

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CÂMPUS PATO BRANCO  
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E  
DESENVOLVIMENTO DE SISTEMAS**

**ADRIANO PARTECA**

**UTILIZAÇÃO DA METODOLOGIA *SCRUM* PARA  
DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE  
ACADEMIA**

**TRABALHO DE CONCLUSÃO DE CURSO**

**PATO BRANCO**

**2014**

**ADRIANO PARTECA**

**UTILIZAÇÃO DA METODOLOGIA *SCRUM* PARA  
DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE  
ACADEMIA**

Trabalho de conclusão de curso de graduação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco.

Orientador: Prof. Me. Newton Carlos Will

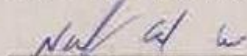
**PATO BRANCO**

**2014**

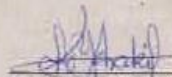
ATA Nº: 252

**DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO ADRIANO PARTECA.**

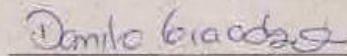
Às 19:00 hrs do dia 17 de dezembro de 2014, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Newton Carlos Will (Orientador), Lucilia Yoshie Araki (Convidada) e Danilo Giacobbo (Convidado), para avaliar o Trabalho de Diplomação do aluno Adriano Parteca, matrícula 1066790, sob o título **Utilização da Metodologia Scrum para Desenvolvimento de um Sistema para Gerenciamento de Academia**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 19:35 hrs foi encerrada a sessão.



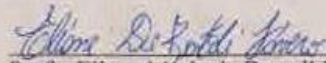
Prof. Newton Carlos Will, M.Sc.  
Orientador



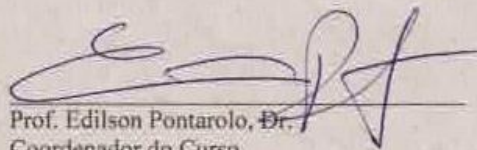
Profa. Lucilia Yoshie Araki, M.Sc.  
Convidada



Prof. Danilo Giacobbo, M.Sc.  
Convidado



Profa. Eliane Maria de Bortoli Fávero, M.Sc.  
Coordenadora do Trabalho de Diplomação



Prof. Edilson Pontarolo, Dr.  
Coordenador do Curso

## RESUMO

PARTECA, Adriano. Utilização da Metodologia Scrum para desenvolvimento de um sistema para gerenciamento de academia. 2014. 78 f. Monografia (Trabalho de Conclusão de Curso). Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2014.

De maneira geral, a procura das pessoas para frequentar uma academia tem sido cada vez maior, pois a busca por um corpo definido ou um melhor condicionamento físico está sendo cada vez mais necessário, tanto para a aparência quanto para uma melhor qualidade de vida. Muitas vezes as formas de controle utilizadas por essas empresas não são as mais adequadas ou elas simplesmente não possuem controle algum. Neste projeto foi realizado o estudo de um software que atenda aos requisitos de uma academia e proporcione uma gestão adequada da mesma. O principal objetivo desse projeto é o gerenciamento das informações, como por exemplo: controle de matrículas, controle de clientes, controle das atividades físicas que cada cliente faz. A melhor maneira encontrada foi a de utilizar a Metodologia Scrum e os paradigmas da orientação a objetos, sendo feito todos os passos necessários para um correto levantamento de dados.

**Palavras-chave:** Gerenciamento de Academias. Orientação a Objetos. Análise e Projeto. Metodologia *Scrum*.

## **ABSTRACT**

PARTECA, Adriano. Application the Scrum methodology for developing a management system for academy. 2014. 78 f. Monograph (Conclusion of course Work). College in Technology Systems Analysis and Development. Federal Technological University of Paraná, Campus Pato Branco. Pato Branco, 2014.

In general, the demand for people to attend an academy has been increasing, as the search for a defined body or a better physical conditioning is being increasingly necessary, either for appearance neither a better quality of life. Often the forms of control used by these companies aren't the most appropriate or they simply don't have any control. In this project was realized a study of a software that contemplate the requirements of an academy and to provide adequate management. The objective of the project is the management of information, for example: control enrollment, customer control, control of physical activities of each customer. The best way found was to use the Scrum methodology and paradigms of object orientation, and made all the necessary phases for correct survey information.

Keywords: Academy Management. Object Orientation. Analysis and Design. Scrum methodology.

