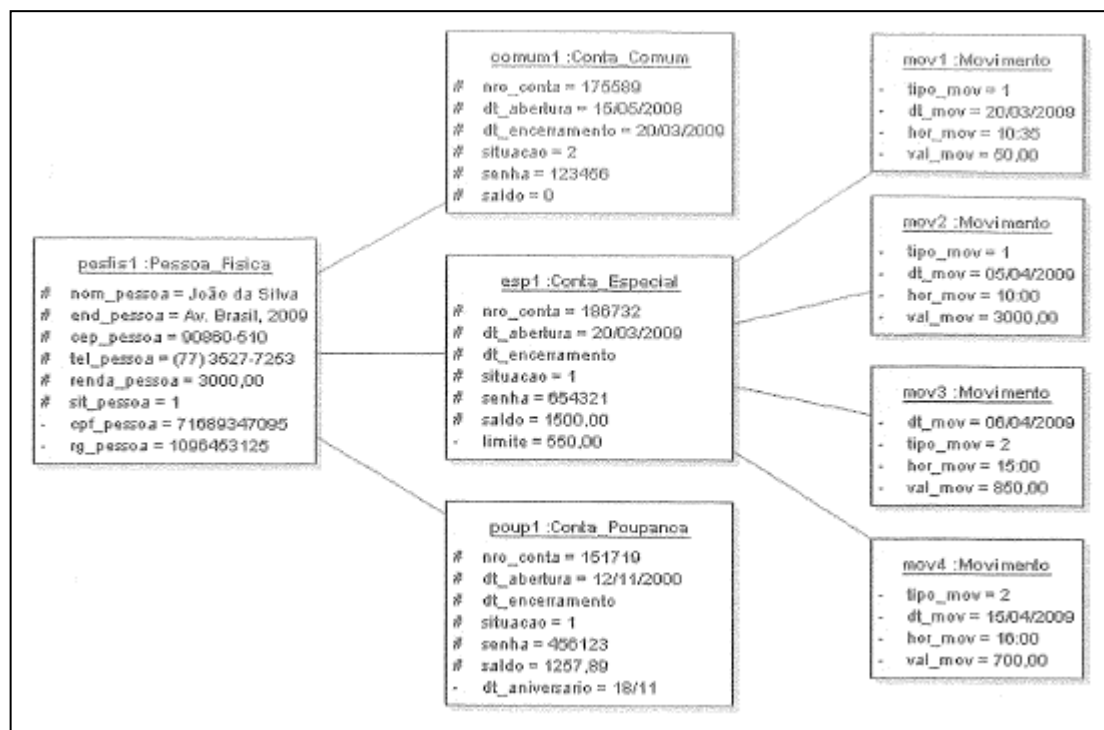


Figura 10 - Exemplo de Diagrama de Objetos

FONTE: GUEDES, (2011)



#### 2.6.4. Diagrama de Sequência

Diagrama de Sequência consiste no comportamento e ordem de ação dentro do sistema. Depende do Diagrama de Caso de Uso e do Diagrama de Classes, para que possa ser bem estruturado, pois utiliza os atores do Diagrama de Caso de Uso e as classes do Diagrama de Classes, para determinar como serão feitos os processos dentro do sistema e quais métodos e objetos serão disparados e quais os retornos que serão apresentados.

A figura 11 exemplifica um Diagrama de Sequência:

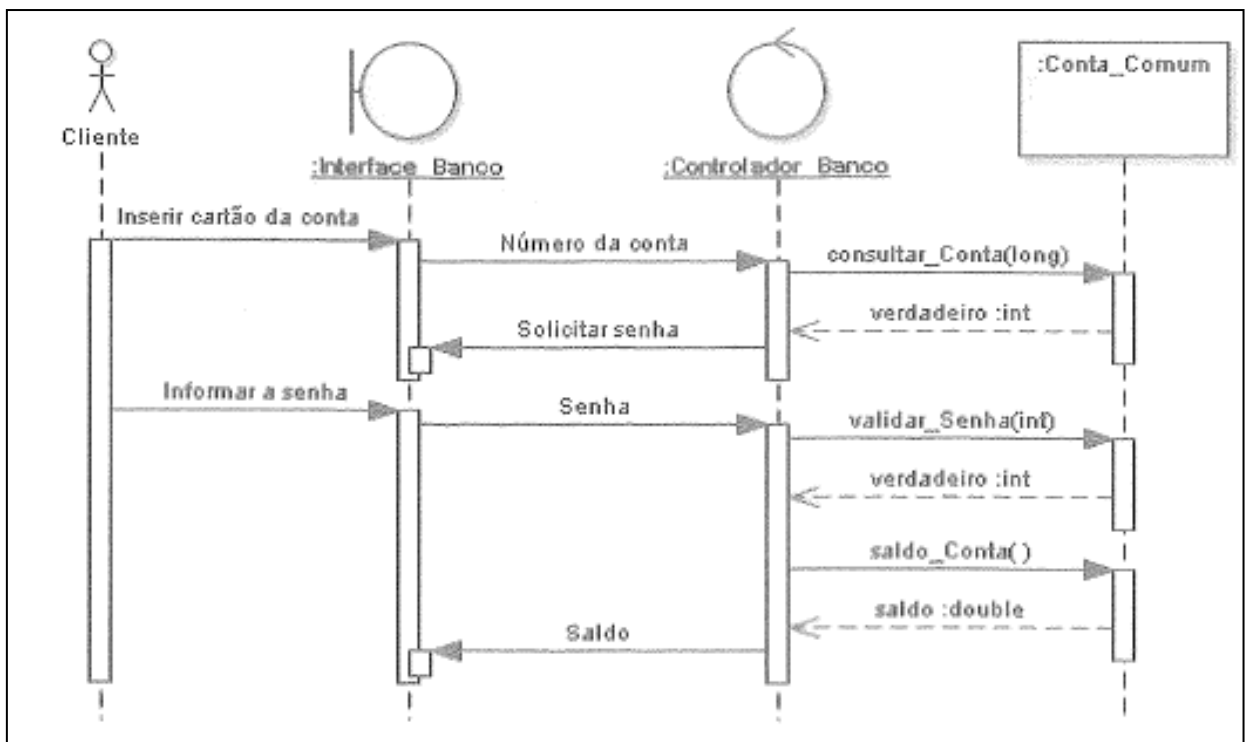


Figura 11 - Exemplo de Diagrama de Sequência

FONTE: GUEDES, (2011)

### **3. CONTEXTUALIZAÇÃO DO NEGÓCIO**

#### **3.1. Descrição do negócio**

O presente trabalho consiste em desenvolver o Sistema Kairos que criará a grade de aulas para o corpo docente. O sistema será desenvolvido em linguagem C#, utilizando a plataforma de desenvolvimento *.NET Framework 4.0* e os Algoritmos Genéticos, inspirado na Teoria da Seleção Natural proposta por Darwin.

O Sistema Kairos usará os dados inseridos pelo usuário para gerar a grade de horários de maneira eficaz.

#### **3.2. Descrição do cenário atual**

Não é possível afirmar que exista hoje no mercado uma solução infalível para o problema das grades de aula. O que se pode afirmar é que estão disponíveis na *internet* diversos *softwares* gratuitos e comerciais que prometem solucionar de modo muito fácil a situação.

Dentre algumas soluções destacam-se:

O *Times'Cool* é um serviço online para professores que desejam organizar suas rotinas de turmas e horários. O interessante neste sistema é que ele traz um tutorial online com dicas e o passo-a-passo para usar a ferramenta. É um *software* gratuito que apenas solicita um cadastro para que seja possível sua utilização. Maiores detalhes poderão ser vistos em [<http://timescool.lotimiza.com/>](http://timescool.lotimiza.com/).

O *Asc Timetable* é um *software* proprietário desenvolvido pela *Applied Software e Princeton Systems*, para gerar grades de horários para escolas e universidades. É fornecido sob Licença de Uso individual para uma instituição não limitado pelo número de computadores no local. Dentre suas diversas funcionalidades uma interessante é que ele também verifica se o horário gerado atende a todas as condições impostas. Ao realizar

mudanças manualmente, o programa o informa se houver algum erro. Outras informações sobre o sistema poderão ser obtidas em [www.asctimetables.com/](http://www.asctimetables.com/).

O Cronos é uma ferramenta *on-line* desenvolvida pela empresa Sistema Cronos localizada em Lavras - Minas Gerais, para a criação de grades de horário para instituições de ensino. A licença de uso pode ser adquirida através do *site* do sistema no endereço *on-line* [www.sistematicronos.com.br/](http://www.sistematicronos.com.br/), sendo disponibilizadas de acordo com a quantidade de turma que se deseja gerar grades. Para até cinco turmas o uso é gratuito.

O primeiro algoritmo desenvolvido para este sistema foi apresentado como trabalho de conclusão de curso de Ciência da Computação da Universidade Federal de Lavras (UFLA) e publicado em um evento internacional em 2006.

Atualmente possuem mais de 3000 instituições de ensino cadastradas e a *interface* atual foi apresentada em Junho de 2011 e modernizada em Junho de 2012, bem como evoluções importantes nos algoritmos do Cronos.

### **3.3. Descrição do problema**

O problema da grade de aulas nas instituições de ensino ocorre devido às restrições e preferências por parte dos professores e a quantidade de turmas existentes. Quanto maior a quantidade de turmas, maiores serão os problemas.

KOSTKO et al.(2005), descrevem que a alocação de professores é um problema complexo, que por envolver inúmeras variáveis gera dificuldade para quem o faz e também há uma imensa dificuldade em atender as preferências e necessidades individuais de cada professor.

LIMA JÚNIOR e CORRÊA (2010) descrevem:

“O problema de alocação de professores e disciplinas em grades horárias é um problema de difícil solução, principalmente quando o número de disciplinas e professores é elevado. Este problema é considerado NP-Difícil (Não-Determinístico de tempo polinomial), ou seja, o esforço computacional necessário para a sua resolução cresce exponencialmente com o tamanho do problema (GAREY, 1979). A busca por uma solução manual do problema pode demandar o esforço de muitas pessoas durante vários dias e, mesmo assim, não gerar uma boa solução (PEREIRA, 2001) ”.

PAIM e GREIS (2008a) apresentam este problema como sendo típico na elaboração de horários, que consiste em agendar uma sequência de encontros (aulas, exames) entre professores e alunos em determinado período, satisfazendo um conjunto de restrições de vários tipos.

Um exemplo típico é apresentado nas tabelas que se seguem.

O exemplo da tabela 2 apresenta nas colunas os horários e nas linhas os professores. É possível observar que a disciplina de matemática representada por “Mat”, necessita de 4 horas-aula por semana distribuída em dois dias (terça-feira e quinta-feira) de dois períodos (7:00-9:00 e 8:00-9:00) cada e é ministrada pelo professor Paulo na sala 102.

Tabela 2: Tabela com horário escolar de uma turma com cinco dias e quatro períodos.

Tabela 2 - Tabela com horário escolar de uma turma com cinco dias e quatro períodos.

Turma 1	Segunda	Terça	Quarta	Quinta	Sexta
Turma 2	Segunda	Terça	Quarta	Quinta	Sexta
Turma 3	Segunda	Terça	Quarta	Quinta	Sexta
Turma 4	Segunda	Terça	Quarta	Quinta	Sexta
Período 1 (7:00 – 8:00)	Hist Prof. Paulo Sala 102	Mat Prof. José Sala 303	Bio Prof. Pedro Lab Bio	Mat Prof. José Sala 102	Quim Prof. Cida Lab Quim
Período 2 (8:00 – 9:00)	Hist Prof. Paulo Sala 102	Mat Prof. José Sala 303	Bio Prof. Pedro Lab Bio	Mat Prof. José Sala 102	Quim Prof. Cida Lab Quim
Período 3 (9:00 – 10:00)	Fis Prof. Cida Lab Fis	Port Prof. Carla Sala 303	Geo Prof. Paulo Sala 302	Lit Prof. Carla Sala 102	EdFis Prof. Eloi Ginásio
Período 4 (10:00 – 11:00)	Fis Prof. Cida Lab Fis	Port Prof. Carla Sala 303	Geo Prof. Paulo Sala 302	Lit Prof. Carla Sala 102	EdFis Prof. Eloi Ginásio
Período 5 (11:00 – 12:00)	Ing Prof. Sara Sala 205	Art Prof. Iris Sala 205	Ing Prof. Sara Sala 205	Art Prof. Iris Sala 205	Fil Prof. Paulo Sala 205

FONTE: <http://guaiba.ulbra.br/seminario/eventos/2008/artigos/administracao/376.pdf> Data de Acesso

15/09/2013

No exemplo da tabela 3 as colunas representam os horários e as linhas representam os professores, de tal forma que, um professor y em um horário z está alocado.

Observe que:

- Disponível, fica vazio;
- Indisponível, recebe valor x;
- Lecionando, recebe o valor referente à turma (0,1,2,3..n).

Tabela 3: Mesma tabela de horário onde é indicada a disponibilidade dos professores e aqueles alocados para a turma 4.

Tabela 3 - Disposição de horário, professores e turmas.

Professor	Segunda					Terça					Quarta					Quinta					Sexta				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Paulo	4	4	x	x	x	x	x						4	4		x	x	x	x	x	x				4
José						4	4									4	4								
Pedro	x	x	x	x	x						4	4									x	x	x	x	x
Cida			4	4							x	x	x	x	x	x	x	x	x	x	4	4			
Carla			x	x	x	x		4	4									4	4						
Sara					4				x	x	x	x			4										
Iris										4										4					
Eloi	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								4	4	

FONTE: <http://guaiba.ulbra.br/seminario/eventos/2008/artigos/administracao/376.pdf> Data de Acesso 15/09/2013

Na tabela 4 é utilizada a mesma tabela anterior, só que agora há uma substituição de “professor” por “sala”.

Tabela 4 - Disposição de salas, turmas e horários.

Professor	Segunda					Terça					Quarta					Quinta					Sexta				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Sala 102	4	4														4	4	4	4						
Sala 205					4					4				4						4					4
Sala 302													4	4											
Sala 303						4	4	4	4																
Lab Quim																					4	4			
Lab Fis			4	4																					
Lab Bio											4	4													
Ginásio																							4	4	

FONTE: <http://guaiba.ulbra.br/seminario/eventos/2008/artigos/administracao/376.pdf> Data de Acesso 15/09/2013

A Universidade Federal de Pernambuco em sua página *web* apresenta algumas situações de restrições máximas e moderadas a fim de evitar ao máximo o problema com a elaboração da grade de horários. Por exemplo:

Na situação de restrições máximas citam:

- Não deve haver choque de horários:
  1. De professores em turmas;
  2. De aulas em salas ambientes;
  3. De alunos em aulas (para instituições de ensino que adotam subdivisões de turmas em grupos).
- Não haver reingresso de aula de uma mesma disciplina no mesmo dia a não ser de forma consecutiva.
- Não haver aula vaga para qualquer turma. Se há menos aulas que o possível na grade, os alunos devem sair mais cedo.
- Não haver mais de duas aulas por disciplina, numa mesma série, em um mesmo dia.
- Não haver aulas de uma disciplina em dois dias consecutivos (exceto disciplinas de carga horária alta).