## LISTA DE SIGLAS

AOP	Programação Orientada a Aspecto
API	<i>Application Programming Interface</i>
CRUD	<i>Create, Read, Update e Delete</i>
EJB	<i>Enterprise JavaBeans</i>
EL	Expression Language
GSN	Goal Structuring Notation
HTTP	<i>Hypertext Transfer Protocol</i>
IDE	<i>Integrated Development Environment</i>
IIS	<i>Internet Information Services</i>
JDBC	<i>Java Database Connectivity</i>
JDO	<i>Java Data Objects</i>
JPA	<i>Java Persistence API</i>
JSP	<i>JavaServer Pages</i>
MVCC	<i>Multi Version Concurrency Control</i>
SAD	Sistemas de Apoio à Decisão
SIG	Sistema de Informação Gerencial
UML	Linguagem de Modelagem Unificada
USP	Universidade de São Paulo
XP	<i>Extreme Programming</i>

## LISTA DE FIGURAS

Figura 1 - Ciclo de vida de software baseado em ciclos interativos .....	16
Figura 2 - Exemplo das diferentes ênfases do processo unificado .....	16
Figura 3 - Exemplo de Arquitetura dos Sistemas de Apoio à Decisão .....	18
Figura 4 - Tela inicial ferramenta <i>Astah</i> * .....	20
Figura 5 - Tela Spring Tool Suite.....	21
Figura 6 - Exemplo de um quadro kanban .....	27
Figura 7 - Exemplo de um quadro kanban mais simplificado .....	28
Figura 8 - Práticas XP .....	31
Figura 9 - Componentes básicos do Spring .....	32
Figura 10 - Padrão MVC .....	34
Figura 11 - Padrão <i>Front Controller</i> no contexto do <i>Spring MVC</i> .....	35
Figura 12 - O ciclo de vida de uma requisição .....	35
Figura 13 - Sistema de Grids Bootstrap .....	38
Figura 14 - Exemplo de Sistema de Grids Bootstrap .....	38
Figura 15 - Diagrama de Casos de Uso dos Cadastros .....	45
Figura 16 - Diagrama de Casos de Uso dos Demais Controles.....	47
Figura 17 - Diagrama de Classes.....	52
Figura 18 - Tela de Login do Sistema .....	56
Figura 19 - Tela Principal do Sistema .....	56
Figura 20 - Tela de Cadastro de Funcionários .....	57
Figura 21 - Tela de Inclusão de Funcionários .....	58
Figura 22 - Tela de Cadastro de Clientes.....	58
Figura 23 - Tela de Inclusão de Clientes.....	59
Figura 24 - Tela de Cadastro de Contas a Receber.....	59
Figura 25 - Tela de Cadastro de Contas a Pagar.....	60
Figura 26 - Arquivo Context.xml .....	61
Figura 27 - Consulta ao Banco de Dados Pelo Nome do Cliente.....	61
Figura 28 - Consulta ao Banco de Dados .....	62
Figura 29 - Consulta Spring Data.....	62
Figura 30 - Consulta do Spring Data com Paginação .....	63
Figura 31 - Método de Login .....	63

## LISTA DE QUADROS

Quadro 1 - Questionário.....	41
Quadro 2 - Casos de uso dos cadastros .....	46
Quadro 3 - Casos de uso dos demais controles.....	48
Quadro 4 - Caso de uso detalhado Controlar turmas.....	49
Quadro 5 - Caso de uso detalhado Verificar pendências .....	50
Quadro 6 - Requisito Controlar turmas.....	51
Quadro 7 - Requisito Verificar pendências .....	51
Quadro 8 - Requisitos de conceitos .....	54
Quadro 9 - Requisitos Suplementares .....	55
Quadro 10 - Consultas / Relatórios .....	55
Quadro 13 - Caso de uso detalhado Receber pendências.....	67
Quadro 14 - Caso de uso detalhado Controlar pendências a pagar .....	68
Quadro 15 - Caso de uso detalhado Incluir pendências pagas .....	68
Quadro 16 - Caso de uso detalhado Controlar cheques emitidos .....	68
Quadro 17 - Caso de uso detalhado Controlar cheques recebidos.....	69
Quadro 18 - Caso de uso detalhado Matricular aluno .....	70
Quadro 19 - Caso de uso detalhado Cancelar matrícula .....	70
Quadro 20 - Caso de uso detalhado Bloquear matrícula .....	71
Quadro 21 - Requisito Receber pendências.....	72
Quadro 22 - Requisito Controlar pendências à pagar .....	72
Quadro 23 - Requisito Incluir pendências pagas.....	73
Quadro 24 - Requisito Controlar cheques emitidos.....	73
Quadro 25 - Requisito Controlar cheques recebidos.....	73
Quadro 26 - Requisito Matricular aluno.....	74
Quadro 27 - Requisito Cancelar matrícula .....	75
Quadro 28 - Requisito Bloquear matrícula .....	75
Quadro 29 - Requisito Gerar relatórios .....	78