Na situação de restrições moderadas citam:

- Não haver aulas de uma disciplina na sexta-feira e também na segunda-feira.
- Não haver aulas duplas (geminadas) quebradas pelo recreio.
- Não ocorrer mais que três disciplinas por dia para cada turma.
- Não coincidir muitas aulas de disciplinas de uma mesma área em um mesmo dia.
- Prever substituições de professores sem necessidade de elaborar um novo horário.
- Garantir que nenhum professor tenha apenas uma aula em qualquer dia da semana.
- Garantir que nenhum professor tenha mais que cinco aulas consecutivas.
- Não haver "janelas" maiores que dois horários para qualquer professor.
- Garantir horários de encontro por disciplina para os professores.

PAIM e GREIS (2008b) descrevem que a literatura existente sobre o assunto tem apresentado uma grande série de variantes deste problema que se diferenciam pelo tipo de instituição e pelos tipos de restrições.

### 3.4. O sistema

O sistema Kairos é uma ferramenta desenvolvida para facilitar a criação de grades de horários para instituições de ensino. Com essa ferramenta é possível elaborar rapidamente grades de horários usando as informações fornecidas pelos professores dentro de suas preferências e restrições.

As instituições de ensino serão responsáveis por colher as informações dos professores para todas as disciplinas que irão ministrar. O responsável pela criação da grade deverá cadastrar os professores, turmas, disciplinas, turno e horários de acordo com as informações fornecidas.

O cadastro será feito uma única vez e poderá ser atualizado de acordo com a necessidade da instituição e do professor. Haverá ainda a possibilidade de exclusão de todas as informações caso o profissional se desligue da instituição.

O sistema Kairos é um *software* proprietário e pode ser adquirido através da compra de licença de uso com período de utilização de acordo com o solicitado pelo cliente e descrito no contrato (ANEXO1- Modelo de contrato).

## **4. DESENVOLVIMENTO**

### **4.1. Requisitos**

Os requisitos foram levantados levando em consideração o problema de alocação de professores na grade horária das instituições de ensino.

#### **4.1.1. Requisitos Funcionais**

- O sistema deverá permitir o cadastro, alteração e consulta de professor, exibindo o nome, RG, e suas preferências em relação ao dia/turno e às disciplinas a ministrar.
- O sistema deverá permitir o cadastro, alteração e consulta de cursos, exibindo o nome, quantidade de períodos, disciplinas e turmas de um período específico.
- O sistema deverá permitir o cadastro, alteração e consulta de turnos, exibindo descrição de turnos, e horários do mesmo.
- O sistema conterà uma conta de “administrador”, e somente esse administrador poderá alterar os cadastros dos professores, dos cursos e dos turnos. Deverá também haver um usuário “secretária”, que poderá consultar todos os dados cadastrados no sistema e gerar a grade horária.

#### **4.3.2 Requisitos não funcionais**

- A interface com o usuário é de vital importância para o sucesso do sistema. No intuito de tornar o sistema intuitivo para que não haja problemas em sua utilização, as telas possuem poucos elementos, exibindo somente o estritamente necessário para o funcionamento.



- Embora não seja um requisito essencial ao sistema, dado que idealmente o sistema seria usado somente uma vez por período acadêmico, deve ser considerada por corresponder a um fator de qualidade de *software*. Dessa forma o sistema conterá otimizações internas para funcionamento eficaz, baseadas em testes de esforço para garantir sua melhoria.
- Por ser um sistema cliente o aplicativo não dará opção para que os próprios professores alterem seus dados, ficando isso a cargo de uma versão *intraweb* futura, não sendo o foco da versão à qual esse documento se propõe.

## 4.2. Modelo de Entidade e Relacionamento

Na figura 12, é possível visualizar o Modelo de Entidade e Relacionamento.

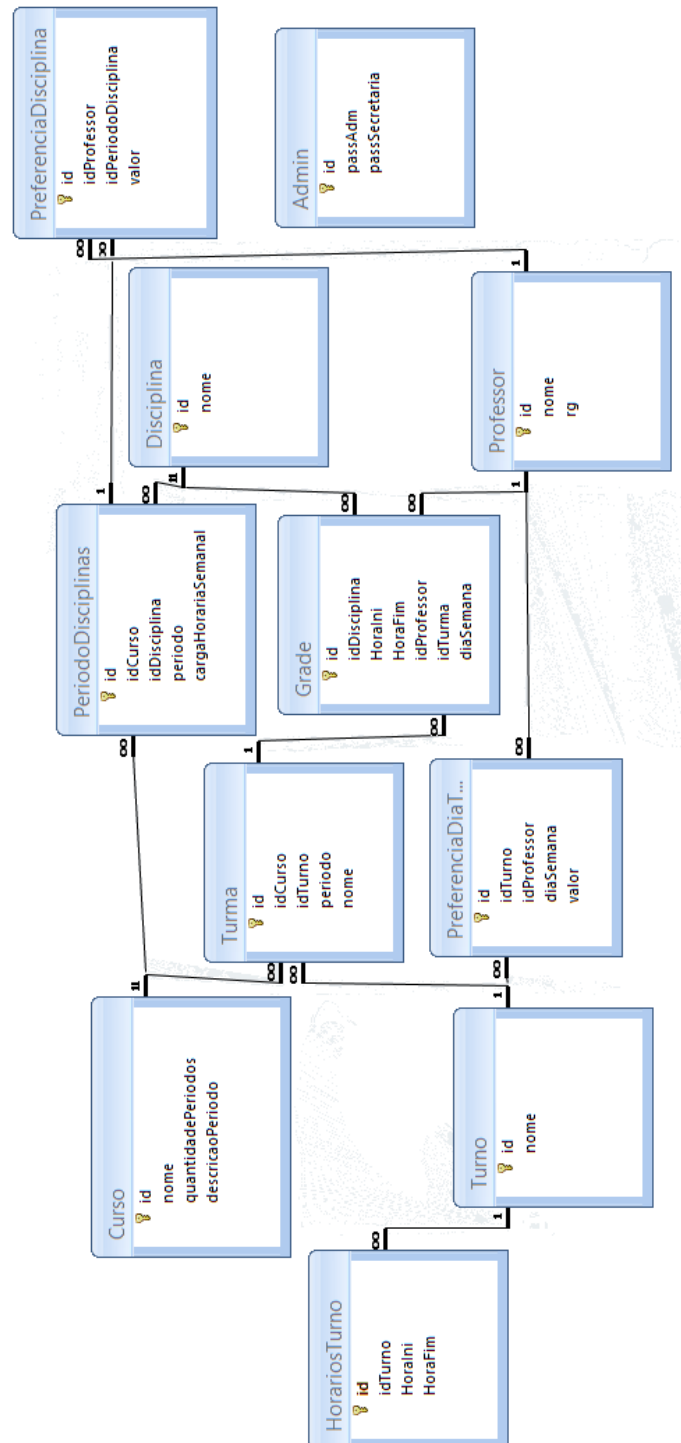


Figura 12 - Modelo de Entidade Relacionamento

(FONTE: do autor)

### 4.3. Dicionário de Dados

Abaixo, pode ser analisado o dicionário de dados do projeto Kairos.

Entidade: Admin					
Descrição: Estrutura que armazena as senhas dos usuários do sistema					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Admin
passAdm	Simples	Sim	Texto	256	
passSecretaria	Simples	Sim	Texto	256	
Chave Primária:	id				
Chave Estrangeira:	-				
Entidade: Curso					
Descrição: Estrutura que armazena as informações sobre o curso					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Curso
nome	Simples	Sim	Texto	50	Nome do curso
quantidadePeriodos	Simples	Sim	Numero	Inteiro longo	Quantidade de Períodos
descricaoPeriodo	Simples	Sim	Texto	25	Descrição do Período
Chave Primária:	id				
Chave Estrangeira:	-				

Entidade: Grade					
Descrição: Estrutura que armazena os dados da grade gerada					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Grade
idDisciplina	Simples	Sim	Numero	Inteiro longo	ID da tabela Disciplina
HoraIni	Simples	Sim	Data/Hora		Hora Inicial
HoraFim	Simples	Sim	Data/Hora		Hora Final
idProfessor	Simples	Sim	Numero		ID da tabela Professor
idTurma	Simples	Sim	Numero		ID da tabela Turma
diaSemana	Simples	Sim	Numero	Inteiro longo	Dia da Semana
Chave Primária:	id				
Chave Estrangeira:	idDisciplina, idProfessor e idTurma				
Entidade: HorariosTurno					
Descrição: Estrutura que armazena os horários dos turnos cadastrados					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela HorariosTurno
idTurno	Simples	Sim	Numero	Inteiro longo	ID da tabela Turno
HoraIni	Simples	Sim	Data/Hora		Hora Inicial
HoraFim	Simples	Sim	Data/Hora		Hora Final
Chave Primária:	Id				
Chave Estrangeira:	idTurno				
Entidade: Disciplina					
Descrição: Estrutura que armazena as informações sobre as disciplinas					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Disciplina
nome	Simples	Sim	Texto	50	Nome da Disciplina
Chave Primária:	id				
Chave Estrangeira:	-				

Entidade: PeríodoDisciplinas					
Descrição: Estrutura que armazena as disciplinas de cada período					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela PeríodoDisciplinas
idCurso	Simples	Sim	Numero	Inteiro longo	ID da tabela Curso
idDisciplina	Simples	Sim	Numero	Inteiro longo	ID da tabela Disciplina
periodo	Simples	Sim	Numero	Inteiro longo	Período
cargaHorariaSemanal	Simples	Sim	Data/Hora		Carga Horária Semanal
Chave Primária:	id				
Chave Estrangeira:	idCurso, idDisciplina				
Entidade: PreferenciaDiaTurno					
Descrição: Estrutura que armazena as preferências de cada professor para dias e turnos.					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela PreferenciaDiaTurno
idTurno	Simples	Sim	Numero	Inteiro longo	ID da tabela Turno
idProfessor	Simples	Sim	Numero	Inteiro longo	ID da tabela Professor
diaSemana	Simples	Sim	Numero	Inteiro longo	Dia da semana
valor	Simples	Sim	Numero	Inteiro longo	Valor
Chave Primária:	id				
Chave Estrangeira:	idTurno, idProfessor				

<b>Entidade: Turma</b>					
Descrição: Estrutura que armazena as informações das turmas					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Turma
idCurso	Simples	Sim	Numero	Inteiro longo	ID da tabela Curso
idTurno	Simples	Sim	Numero	Inteiro longo	ID da tabela Turno
periodo	Simples	Sim	Numero	Inteiro longo	Período
nome	Simples	Sim	Texto	20	Nome da Turma
Chave Primária:	id				
Chave Estrangeira:	idCurso, idTurno				
<b>Entidade: Professor</b>					
Descrição: Estrutura que armazena as informações dos professores					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Professor
nome	Simples	Sim	Texto	50	Nome do professor
rg	Simples	Sim	Texto	15	RG do Professor
Chave Primária:	Id				
Chave Estrangeira:	-				
<b>Entidade: PreferenciaDisciplina</b>					
Descrição: Estrutura que armazena as preferências de cada professor para as disciplinas					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela PreferenciaDisciplina
idProfessor	Simples	Sim	Numero	Inteiro longo	ID da tabela Professor
idPeriodoDisciplina	Simples	Sim	Numero	Inteiro longo	ID da tabela PeriodoDisciplina
valor	Simples	Sim	Numero	Inteiro longo	Valor
Chave Primária:	id				
Chave Estrangeira:	idProfessor, idPeriodoDisciplina				

<b>Entidade: Turno</b>					
Descrição: Estrutura que armazena as informações dos turnos					
Atributo	Classe	Obrigatório	Tipo	Tamanho	Descrição
id	Determinante	Sim	Numero	Inteiro longo	ID da tabela Turno
nome	Simples	Sim	Texto	25	Nome do Turno
Chave Primária:	id				
Chave Estrangeira:	-				

#### **4.4. Hardware e Software**

O uso da linguagem *.NET* permite não especificar qual será o sistema operacional, e a máquina em que o programa irá executar. Tendo como requisito apenas que o *.NET Framework 4.0* esteja corretamente instalado na máquina utilizada.

#### **4.5. Estrutura de dados do sistema**

O sistema utiliza uma estrutura de dados orientada a objetos para representar os cromossomos.

Cada cromossomo possui uma lista com os cursos previamente cadastrados, cada curso possui  $X$  períodos, sendo  $X$  o número de períodos cadastrados individualmente para cada curso.

Os períodos por sua vez possuem uma lista com as disciplinas nele contidas, tal quais as turmas nele existentes.

As turmas possuem uma lista fixa de dias, sendo 7 no total, representando os dias da semana, com cada dia possuindo uma lista de horários  $Y$ , sendo  $Y$  o conjunto de horários cadastrados para o turno especificado para a turma.

Cada horário armazena o professor e a disciplina, por fim representando a grade a ser montada.

Os professores são armazenados em uma lista à parte dessa estrutura, contendo uma lista com as disciplinas que ministram e com os dias e turnos que têm disponíveis, tal qual uma pontuação de 0 a 10 representando sua preferência por aquele item em específico.

#### **4.6. Geração da população inicial no sistema**

A criação da população inicial funciona mesclando essas duas estruturas, primeiro sorteando aleatoriamente uma disciplina, depois sorteando um professor dentre aqueles que pontuaram aquela disciplina com um valor de preferência superior a 0. E por fim sorteando



um horário aleatório que esteja em um dia e turno que o professor tenha pontuado com um valor de preferência também superior a 0 e que já não esteja ocupado.

#### **4.7. Utilização de *crossover* no sistema**

Para o *crossover* é realizado o sorteio de dois cromossomos, X e Y, através de uma roleta viciada onde o individuo com maior aptidão possui mais chances de ser escolhido.

Para cada turma em X e Y é realizado o sorteio de um dia e horário H. O cromossomo Z, gerado à partir de X e Y, mantém as propriedades de X, exceto quanto ao horário H, que passa a ser de Y. Quanto ao horário H de X em Z, é realizada uma troca equivalente, alterando-o para o horário onde antes ficava a disciplina ocupada em H no X. Esse processo é realizado duas vezes, para cada turma nos cromossomos X e Y.

Por fim, é realizado um sorteio através de uma roleta viciada invertida na população, com o cromossomo que possui o menor valor de aptidão tendo mais chances de ser escolhido. O cromossomo sorteado é excluído da população, e Z é acrescentado em seu lugar.

A mutação é realizada de forma análoga ao *crossover*, conquanto que  $Y = X$ .

#### **4.8. Diagrama de caso de uso**

GUEDES, GILLEANES T.A, explica que o diagrama de caso de uso é o diagrama mais geral e informal da UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada), utilizada nas fases de levantamento e análise de requisito do sistema e será consultado durante todo o processo de modelagem e ainda pode servir de base para outros diagramas.

No sistema Kairos o diagrama de caso de uso é composto por dois atores e 10 elipses com suas respectivas funções.

Os atores Administrador e Secretária têm suas funções definidas dentro do sistema de acordo com seus privilégios. O Administrador está ligado a Secretária por generalização, pois

herda dela todas as suas funções. A Secretária por sua vez não tem os privilégios do Administrador.

A figura 13 apresenta o diagrama do sistema Kairos onde:

- A Secretária deverá “Efetuar *Login*” para ter acesso ao sistema. Terá como funções “Cadastrar Turno”, “Cadastrar Curso”, “Cadastrar Turma”, “Cadastrar Disciplina”, “Cadastrar Professor”, “Gerar Grade”, e “Visualizar Grade”.

A função de “Cadastrar Turma” só irá ocorrer se um curso já estiver cadastrado e o mesmo ocorre para a função “Cadastrar Disciplina”.

- O Administrador terá todos os privilégios da secretária e poderá também “Alterar Cadastro de Professor”, sendo que para essa situação ocorrer, haverá a necessidade de um cadastro existente de professor. E ainda, o Administrador poderá “Alterar Cadastro de Curso” e para esta situação ocorrer, também será necessária à existência de um curso previamente cadastrado.

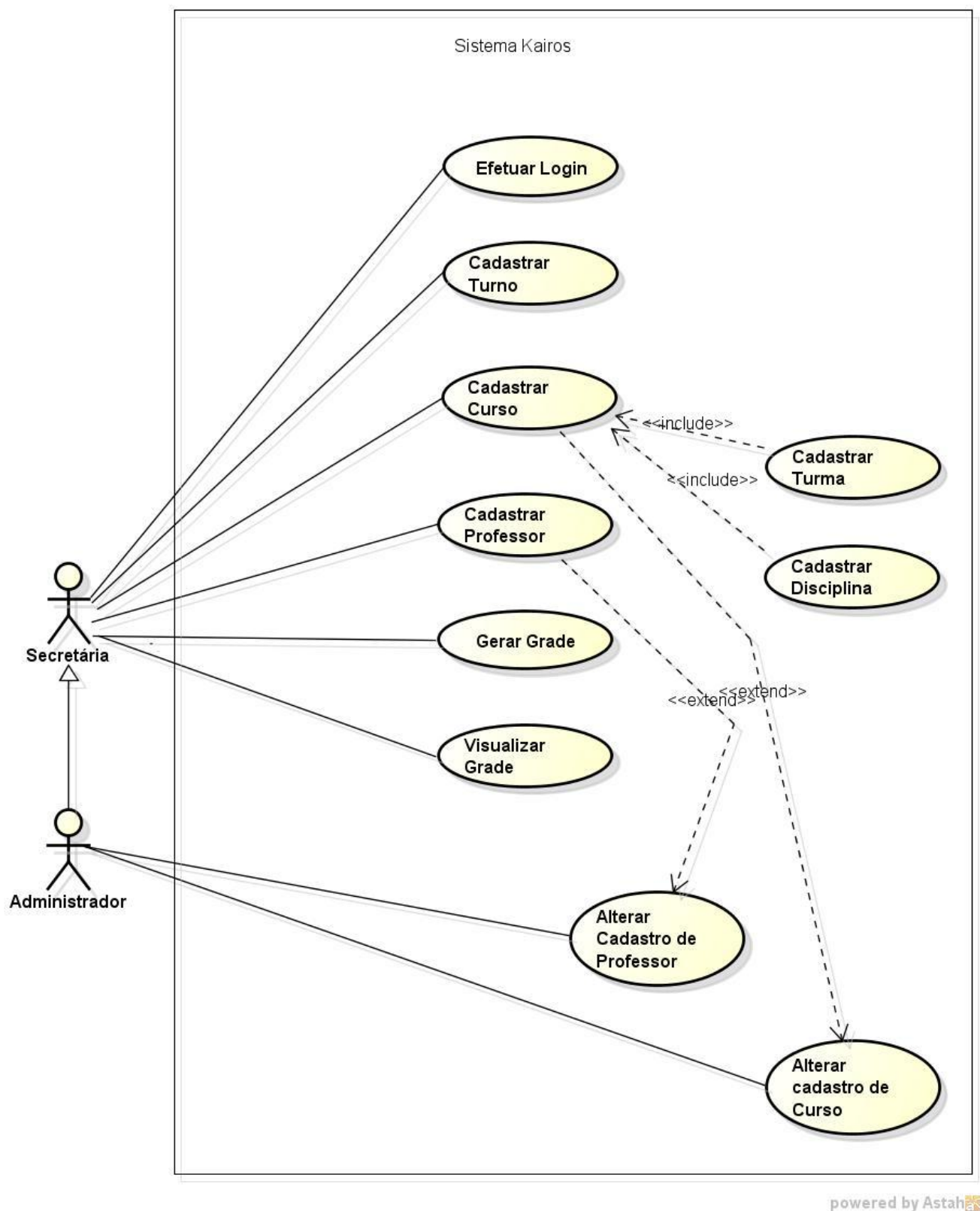


Figura 13 - Diagrama de Caso de Uso

(FONTE: Do autor)

#### **4.8.1. Descrições dos Casos de Uso**

##### **4.8.1.1.CDU 01 - Efetuar Login**

Este caso de uso inicia-se quando o usuário deseja acessar o sistema.

Atores: Secretária e Administrador.

Pré-condição: O usuário deverá estar cadastrado no sistema.

Fluxo Principal:

1. O usuário acessa a ferramenta instalada no computador.
2. O usuário preenche os campos de Usuário e Senha.
3. O usuário clica no botão Login.
4. O sistema exibe sua tela inicial. **(E-1)**
5. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

E-1) O usuário recebe uma mensagem de dados inválidos.

##### **4.8.1.2.CDU 02 – Cadastrar Turno**

Este caso de uso inicia-se quando o usuário deseja cadastrar um Turno.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema.

Fluxo Principal:

1. O usuário solicita cadastrar um Turno.
2. O sistema exibe os campos a serem preenchidos para efetuar o cadastro.
3. O usuário preenche os dados referentes ao cadastro com horário de início e término e clica no botão com sinal de adição. **(E-1)**
4. O sistema salva as informações automaticamente.

5. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

E-1) O usuário desiste de cadastrar o Turno.

E-2) O usuário seleciona o Turno e clica no botão com sinal de subtração.

E-3) O cadastramento é cancelado e o caso de uso termina.

#### **4.8.1.3.CDU 03 – Cadastrar Curso**

Este caso de uso inicia-se quando o usuário deseja cadastrar um Curso.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema e ao menos um Turno cadastrado.

Fluxo Principal:

1. O usuário solicita cadastrar um Curso.
2. O sistema exibe os campos a serem preenchidos para efetuar o cadastro.
3. O usuário preenche os dados referentes ao cadastro. **(E-1)**
4. O sistema salva as informações automaticamente. **(E-2)**
5. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

E-1) O usuário desiste de cadastrar o Curso.

E-2) O usuário seleciona o Curso e clica no botão com sinal de subtração.

E-3) O cadastramento é cancelado.

#### **4.8.1.4.CDU 04 – Cadastrar Turma**

Este caso de uso inicia-se quando o usuário deseja cadastrar uma Turma.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema, um Turno cadastrado e um Curso.

Fluxo Principal:

1. O usuário solicita cadastrar uma Turma.
2. O sistema exibe os campos a serem preenchidos para efetuar o cadastro.
3. O usuário preenche os dados referentes ao cadastro seleciona o turno e clica no botão com sinal de adição. **(E-1)**
4. O sistema salva e exibe a informação inserida. **(E-2)**
5. Fim do caso de uso. **(E-3)**

Fluxo Alternativo: Não se aplica.

Fluxo de exceção:

- E-1) O usuário desiste de cadastrar a Turma.
- E-2) O usuário seleciona a Turma e clica no botão com sinal de subtração.
- E-3) O cadastramento é cancelado e o caso de uso termina.

#### **4.8.1.5.CDU 05 – Cadastrar Disciplina**

Este caso de uso inicia-se quando usuário deseja cadastrar uma Disciplina.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema, um Turno cadastrado e um Curso.

Fluxo Principal:

1. O usuário solicita cadastrar uma Disciplina.
2. O sistema exibe os campos a serem preenchidos para efetuar o cadastro.
3. O usuário preenche os dados referentes ao cadastro e clica no botão com sinal de adição. **(E-1)**
4. O sistema salva as informações e exibe o item cadastrado. **(E-2)**
5. Fim do caso de uso. **(E-3)**

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

- E-1) O usuário desiste de cadastrar a Disciplina.
- E-2) O usuário seleciona o item e clica no botão com sinal de subtração.
- E-3) O cadastramento é cancelado.

#### **4.8.1.6.CDU 06 – Cadastrar Professor**

Este caso de uso inicia-se quando o usuário devidamente validado no sistema deseja inserir um Professor

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema e a existência de Turno, Curso, Turma e Disciplina cadastrados.

Fluxo Principal:

1. O usuário solicita cadastrar um Professor.
2. O sistema exibe os campos a serem preenchidos para efetuar o cadastro.
3. O usuário preenche os dados referentes ao cadastro. **(E-1)**
4. O sistema salva as informações automaticamente. **(E-2)**
5. Fim do caso de uso. **(E-3)**

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

E-1) O usuário desiste de cadastrar o Professor.

E-2) O usuário seleciona o Professor clica no botão com sinal de subtração.

E-3) O cadastramento é cancelado.

#### **4.8.1.7.CDU 07 – Alterar Cadastro de Professor**

Este caso de uso inicia-se quando usuário deseja alterar os dados de um Professor que já se encontra cadastrado no sistema.

Atores: Administrador.

Pré-condição: Usuário validado no sistema e um cadastro de professor existente.

Fluxo Principal:

1. O usuário solicita uma busca através do nome.
2. O sistema apresenta o nome solicitado
3. O usuário seleciona o nome e clica no botão com sinal de subtração.
4. O usuário insere os novos dados.

5. O sistema salva a alteração.
6. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção: Não se aplica.

#### **4.8.1.8.CDU 08 – Alterar Cadastro de Curso**

Este caso de uso inicia-se quando usuário deseja alterar os dados de um Curso que já se encontra cadastrado.

Atores: Administrador.

Pré-condição: Usuário cadastrado no sistema e um cadastro de Curso existente

Fluxo Principal:

1. O usuário solicita a busca através do nome do Curso.
2. O usuário informa as alterações do Curso.
3. O sistema salva a informação automaticamente.
4. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de exceção: Não se aplica.

#### **4.8.1.9.CDU 09 - Gerar Grade**

Este caso de uso inicia-se quando o usuário deseja gerar uma grade de horário.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema e a existência de ao menos um Turno, um Curso, uma Turma, uma Disciplina e um professor.

Fluxo Principal:

1. O usuário seleciona todos os itens necessários para gerar a grade. **(E-1)**.
2. O usuário clica no botão “Gerar Grade”. **(E-2)**



3. O sistema apresenta a tela com a grade gerada.
4. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de Exceção:

- E-1) O usuário não inseriu todos os dados necessários.
- E-2) O sistema não gera a grade.

#### **4.8.1.10.CDU 10 – Visualizar Grade**

Este caso de uso inicia-se quando usuário deseja visualizar uma grade já elaborada.

Atores: Administrador e Secretária.

Pré-condição: Usuário validado no sistema e ao menos uma grade gerada.

Fluxo Principal:

1. O usuário clica em “Visualizar Grade”. **(E-1)**
2. O sistema apresenta a(s) grade(s) existente(s). **(E-2)**
3. Fim do caso de uso.

Fluxo Alternativo: Não se aplica.

Fluxo de exceção:

- E-1) Não existe nenhuma grade gerada.
- E-2) O sistema não exibe a grade.

#### 4.9. Diagrama de Classe do sistema

Abaixo veremos os diagrama de classe do sistema Kairos, conforme Figura 14 e Figura 15.

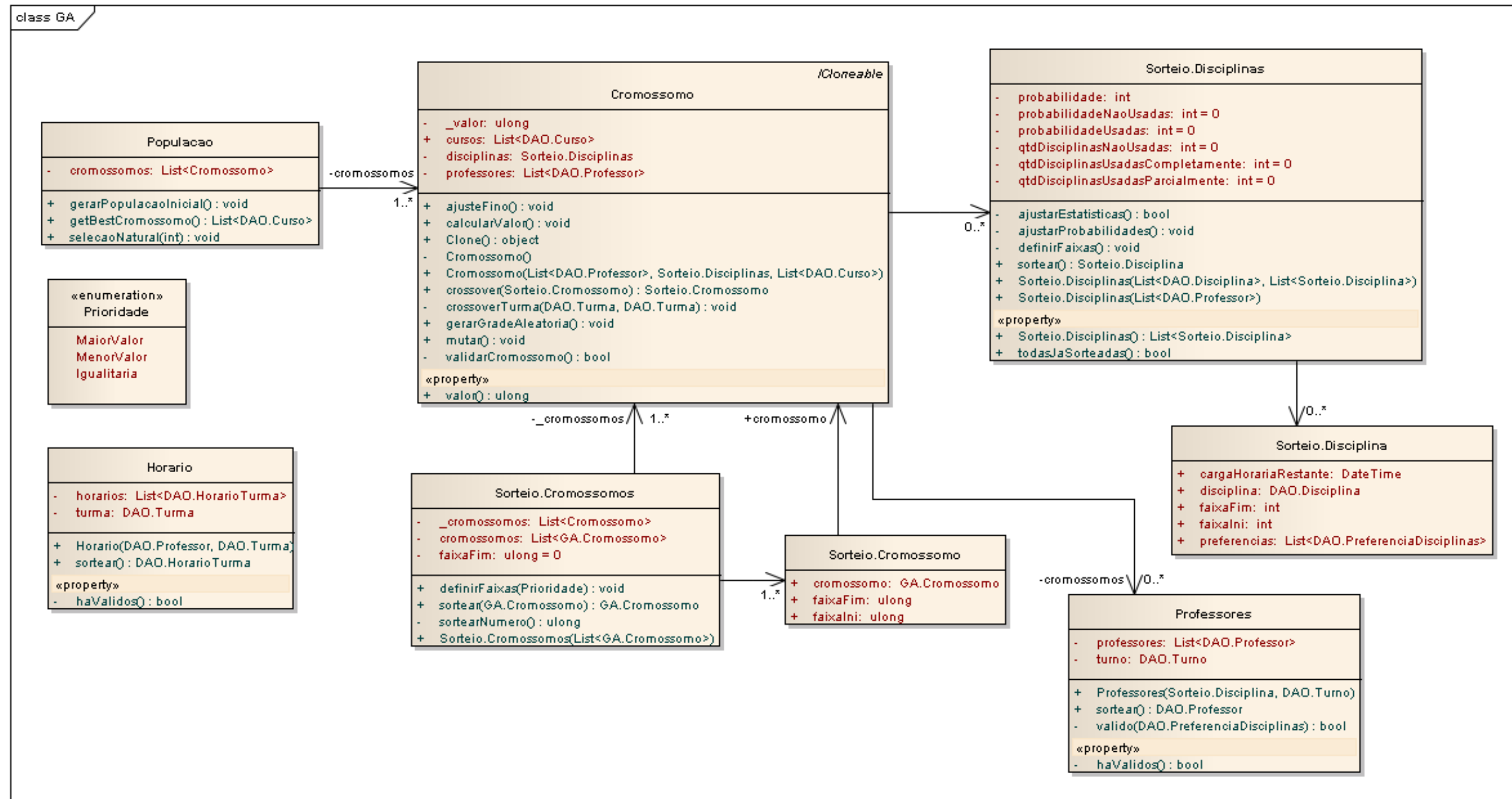


Figura 14 - Diagrama GA

(Fonte: do autor)