## SUMÁRIO

1 INTRODUÇÃO .....	11
1.1 OBJETIVOS.....	11
1.1.1 Objetivo Geral.....	11
1.1.2 Objetivos Específicos.....	12
1.2 JUSTIFICATIVA .....	12
1.3 ORGANIZAÇÃO DO TEXTO .....	13
2 REFERENCIAL TEÓRICO E MATERIAIS .....	14
2.1 MODELAGEM DE SISTEMAS E ORIENTAÇÃO A OBJETOS.....	14
2.2 LINGUAGEM DE MODELAGEM UNIFICADA (UML) .....	14
2.3 PROCESSO UNIFICADO .....	15
2.4 SISTEMAS DE INFORMAÇÃO GERENCIAL E DE APOIO À DECISÃO .....	16
2.4.1 Sistemas de Informação Gerencial.....	17
2.4.2 Sistemas de Apoio à Decisão (SAD) .....	17
2.4.3 Arquitetura dos Sistemas de Apoio à Decisão .....	17
2.5 ASTAH* COMMUNITY.....	19
2.6 SPRING TOOL SUITE .....	21
2.7 POSTGRESQL .....	22
2.8 APACHE TOMCAT .....	22
2.9 SCRUM.....	23
2.10 EVENTOS SCRUM.....	25
2.10.1 Sprint .....	25
2.10.2 Reunião Diária .....	26
2.11 ARTEFATOS DO SCRUM .....	26
2.11.1 Backlog do Produto.....	26
2.12 KANBAN .....	27
2.13 EXTREME PROGRAMMING (XP).....	29
2.14 LINGUAGEM JAVA.....	31
2.15 SPRING .....	32
2.15.1 Spring MVC .....	33
2.15.2 Dispatcher Servlet.....	34
2.15.3 Spring Security.....	36
2.16 BOOTSTRAP .....	37

2.17 ANGULARJS.....	39
2.18 METODOLOGIA .....	39
3 MODELAGEM E IMPLEMENTAÇÃO DO SISTEMA.....	44
3.1 SISTEMA .....	44
3.2 CASOS DE USO.....	45
3.3 CASOS DE USO COMPLETO.....	49
3.3.1 Níveis de Detalhamento de um Caso de Uso .....	49
3.4 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS.....	50
3.5 DIAGRAMA DE CLASSES .....	51
3.6 REQUISITOS DE CONCEITOS.....	52
3.7 REQUISITOS SUPLEMENTARES .....	54
3.8 CONSULTAS / RELATÓRIOS .....	55
3.9 APRESENTAÇÃO DO SISTEMA.....	56
3.10 IMPLEMENTAÇÃO DO SISTEMA.....	60
4 CONCLUSÃO.....	64
REFERÊNCIAS.....	65
APÊNDICE A - NÍVEIS DE DETALHAMENTO DE UM CASO DE USO .....	67
APÊNDICE B - REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS .....	72

## **1 INTRODUÇÃO**

Uma academia, mesmo não sendo de grande porte, lida com diversos tipos de atividades e também de pessoas. As atividades existentes visam atender desde uma pessoa que sonha com um corpo escultural e definido até aquela pessoa que precisa perder peso e ter uma qualidade de vida melhor. Mas claro que tudo isso deve ter seus devidos cuidados para que o cliente não seja prejudicado com exercícios feitos de forma errada. Para isso não ocorrer é necessário ter um acompanhamento adequado por profissionais especializados na área e também um sistema no qual possam ser armazenadas todas as informações necessárias.

Geralmente no balcão ou mesa na entrada de muitas academias estão guardadas as fichas de cadastro com todas as informações dos clientes, tornando-se com o tempo uma grande quantidade de papéis arquivados e, muitas vezes, desnecessários. Tendo um sistema que consiga controlar isso, a rapidez e segurança nos dados serão muito maiores.