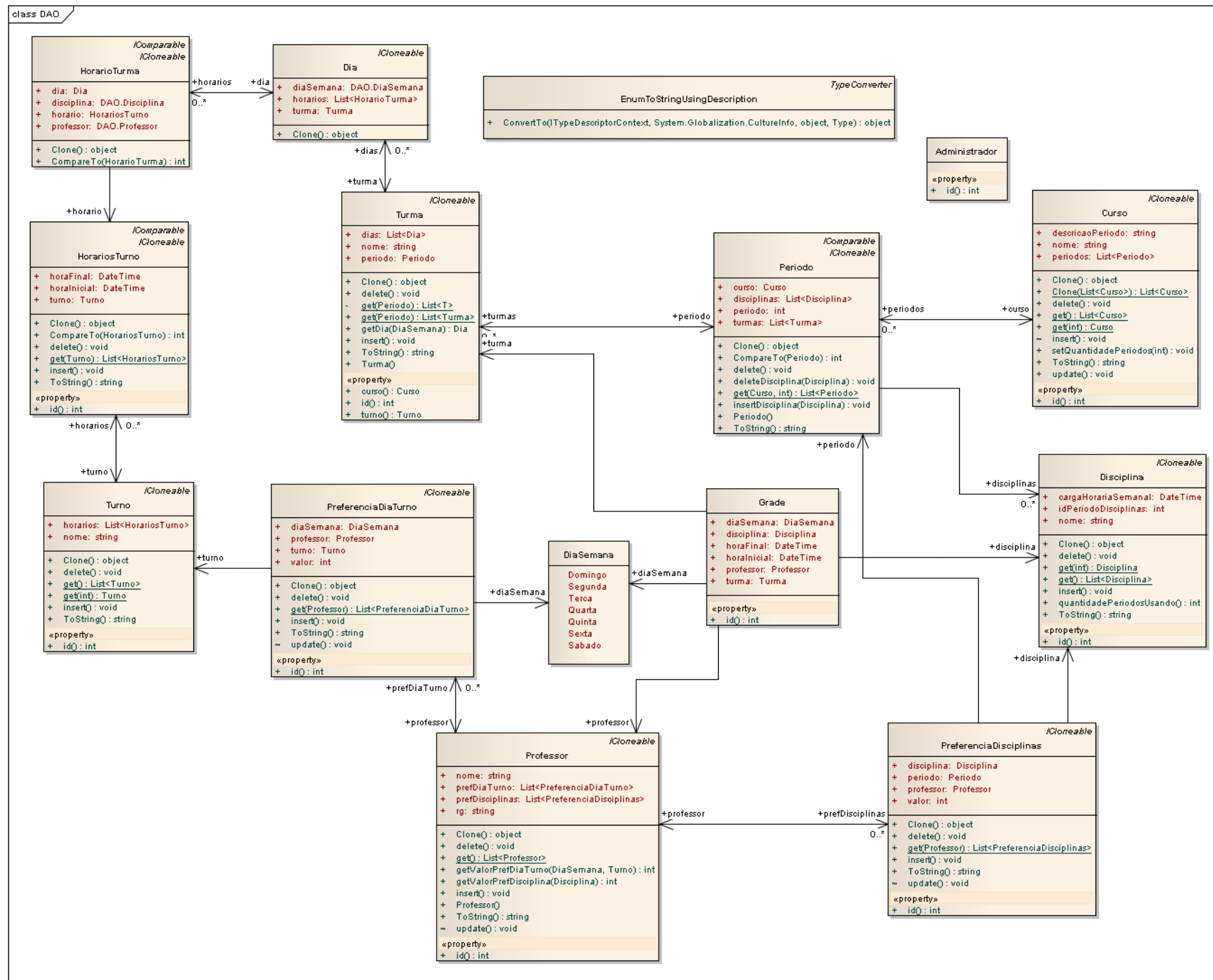


Figura 15 - Diagrama DAO

(Fonte: do autor)

#### 4.10. Diagrama de Sequência do sistema

A seguir, são apresentados os diagramas de sequência elaborados, conforme as figuras: 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27 e 28.

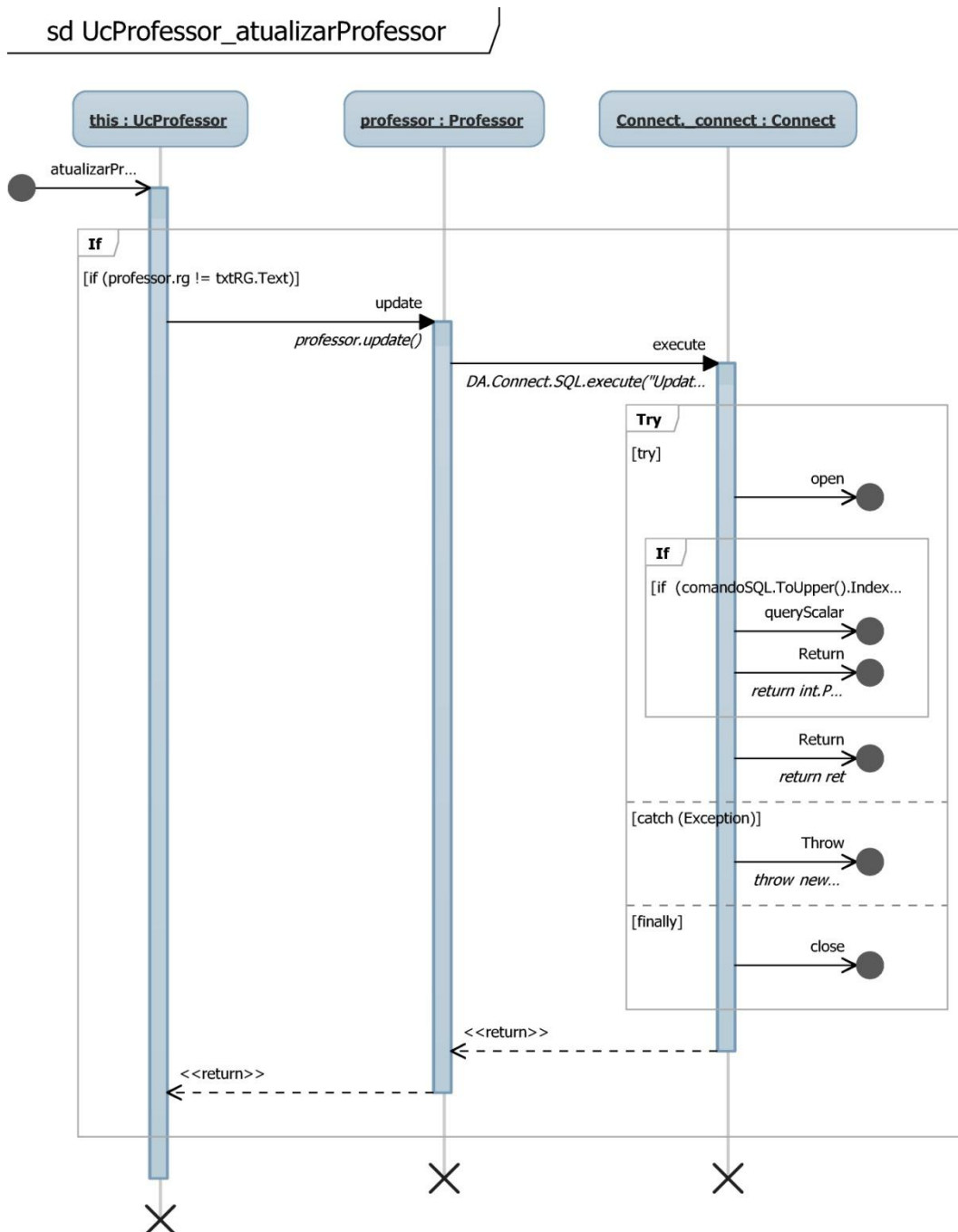


Figura 16 - Diagrama de sequência atualizarProfessor  
(FONTE: do autor)



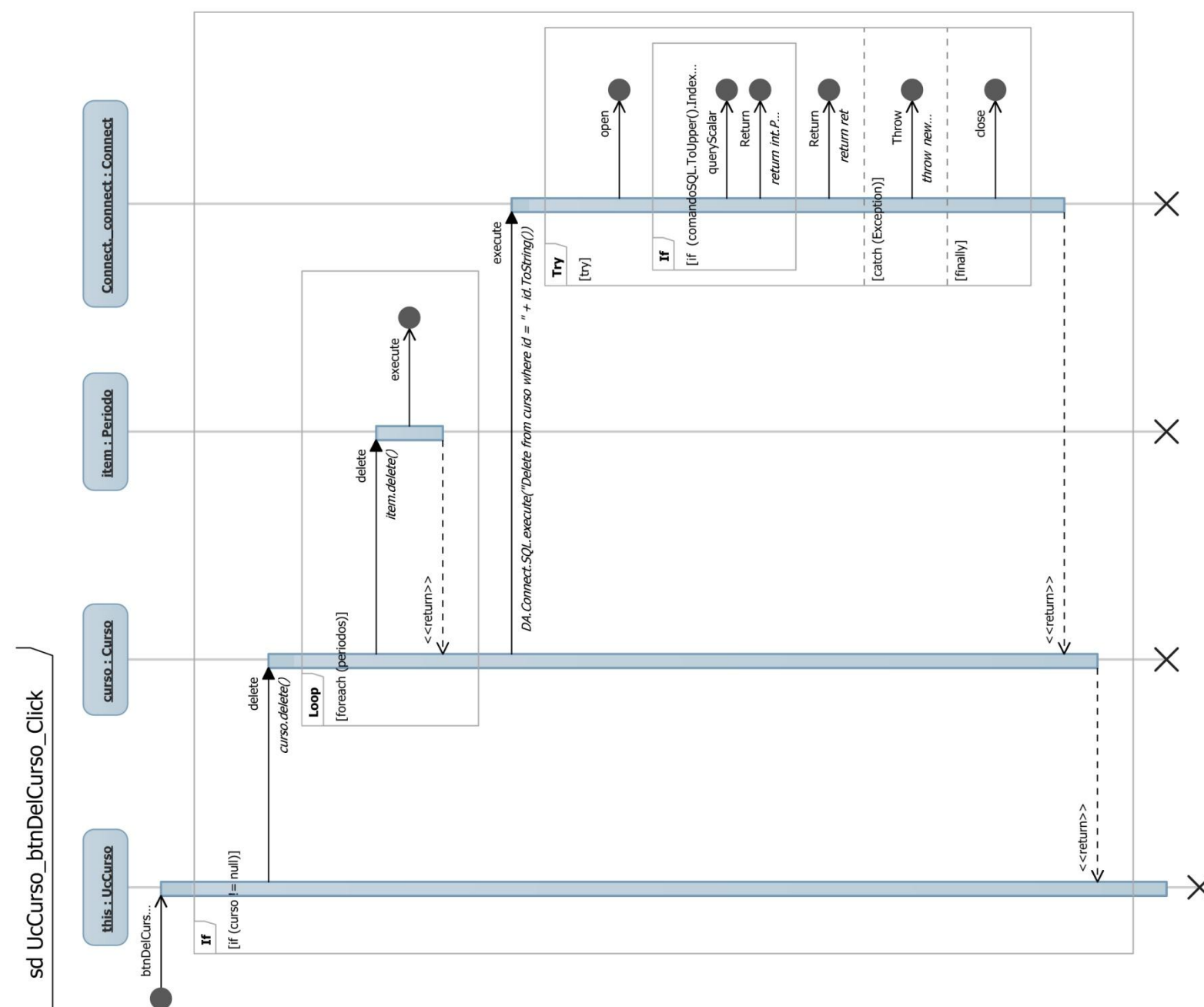


Figura 18- Diagrama de sequência DelCurso

(FONTE: do autor)

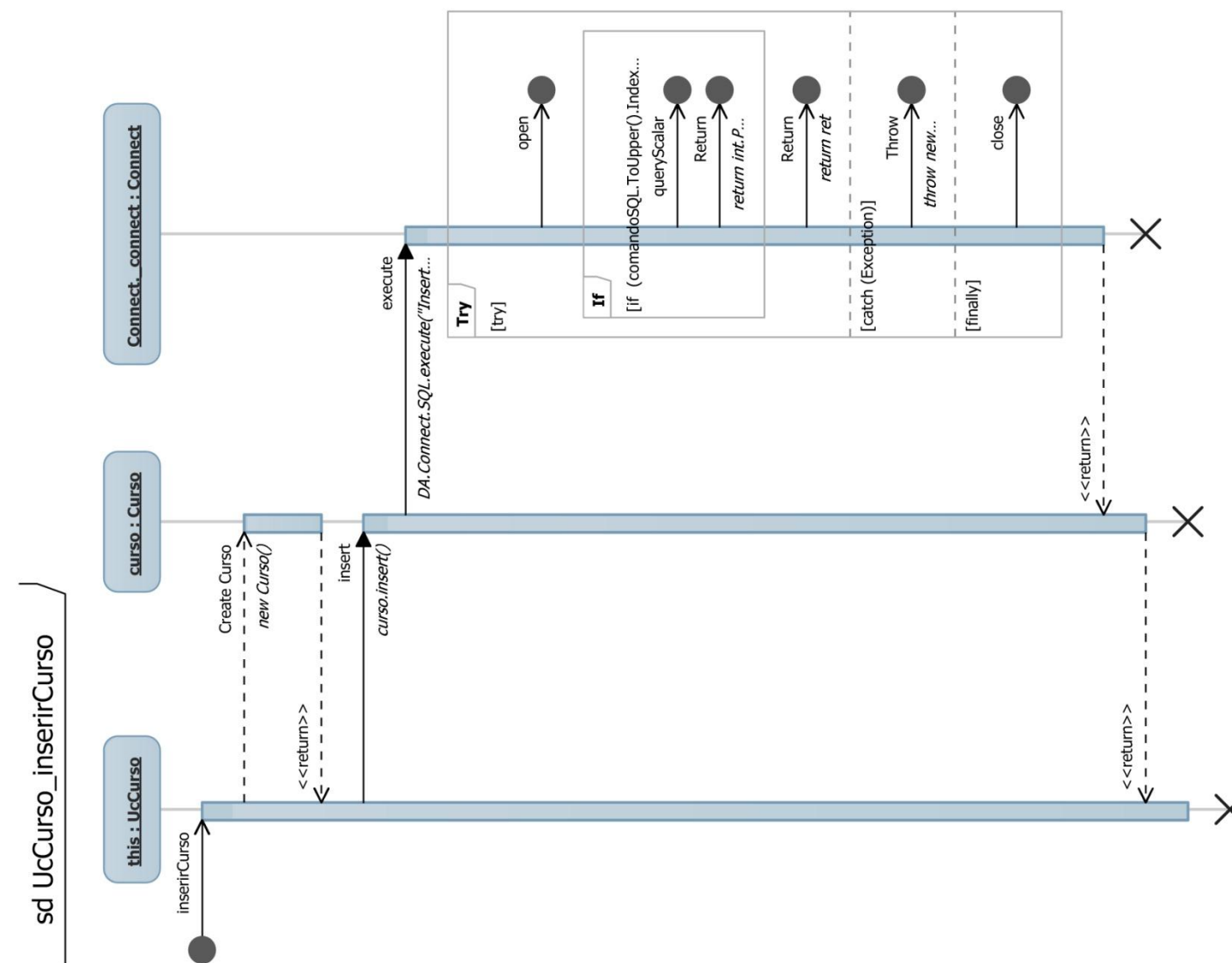


Figura 19 - Diagrama de sequência InserirCurso

(FONTE: do autor)

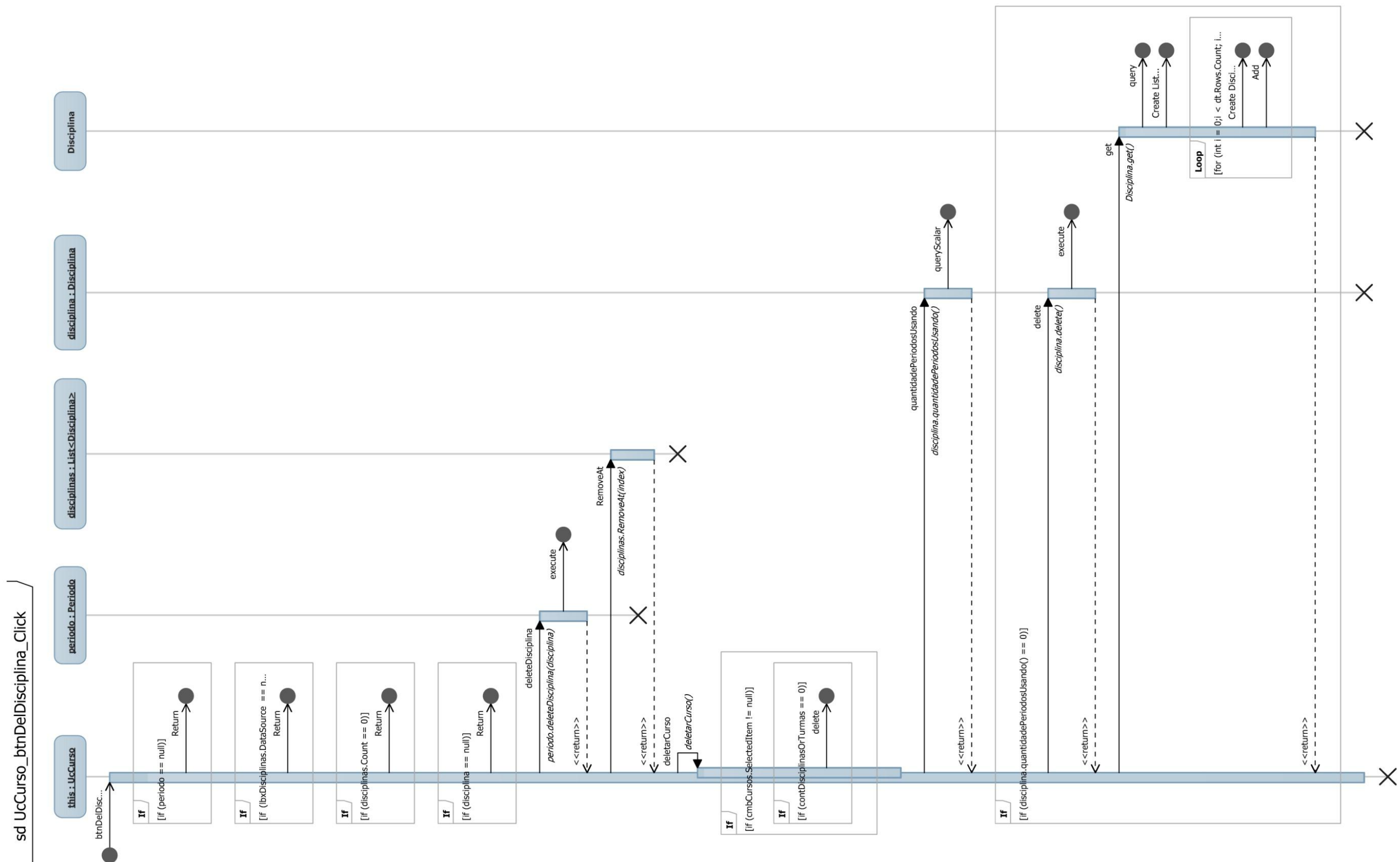


Figura 20 - Diagrama de sequência DelDisciplina

(FONTE: do autor)



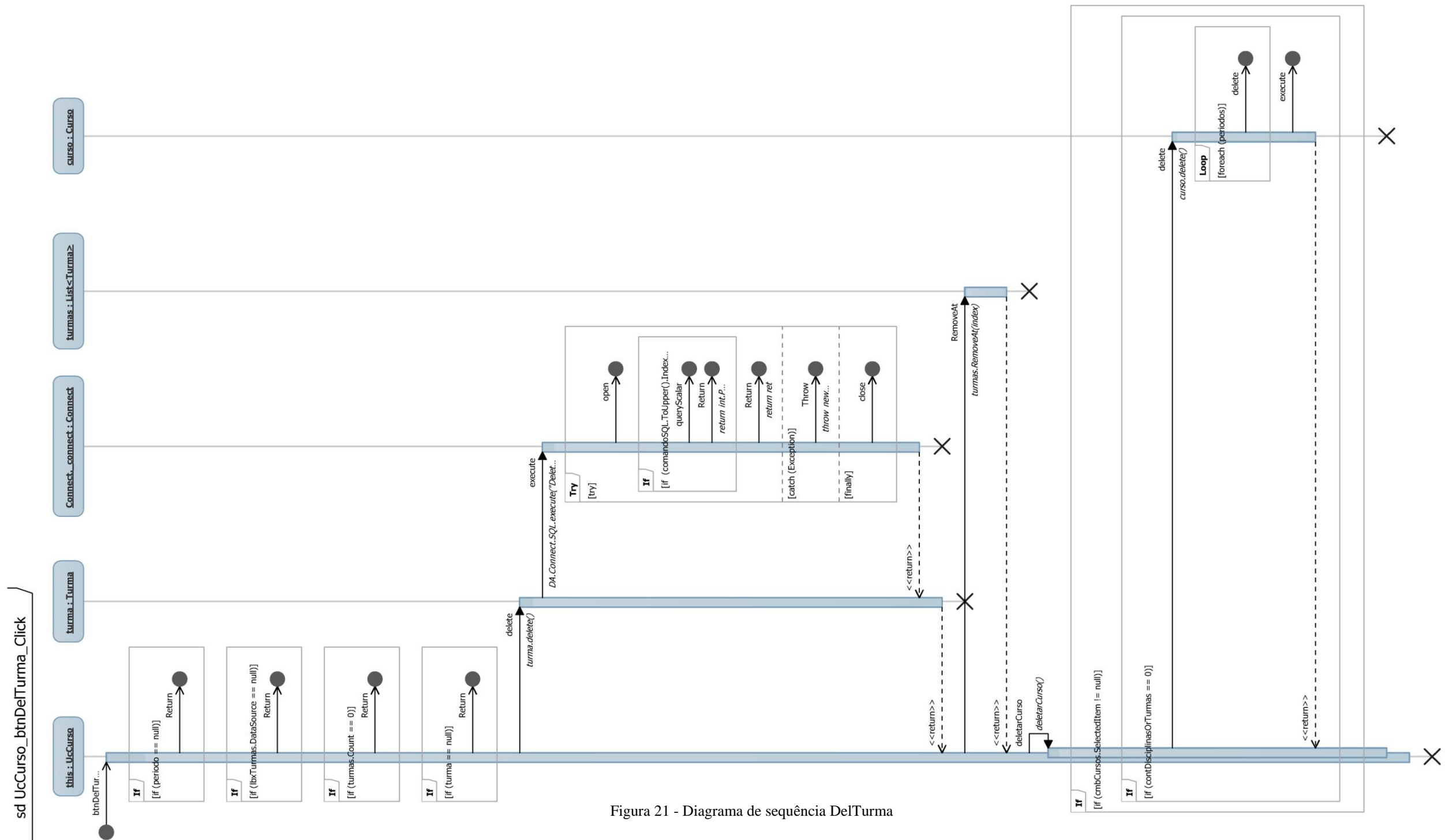


Figura 21 - Diagrama de sequência DelTurma

(FONTE: do autor)

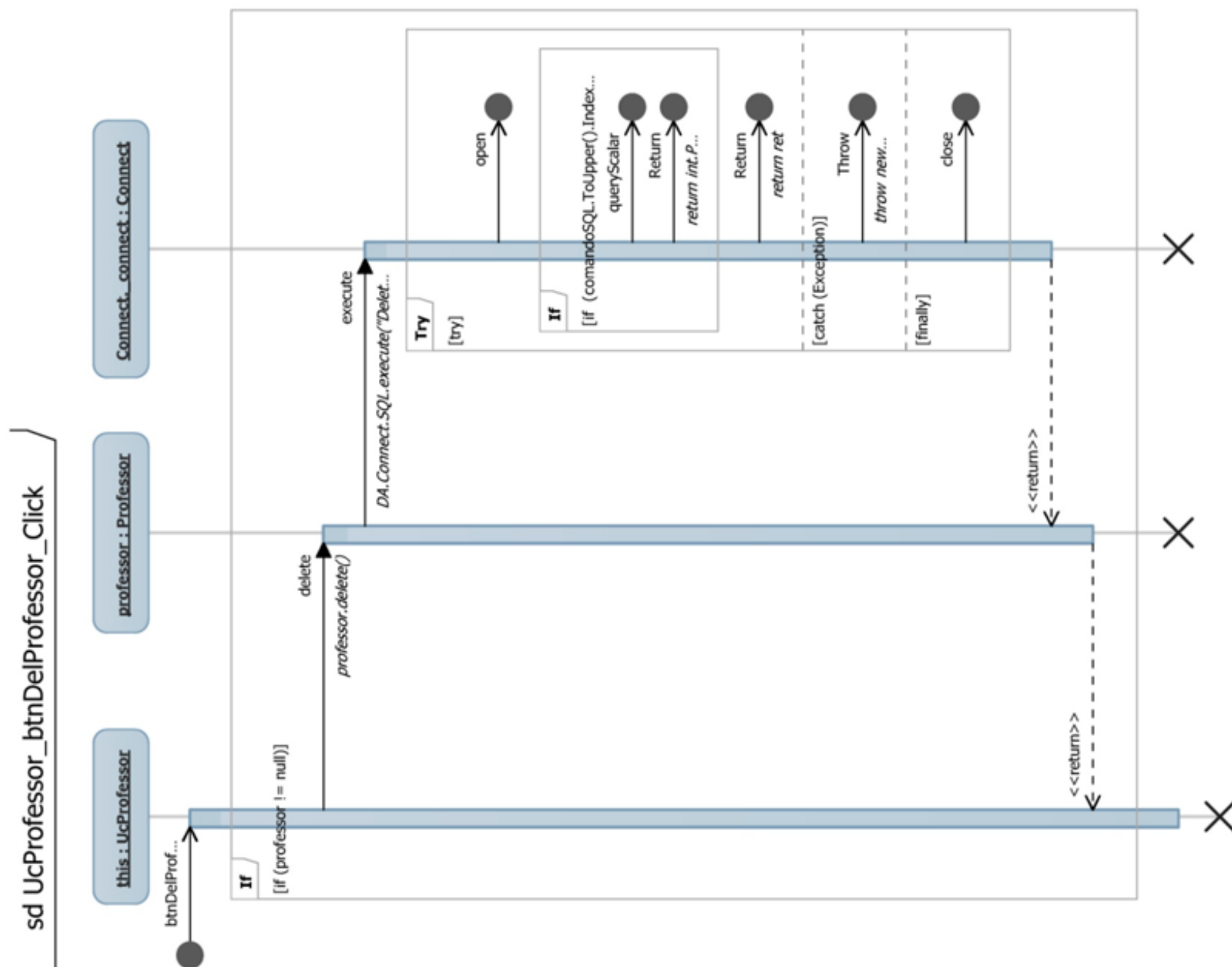


Figura 22 - Diagrama de sequência DelProfessor

(FONTE: do autor)

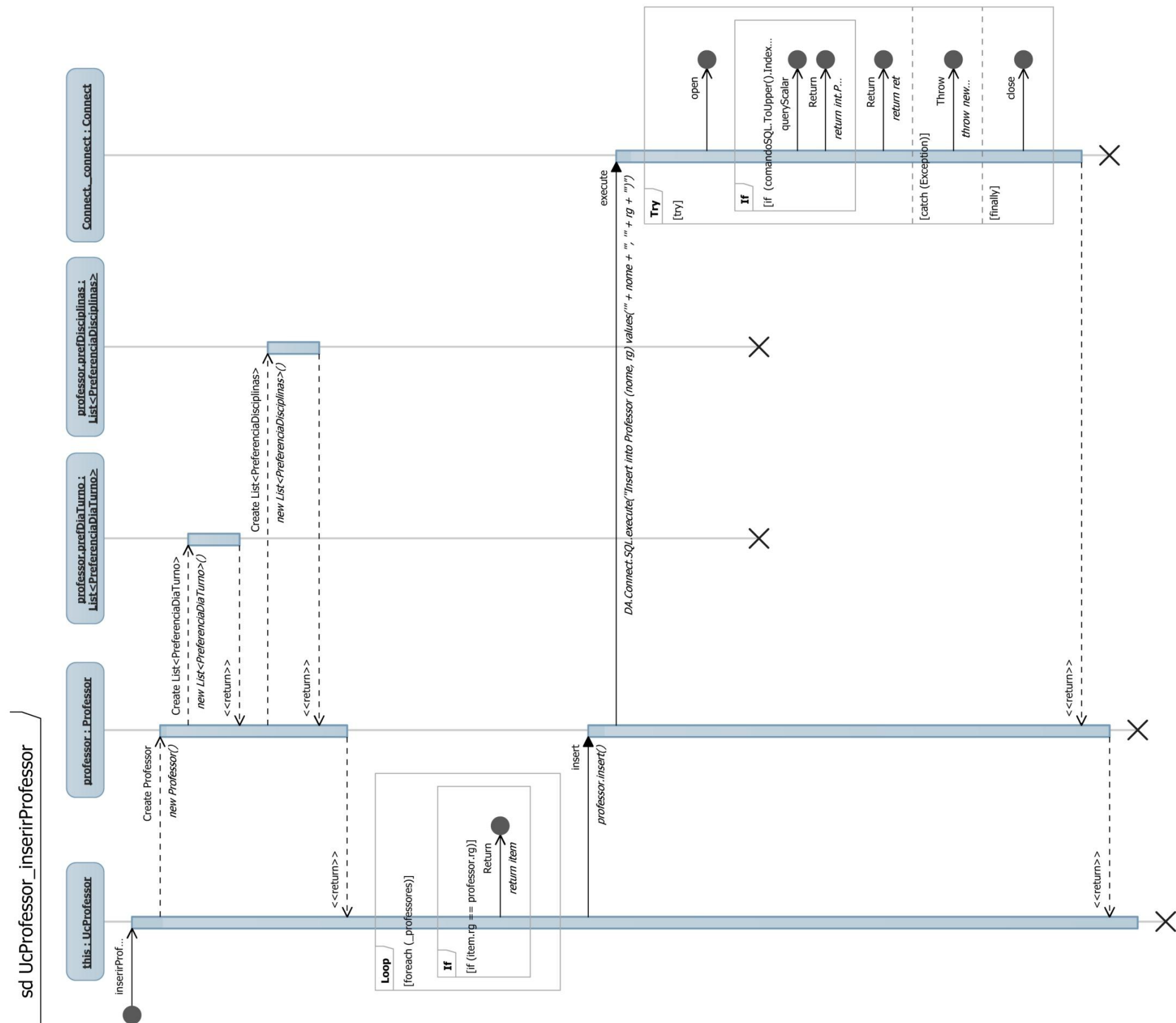


Figura 23 -- Diagrama de sequência InserirProfessor

(FONTE: do autor)

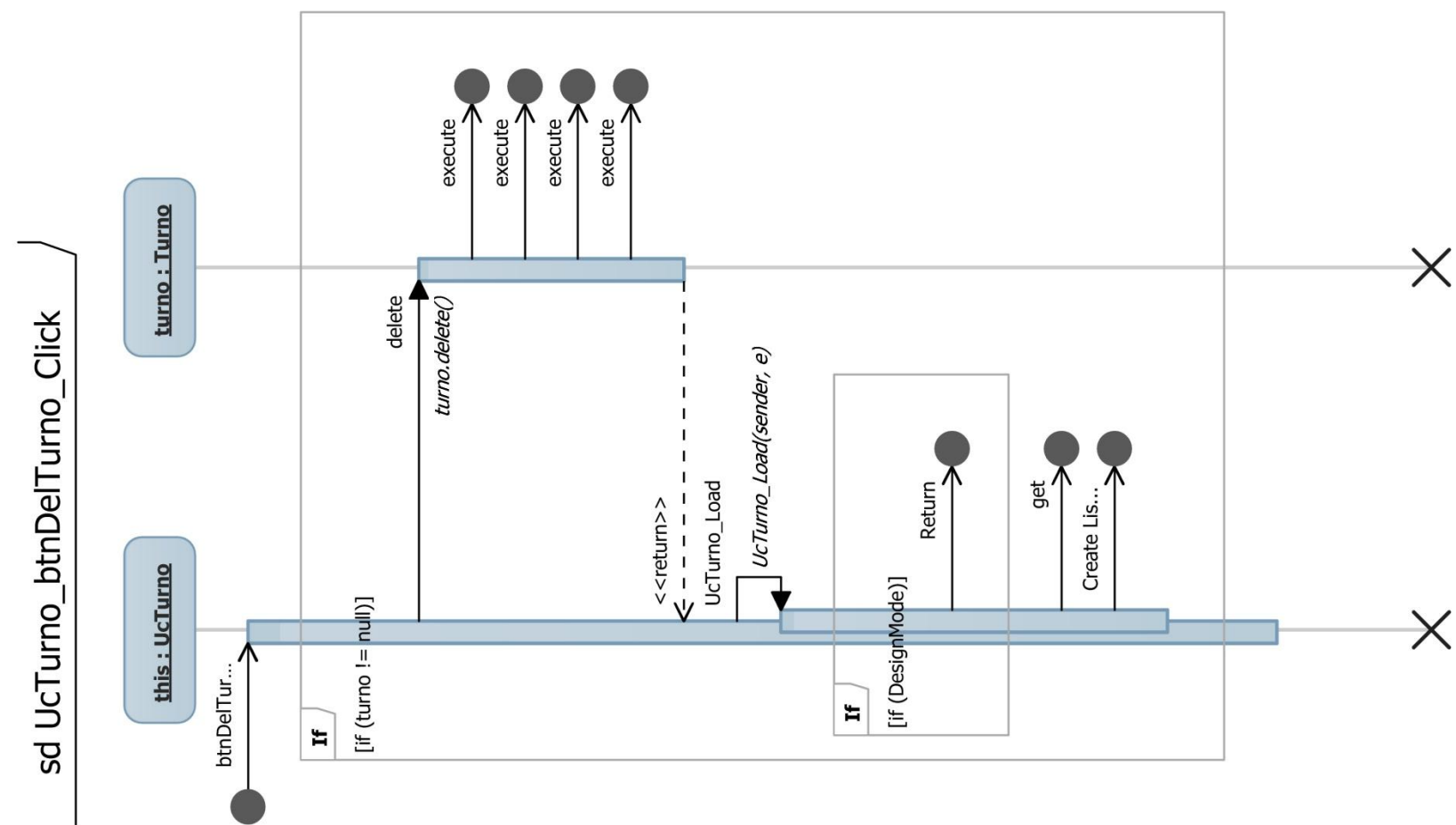


Figura 24 - Diagrama de sequência DelTurno

(FONTE: do autor)

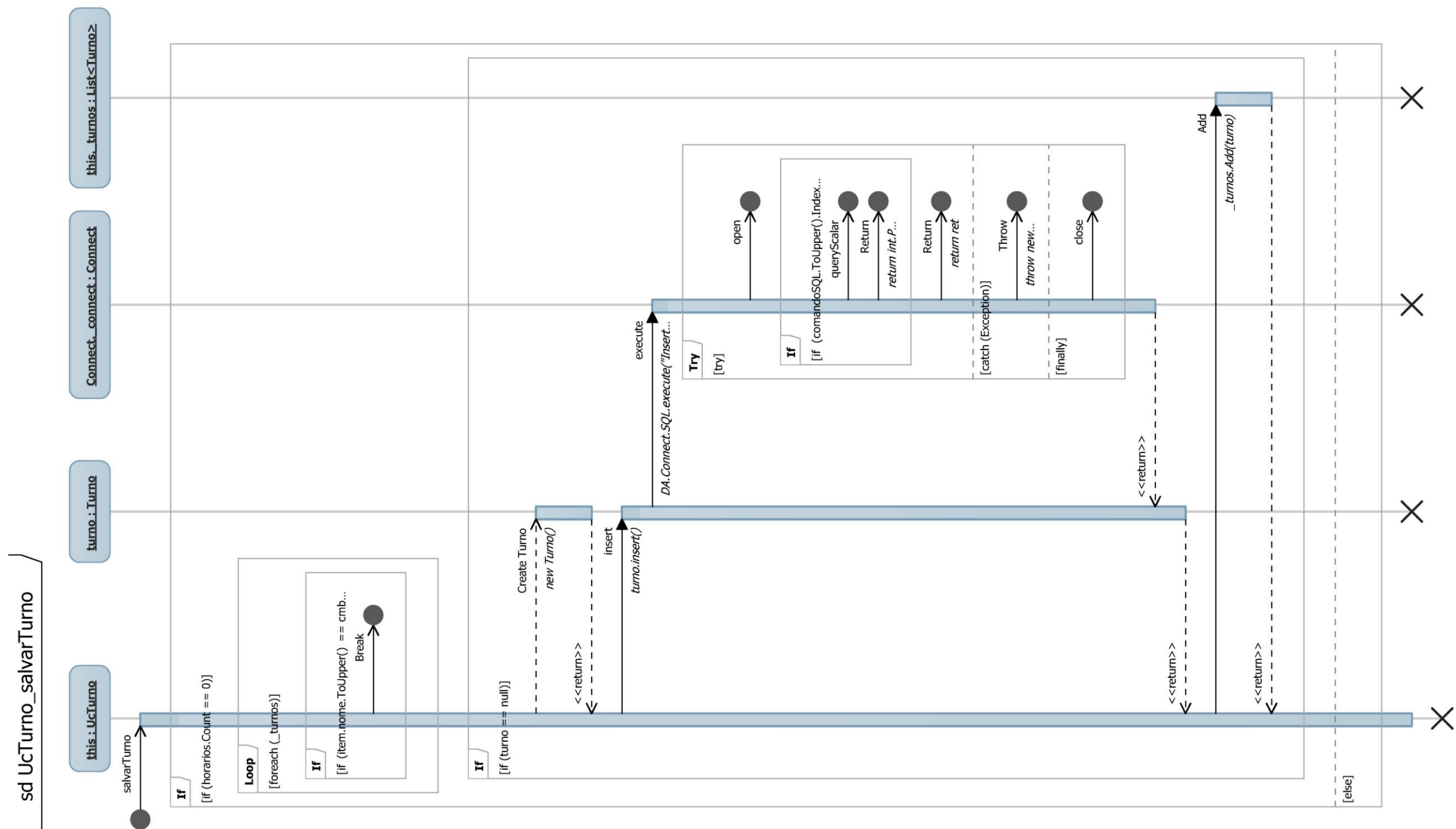


Figura 25 - Diagrama de sequência SalvaTurno

(FONTE: do autor)

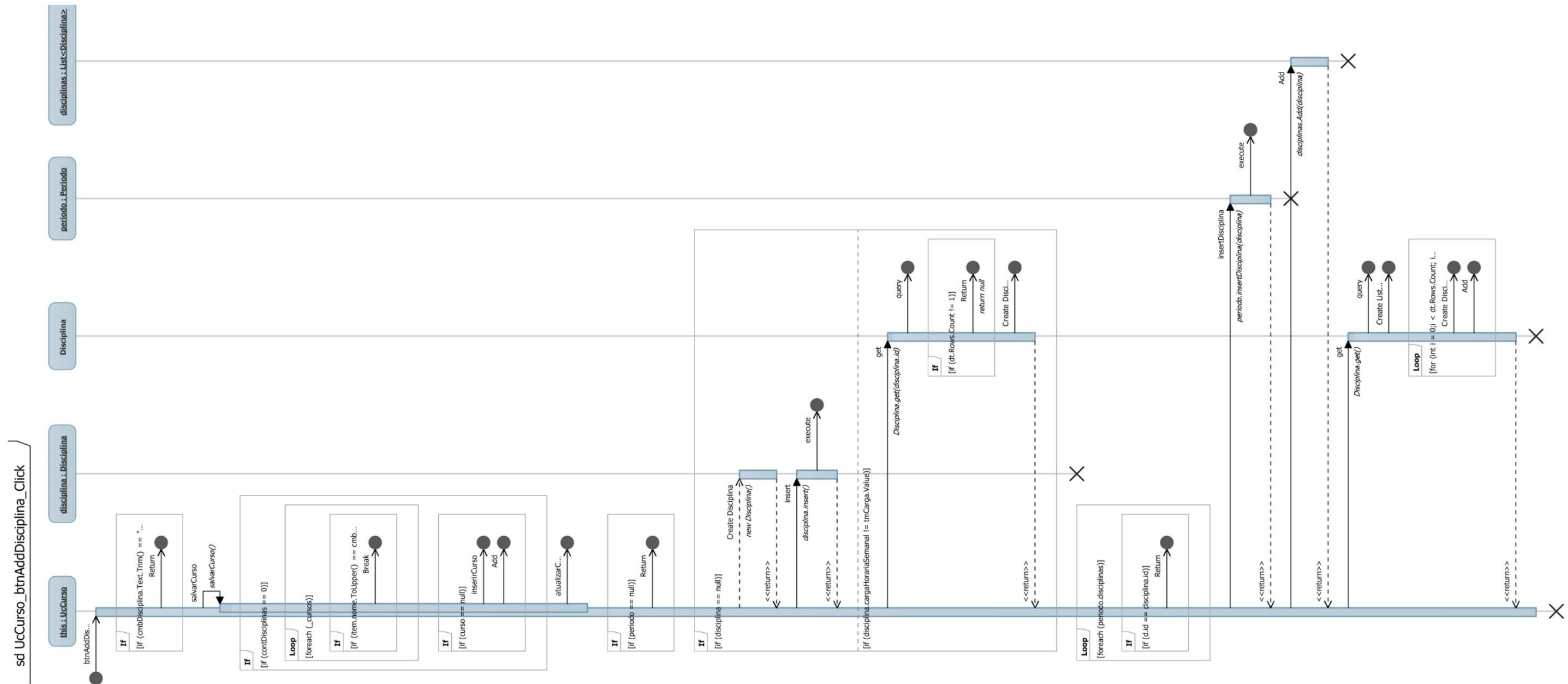


Figura 26 - Diagrama de sequência AddDisciplina

(FONTE: do autor)







#### 4.11. Diagrama de Navegabilidade

Na figura 29 é apresentado o diagrama de navegabilidade do sistema Kairos

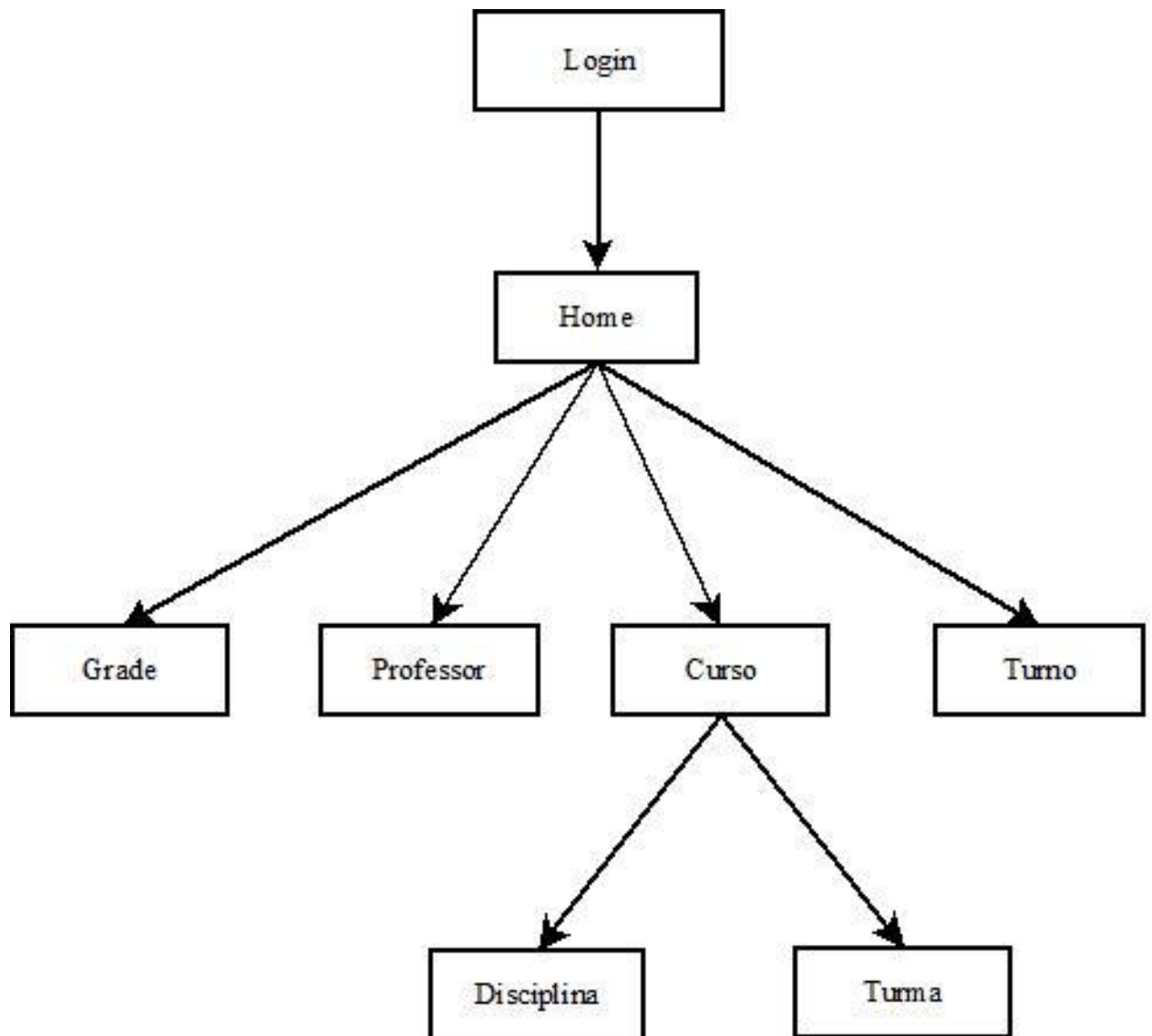


Figura 29 - Diagrama de navegabilidade

(FONTE: do autor)

## 4.12. Softwares Utilizados

### 4.12.1. SQLite

O SQLite é uma biblioteca de *software* sem servidor, sem configurações, sendo um banco de dados SQL relacional. Segundo o próprio desenvolvedor, SQLite é o maior motor de banco de dados SQL do mundo. Além disso, SQLite tem seu código aberto, sendo assim considerado *um open source*.

SQLite nada mais é que um mecanismo de banco de dados SQL embutido a um programa, sendo mais leve, mais rápido e muitas vezes mais pratico. E diferente dos demais bancos de dados, o SQLite não tem seus processos em um Servidor paralelo ou seja, escreve e lê em arquivos de disco comuns.

Também é multi plataforma, permitindo que os arquivos sejam transferidos para quaisquer sistemas operacionais de diferentes arquiteturas computacionais.

As atualizações do SQLite sempre são muito confiáveis. Segundo o fabricante desta biblioteca, são realizados os mais diversos tipos de testes possíveis utilizando milhões de instruções SQL e diversas ferramentas que simulam falhas, erros, queda de energia entre outras falhas para deixar a ferramenta cada vez melhor sem perder sua segurança e credibilidade. Ainda segundo o fabricante, é obvio que ainda há a possibilidade de algum erro atrapalhar o funcionamento da ferramenta, mas que o SQLite, inclusive por ser de código aberto, é honesto e não esconde seus *bugs*, informando ao usuário/desenvolvedor seus bug e até apontando os mais críticos, ao contrário dos concorrentes comerciais que preferem omitir estes fatos.

A base de códigos do SQLite é desenvolvida, por uma equipe internacional de desenvolvedores que se dedicam integralmente ao projeto fazendo com que a ferramenta seja sempre compatível e segura.

O código-fonte do SQLite é gratuito, porém é possível dar um apoio profissional a causa. Há também empresas que patrocinam como a *Nokia*, a *Oracle*, a *Mozilla*, a *Adobe*, a *Bentley* e a *Bloomberg*. Atualmente o SQLite encontra-se na versão 3.8.1.

#### 4.12.2. C#

Pronuncia-se C – *Sharp*, é uma linguagem de programação utilizada para os mais diversos fins que executam sobre o *.NET Framework*. C# também é uma linguagem orientada a objetos sendo considerada pela maioria dos programadores uma das mais simples de ser aprendida, poderosa e com uma tipagem segura, além de rapidez.

Sua fabricante, a *Microsoft* afirma que o C# é a principal linguagem de programação para *.NET*, tendo um parentesco com C++, é uma linguagem simples de implementar.

#### 4.12.3. .NET-Framework 4.0

O *.NET Framework* é um ambiente de execução gerenciado que oferece uma variedade de serviços para seus aplicativos em execução.

Ele consiste de dois componentes principais: o *Common Language Runtime* (CLR), que é o mecanismo de execução que lida com aplicativos que estão executando, e o *.NET Framework*, que fornece uma biblioteca de código testado e reutilizável que os desenvolvedores podem chamar a partir de suas próprias aplicações.

O *.NET Framework* inclui uma ampla gama de serviços para execução de aplicações, tais como:

- Gerenciamento de memória: Em muitas linguagens de programação, os programadores são responsáveis por alocar e liberar memória e para lidar com a duração dos objetos. No *.NET Framework*, o *CLR* fornece esses serviços a favor da aplicação. Em linguagens de programação tradicionais, tipos básicos são definidos pelo compilador, o que dificulta a interoperabilidade entre linguagens. No *.NET Framework*, tipos básicos são definidos pelo tipo de sistema e são comuns a todas as línguas que têm como alvo o *.NET Framework*.
- Uma extensa biblioteca de classes: Em vez de ter que escrever grandes quantidades de código para lidar com operações de programação de baixo nível comum, os programadores podem usar uma biblioteca de fácil acesso de tipos e seus membros diretamente da biblioteca de classe do *.NET Framework*.

- Estruturas de desenvolvimento e tecnologias: O *.NET Framework* inclui bibliotecas para áreas específicas de desenvolvimento de aplicações, tais como *ASP.NET* para aplicações web, *ADO.NET* para acesso a dados, e o *Windows Communication Foundation* para aplicações orientadas a serviços.

- Interoperabilidade de linguagem: Compiladores de linguagem que visam o *.NET Framework* emitem um código intermediário chamado *Common Intermediate Language* (CIL), que por sua vez, é compilado em tempo de execução pelo *Common Language Runtime*. Com esse recurso, rotinas escritas em uma linguagem são acessíveis para outras, e os programadores podem se concentrar na criação de aplicativos em sua linguagem preferencial.

- Compatibilidade de versões: Com raras exceções, os aplicativos que são desenvolvidos usando uma versão especial do *.NET Framework* podem ser executados sem modificações em uma versão posterior.

- Execução *Side-by-side*: O *.NET Framework* ajuda a resolver conflitos de versão, permitindo que várias versões do *Common Language Runtime* possam coexistir no mesmo computador, e assim um aplicativo pode ser executado na versão do *.NET Framework* para o qual ela foi construída.

- *Multitargeting*: Ao direcionar a biblioteca *Portable Class* do *.NET Framework*, os desenvolvedores podem criar conjuntos que trabalham em múltiplas plataformas *.NET*, como o *.NET Framework*, o *Silverlight*, *Windows Phone 7*, ou *Xbox 360*.

O *.NET Framework* é projetado para atender os seguintes objetivos:

- Proporcionar um ambiente de programação orientada à objetos consistente para que o código seja armazenado e executado localmente, distribuído pela *Internet* ou executado remotamente.

- Para fornecer um ambiente de execução de código que minimiza conflitos de implantação de *software* e versões.

- Para fornecer um ambiente de execução de código que promove execução segura do código, incluindo o código criado por um terceiro desconhecido ou um pouco confiável.

- Proporcionar um ambiente de execução que elimina os problemas no desempenho de scripts ou ambientes.

- Para tornar a experiência do desenvolvedor consistente, através dos diversos tipos de aplicações, tais como aplicativos baseados no *Windows* e aplicativos baseados na *web*.
- Para construir toda a comunicação em padrões da indústria para garantir que códigos baseados no *.NET Framework* podem ser integrados com qualquer outro código.

#### **4.12.4. Requisitos do Sistema:**

Processador com 1GHZ ou superior

512 MB de RAM (1.5 GB caso rodando em uma máquina virtual)

850 MB de espaço no HD (x86) / 2 GB de espaço no hd (x64)

#### **4.12.5. Hardware Utilizado:**

Para o desenvolvimento deste *software* foi utilizado um notebook as seguintes configurações:

- *Windows* 8.1 com arquitetura 64 bits;
- Processador Intel core i7 modelo 3571U 1.9~2.4 *Giga-hertz*;
- Memória RAM de 4 *Gigabytes* DDR3 com frequência de 1600Mhz;
- HD com 500Gigabytes e 5400 de rpm.

#### **4.13. Protótipos de tela do sistema**

O sistema Kairos possui 5 telas, onde as informações estão dispostas de forma simples, o que torna o uso da aplicação fácil de usar.

A figura 30 representa a tela onde deverão ser inseridas as informações referentes aos Cursos, Disciplinas e Turmas.

A interface de usuário para o cadastro de cursos, apresentando um fundo verde-escuro com o título "Cursos" em branco. Abaixo do título, há campos para "Curso" (com "Administração" selecionado), "Quant." (com "1" selecionado) e "Descrição" (com "Semestre" selecionado). Um botão "-" está à direita. Abaixo, um menu suspenso seleciona "1º Semestre". O formulário é dividido em duas seções: "Disciplinas" e "Turmas". A seção "Disciplinas" contém uma lista vazia e, abaixo, campos para "Disciplina" (menu suspenso), "Carga Semanal" (com "01:00" selecionado) e um botão "+". A seção "Turmas" contém uma lista vazia e, abaixo, campos para "Turma" (menu suspenso) e "Turno" (com "Matutino" selecionado) e um botão "+".

Figura 30 - Tela de cadastro de cursos

(FONTE: do autor)

#### 4.13.1. Cursos

Campo “Curso”: Nesse campo poderá ser selecionado um curso já existente para alteração, ou digitado o nome de um novo curso.

Campo “Quant.”: Nesse campo é informado o número de períodos (semestre, anos, etc.).

Campo “Descrição”: Nesse campo é inserida a descrição do período, “semestre”, por exemplo, caso a instituição utilize o regime de aulas semestral.

Botão de deleção de curso “-“: Exclui o curso selecionado

Lista de seleção de período: essa lista é usada para seleção do período, para que possa ter suas disciplinas e turmas alteradas.

Disciplinas do curso: Nessa área pode ser selecionada uma disciplina na lista inferior, ou digitado o nome de uma nova disciplina, e informada a carga horária semanal para que se possa acionar a função do botão “+”, que adiciona a disciplina no período selecionado.

Da mesma forma, o botão de deleção “-“ irá excluir a disciplina do período, previamente selecionada na lista superior.

Turma do curso: Nessa área pode ser digitado o nome de uma nova turma, e informado o turno para que se possa acionar a função do botão “+”, que adiciona a turma no período selecionado.

Da mesma forma, o botão de deleção “-“ irá excluir a turma do período, previamente selecionada na lista superior.

A figura 31 representa a tela onde deverão ser inseridas as informações correspondentes aos Turnos com os horários das aulas.

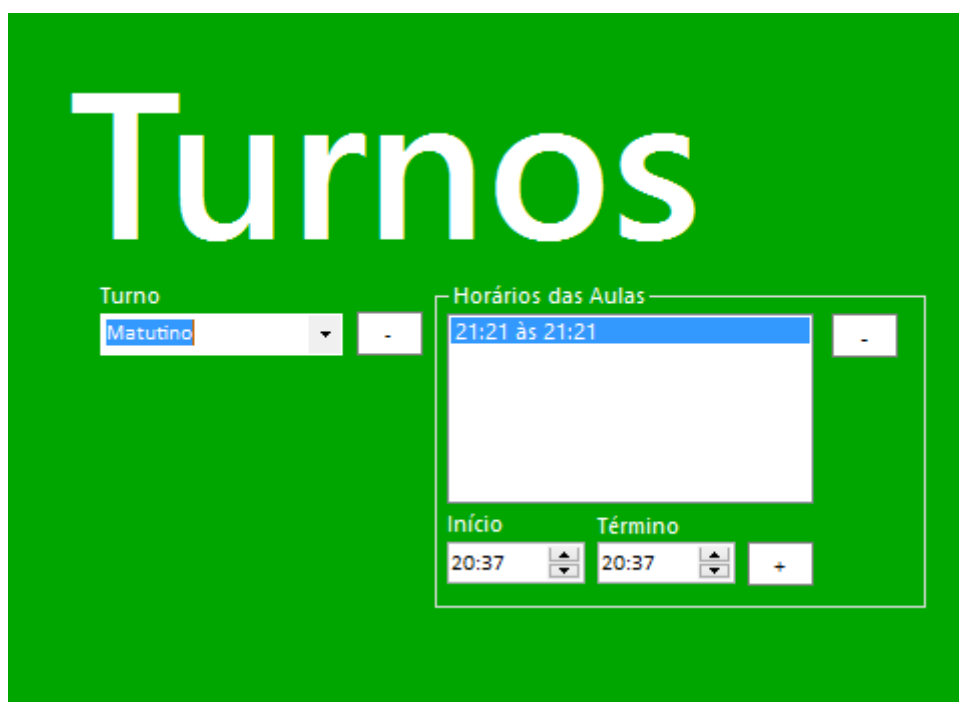
A interface de usuário para o cadastro de turnos, exibida sobre um fundo verde sólido. No topo, o título "Turnos" é escrito em uma fonte branca, grande e sans-serif. Abaixo do título, há duas seções principais. A primeira seção, intitulada "Turno" em verde, contém um menu suspenso com o texto "Matutino" em azul e um botão de seta para baixo. À direita deste menu está um botão quadrado branco com um sinal de menos "-" em verde. A segunda seção, intitulada "Horários das Aulas" em verde, contém uma lista de horários. O primeiro item da lista, "21:21 às 21:21", está selecionado e destacado em azul. À direita desta lista também há um botão quadrado branco com um sinal de menos "-" em verde. Abaixo da lista de horários, há duas colunas de entrada de dados. A primeira coluna, rotulada "Início" em verde, mostra o horário "20:37" em um campo branco com setas de navegação (seta para cima e para baixo) à direita. A segunda coluna, rotulada "Término" em verde, também mostra o horário "20:37" em um campo branco com setas de navegação à direita. À direita destas duas colunas de horários, há um botão quadrado branco com um sinal de plus "+" em verde.

Figura 31 - Tela de cadastro de turnos

(FONTE: do autor)

#### 4.13.2. Turnos

Campo “Turno”: Pode ser selecionado um turno existente para alteração ou digitado o nome de um novo turno.

Campo “Horários das Aulas”: Nessa área deverá ser informado o horário de início e término de uma aula para que se possa acionar a função do botão “+”, que adiciona o horário no período selecionado.

Da mesma forma, o botão de deleção “-“ irá excluir o horário do período, previamente selecionada na lista superior.

A figura 32 representa a tela onde deverão ser inseridas as informações sobre os Professores e suas preferências.

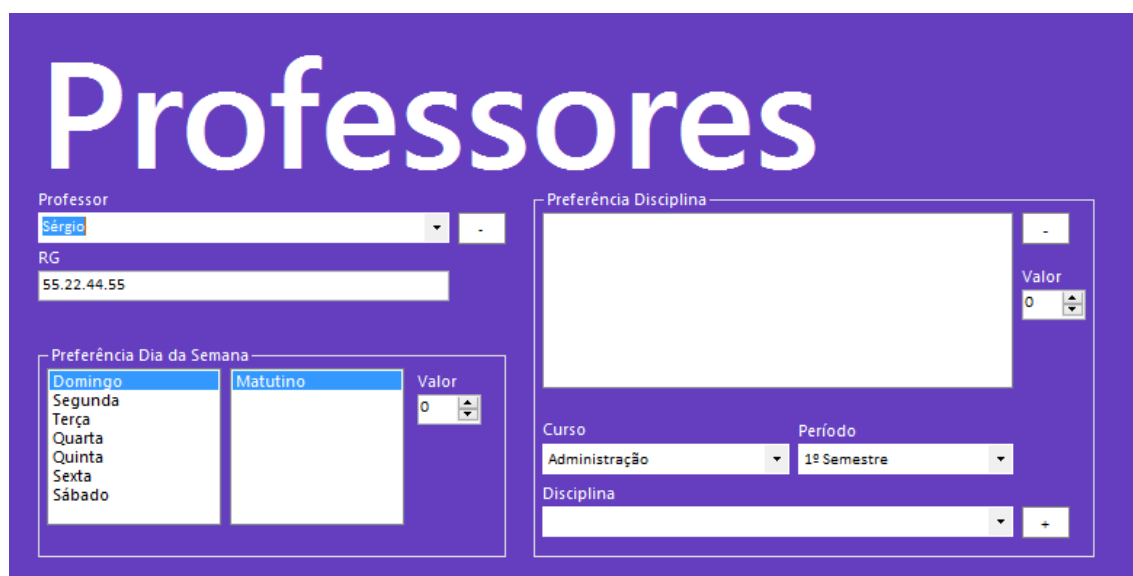
A interface de usuário para o cadastro de professores, intitulada "Professores" em um cabeçalho branco. O formulário é dividido em seções: 1. "Professor": Um campo de texto com o nome "Sérgio" e um botão de deleção "-". Abaixo, um campo "RG" com o valor "55.22.44.55". 2. "Preferência Dia da Semana": Duas listas de seleção. A esquerda, "Dia da Semana", com opções Domingo, Segunda, Terça, Quarta, Quinta, Sexta e Sábado. A direita, "Matutino", com um botão de deleção "-". Um campo "Valor" com o valor "0" e botões de incremento/decremento. 3. "Preferência Disciplina": Um campo de texto grande e vazio, com um botão de deleção "-". Abaixo, campos para "Curso" (Administracão) e "Período" (1º Semestre). 4. "Disciplina": Um campo de texto com um botão de adição "+".

Figura 32 - Tela de cadastro de Professores

(FONTE: do autor)

#### 4.13.3. Professores

Campo “Professor”: Nesse campo poderá ser selecionado um professor já existente para alteração, ou digitado o nome de um novo professor.

Campo “RG”: Nesse campo é informado o RG do professor selecionado.

Botão de deleção de professor “-“: Exclui o professor selecionado

Campo “Preferência Dia da Semana”: Ao selecionar um dia da semana na lista da esquerda e um turno na lista direita, pode-se pontuar (de 0 à 10) a preferência do professor para esse determinado dia/turno.



Campo “Preferência Disciplina”: Ao preencher o curso, período e disciplina, pode-se clicar em “+” para adicioná-lo na lista superior. De forma análoga, ao clicar “-” a disciplina selecionada na respectiva lista é excluída. Enquanto o campo “valor” pontua a preferência do professor para a disciplina selecionada na lista.

A figura 33 representa a tela onde será exibida a grade.

	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
*							

Figura 33 - Tela de exibição e geração das grades

(FONTE: do autor)

#### 4.13.4. Grade

Ao clicar em “Gerar” o sistema irá processar a nova grade automaticamente com base nos dados inseridos previamente no sistema.

Para visualizar a grade gerada pelo sistema deve-se selecionar o curso, o período e a turma desejada.

## CONSIDERAÇÕES FINAIS

O objetivo deste trabalho é utilizar algoritmos genéticos, um conceito de inteligência artificial, para solucionar um problema de análise combinatória pertencente à classe NP-completo.

A geração da grade horária de uma instituição de ensino feita manualmente pode demandar diversos dias de trabalho, além de estar sujeita a erros humanos que acarretariam a uma alteração razoável do que foi feito.

Da mesma forma, sendo esse um problema NP-completo, classe conhecida por sua notável complexidade, um algoritmo que chegue ao melhor resultado possível se torna inviável pelo seu tempo computacional.

A utilização de técnicas heurísticas permite que, embora não se tenha garantia de que a solução encontrada é a melhor, na maioria dos casos é encontrado um resultado satisfatório para o problema proposto.

Com o uso de conceitos não triviais, o trabalho se objetiva em solucionar um problema complexo de modo satisfatório com maior facilidade.

## **IMPLEMENTAÇÕES FUTURAS**

Como trabalhos futuros sugere-se algumas melhorias e implementações no intuito de tornar o sistema mais flexível, tais como:

- Implementação de uma interface online, onde os professores possam ter um usuário para realizar alterações no próprio cadastro;
- Alocação de salas para turmas, com base no espaço da sala em relação à quantidade de alunos, e consideração de laboratórios específicos para determinadas disciplinas;
- Possibilidade de se alterar manualmente a grade gerada pelo sistema;
- Criação de uma versão para dispositivos portáteis;

## REFERÊNCIAS BIBLIOGRÁFICAS

CORMEN, Thomas H. LEISERSON, Charles E. RIVEST, Ronald L. STEIN, Clifford. Tradução. SOUZA, Vandenberg D. de. *Algoritmos Teoria e Prática*. 2. ed. Elsevier: Rio de Janeiro, 2002 – 7ª reimpressão.

GUEDES, Gilleanes T. A.. *UML 2 - Uma Abordagem Prática*. 2ª ed. Editora Novatec: 2011.

KOTSKO et.al. Eliana Gomes da Silva Kotsko, Maria Teresinha Arns Steiner e Artur Lourival da Fonseca Machado. *Otimização na Construção da Grade Horária Escolar*, XXXVSBPO, 2003.

LIMA JÚNIOR, Helio Ferreira e CORRÊA, Marcus Vinicius, *Implementação de um Sistema de Alocação de Professores e Disciplinas em Grades Horárias Utilizando Algoritmos Genéticos*, Universidade Anhembi Morumbi, 2010.

LINDEN, Ricardo. *Algoritmos Genéticos*. 3. ed. Ciência Moderna: Rio de Janeiro, 2012.

MANZANO, José Augusto N. G. e OLVEIRA, Jayr Figueiredo de., *Algoritmos – Lógica para desenvolvimento de Programação de Computadores*. 17ª ed. Editora Érica Ltda: 2005

PAIM, Alexandre da Silva, e GREIS, Ivone Chassot, *Abordagem para Elaboração Automatizada de Tabela de Horários Acadêmicos*, XI Seminário intermunicipal de pesquisa da Universidade Luterana do Brasil, 2008.

SALVATTI, Dirceu Douglas. e BARBOSA, Lisbete Madsen. *Algoritmos*. 1. ed. Pearson Makron Books: São Paulo, 1998.

## REFERÊNCIAS WEBGRÁFICAS

**CONSULTANTS,** aSc Applied Software. aScTimeTables. 2012. <<http://www.asctimetables.com/>>. Acesso em 10 ago. 2013

**CRONOS,** sistema. Cronos Horários Escolares. 2012. <<http://www.sistemacronos.com.br/>>. Acesso em: 10 ago. 2013

**LOTIMIZA, Grupo.** Time's Cool. 2012. <<http://timescool.lotimiza.com/>>. Acesso em 10 ago. 2013

**MÉTODO DE ROLETA.** Acesso em: 24 nov. 13 <<http://www.icmc.usp.br/pessoas/andre/research/genetic/>>

**UNIVERSIDADE FEDERAL DE PERNAMBUCO.** (s.d.).<[www.ufpe.br](http://www.ufpe.br)> Acesso em 23 de novembro de 2013, disponível em Universidade Federal de Pernambuco: <[http://www.ufpe.br/cap/index.php?option=com\\_content&view=article&id=322:horario-escolar&catid=22:informes-gerais&Itemid=211](http://www.ufpe.br/cap/index.php?option=com_content&view=article&id=322:horario-escolar&catid=22:informes-gerais&Itemid=211)>

**PAIM, A. D. (2008).** <http://www.ulbra.br/guaiba/>. Acesso em 15 de setembro de 2013, disponível em Ulbra: [guaibra.ulbra.br/seminários/eventos/2008/artigos/administração/376.pdf](http://guaibra.ulbra.br/seminários/eventos/2008/artigos/administração/376.pdf)

## **ANEXO1 - MODELO DE CONTRATO**

### **CONTRATO DE LICENÇA DE USO E PRESTAÇÃO DE SERVIÇOS DE SOFTWARE**

#### **IDENTIFICAÇÃO DAS PARTES CONTRATANTES**

CONTRATANTE: (Nome da Contratante), com sede em (xxx), na Rua (xxx), nº (xxx), bairro (xxx), Cep (xxx), no Estado (xxx), inscrita no C.N.P.J. sob o nº (xxx), e no Cadastro Estadual sob o nº (xxx), neste ato representada pelo seu diretor (xxx), (Nacionalidade), (Estado Civil), (Profissão), Carteira de Identidade nº (xxx), C.P.F. nº (xxx), residente e domiciliado na Rua (xxx), nº (xxx), bairro (xxx), Cep (xxx), Cidade (xxx), no Estado (xxx).

CONTRATADA: (xxx), com sede em São Paulo, na Rua (xxx), nº (xxx), bairro (xxx), Cep (xxx), no Estado (xxx), inscrita no C.N.P.J. sob o nº 98.090.909/0001-90 e no Cadastro Estadual sob o nº (xxx), neste ato representada pelo seu diretor (xxx), (Nacionalidade), (Estado Civil), (Profissão), Carteira de Identidade nº (xxx), C.P.F. nº (xxx), residente e domiciliado na Rua (xxx), nº (xxx), bairro (xxx), Cep (xxx), Cidade (xxx), no Estado (xxx).

*As partes acima identificadas têm, entre si, justo e acertado o presente Contrato de Licença de Uso e Prestação de Serviços de Software, que se regerá pelas cláusulas seguintes e pelas condições descritas no presente.*

## **DO OBJETO DO CONTRATO**

Cláusula 1ª. O presente instrumento tem como objeto a licença de uso do *Software* (xxx) (Nome do *Software*), bem como a prestação de serviços de *Software* para a CONTRATANTE

## **DA LICENÇA DE USO**

Cláusula 2ª. A presente licença de uso do *Software* (xxx) (Nome do *Software*) terá os aspectos da irretratabilidade e da irrevogabilidade.

## **DA PRESTAÇÃO DE SERVIÇOS**

Cláusula 3ª. A prestação de serviços de *software* compreenderá as seguintes atividades: (xxx) (Descrever pormenorizadamente, listando quais serão os serviços prestados pela Contratada).

## **DAS OBRIGAÇÕES DA CONTRATANTE**

Cláusula 4ª. A CONTRATANTE se responsabiliza por fornecer todos os equipamentos necessários à CONTRATADA, a fim de que esta possa ter condições de realizar perfeitamente o serviço contratado, bem como *hardware* e *software* com a configuração fornecida pela CONTRATADA.

Cláusula 5ª. A CONTRATANTE assume a responsabilidade de contratar funcionários com os seguintes conhecimentos técnicos (xxx) (Descrever os programas que os funcionários

devem saber operar), a fim de que possam operar o *Software* (xxx) (Nome do *Software*).

Cláusula 6ª. A CONTRATANTE se compromete também quanto aos termos do contrato de adesão apresentado na instalação do *Software*.

Cláusula 7ª. A CONTRATANTE se responsabilizará pelos problemas decorrentes do uso incorreto do *Software* (xxx) (Nome do *Software*).

### **DAS ATUALIZAÇÕES**

Cláusula 8ª. Fica acertado entre as partes que a CONTRATADA poderá, sem interferência da CONTRATANTE, realizar todas as alterações que reconhecer como necessárias de uma versão para outra do *Software* (xxx) (Nome do *Software*).

### **DO VALOR E FORMA DE PAGAMENTO**

Cláusula 9ª. A CONTRATANTE pagará à CONTRATADA, pela licença de uso do *Software* a quantia de R\$ (xxx) (Valor Expresso), da seguinte forma (xxx) (Mencionar se a quantia será paga em parcelas ou à vista e qual ou quais os dias de pagamento).

Cláusula 10ª. Pela prestação dos serviços de *Software*, a CONTRATANTE pagará à CONTRATADA a quantia mensal de R\$ (xxx) (Valor Expresso), até o dia (xxx) de cada mês.

### **DA RESCISÃO**

Cláusula 11ª. O presente instrumento poderá ser rescindido por qualquer das partes, devendo a outra ser avisada com 30 (trinta) dias de antecedência.



Cláusula 12ª. O contrato também poderá ser rescindido caso uma das partes descumpra o estabelecido nas cláusulas do presente instrumento, cabendo à parte que ocasionou o rompimento do mesmo, o pagamento de multa rescisória, fixada em (xxx)% do valor previsto na cláusula anterior, à outra parte.

### **DO PRAZO**

Cláusula 13ª. O presente contrato terá prazo de (xxx), iniciando-se no dia (xxx), e terminando no dia (xxx).

### **DOS CASOS OMISSOS**

Cláusula 14ª. Os casos omissos serão resolvidos de comum acordo, mediante reunião das partes para tal finalidade, devendo ser elaborado termo aditivo a este contrato e assinado pelas partes contratantes.

### **DAS CONDIÇÕES GERAIS**

Cláusula 15ª. A CONTRATANTE autoriza a utilização de seu nome pela CONTRATADA, podendo esta apresentá-la como sua cliente em peças de propaganda.

## **DO FORO**

Cláusula 16ª. Para dirimir quaisquer controvérsias oriundas do presente contrato, as partes elegem o foro da comarca de (xxx);

Por estarem assim justos e contratados, firmam o presente instrumento, em duas vias de igual teor, juntamente com 2 (duas) testemunhas.

(Local, data e ano).

(Nome e assinatura do Representante legal da Contratante)

(Nome e assinatura do Representante legal da Contratada)

(Nome, RG e assinatura da Testemunha 1)

(Nome, RG e assinatura da Testemunha 2)

## GLOSSÁRIO

*Adaptation in natural and artificial systems* – Adaptação em sistemas naturais e artificiais.

*Crossover* – Cruzamento

*Crossing – Over* – O mesmo que crossover, ou seja

*Evolutionary Strategies* – Estratégias Evolucionais

*Hardware* – Conjunto de peças que formam uma máquina

*Internet* – Junção de rede de computadores mundial.

*Login* – Autenticação interna que procura verificar se determinada pessoa pode ou não ter acesso ao sistema em questão

*On – Line* – Estar disponível na rede

*Random Walk* – Passo Randômico

*Software* – Conjunto de regras e instruções para a realização de tarefas computacionais

*String* – Tipo de variável que aceita diversos tipos de caracteres