Para os funcionários, a capacidade de armazenar as informações que julgarem úteis dos clientes que praticam algum tipo de atividade na academia aumenta consideravelmente. Isso está sendo cada vez mais necessário, pois informação como estado de saúde, preparo físico e peso da pessoa influenciam nos resultados desejados. Segundo um estudo realizado pela Universidade de São Paulo (USP), a prática de exercícios aeróbios é capaz de interromper o processo degenerativo observado na insuficiência cardíaca (FAPESP, 2014).

Considerando a importância do adequado gerenciamento dos dados de uma academia, é apresentado o projeto de desenvolvimento de um sistema para academia utilizando a metodologia Scrum. É um sistema com finalidades comerciais, visando atender as necessidades dos clientes.

### **1.1 OBJETIVOS**

#### **1.1.1 Objetivo Geral**

Desenvolver um sistema computacional utilizando a metodologia Scrum que auxilie no gerenciamento de academias.

### **1.1.2 Objetivos Específicos**

Para atender e complementar o objetivo geral, os seguintes objetivos específicos foram levantados:

- Fornecer um melhor controle de cadastros de clientes e colaboradores.
- Fornecer dados que facilitem o acompanhamento dos dados históricos da situação física dos alunos da academia.
- Apresentar o controle de contas a pagar e contas a receber de forma a facilitar o trabalho do setor financeiro e a geração de relatórios.
- Emitir relatórios detalhados das atividades realizadas pelos alunos da academia.
- Oferecer uma forma de controle das atividades e das avaliações físicas realizadas pelos alunos da academia.
- Utilizar metodologias específicas para gestão e planejamento do projeto.

### **1.2 JUSTIFICATIVA**

Nos últimos anos, a busca por um corpo perfeito e uma melhor qualidade de vida tem levado mais pessoas a procurar uma academia para realizar seus exercícios físicos. Levando em consideração essa demanda cada vez maior, surgiu a ideia de realizar um estudo e levantar as necessidades que uma academia encontra para gerenciar suas atividades.

Dentro de uma academia, existem várias atividades que podem ser realizadas pelos frequentadores, mas que, muitas vezes não possuem o devido controle. Dentre alguns controles estão: avaliação física e histórico salvo das avaliações realizadas, cadastro do cliente, as atividades físicas que o mesmo deseja fazer, o controle das mensalidades, etc.

Desta forma, neste trabalho é feito o projeto de desenvolvimento de um sistema utilizando a metodologia Scrum visando possibilitar aos instrutores da academia um melhor controle sobre as atividades realizadas pelos frequentadores, melhorando o desempenho e que o resultado almejado pelos alunos seja o melhor possível, trazendo benefícios também ao proprietário por um melhor gerenciamento de sua empresa.

### **1.3 ORGANIZAÇÃO DO TEXTO**

O texto está organizado em capítulos, sendo este o primeiro, apresentando as ideias iniciais do projeto, incluindo os objetivos e a justificativa.

O Capítulo 2 apresenta o referencial teórico e os materiais utilizados, mostrando a importância da modelagem, a modelagem orientada a objetos, o processo unificado e os sistemas de informação gerencial e de apoio à decisão. Quanto aos materiais utilizados, referem-se ao que é necessário para modelar e implementar o projeto proposto, incluindo as tecnologias e ferramentas utilizadas.

O capítulo 3 apresenta a modelagem e a exemplificação da implementação do sistema.

O Capítulo 4 apresenta a conclusão e a perspectiva futura em relação ao projeto. Nos Apêndices A e B estão apresentados os detalhes de casos de uso e os requisitos funcionais e não funcionais do sistema.

## **2 REFERENCIAL TEÓRICO E MATERIAIS**

Neste capítulo serão abordados aspectos relacionados à modelagem, metodologias, tecnologias e sistemas de apoio à decisão utilizados para o projeto.

### **2.1 MODELAGEM DE SISTEMAS E ORIENTAÇÃO A OBJETOS**

A modelagem é uma técnica muito bem aceita. Muitas coisas construídas passam basicamente por um modelo para ter-se uma ideia de qual será o produto final para aprovação do usuário. A construção de modelos é realizada para compreender melhor o sistema que estamos desenvolvendo (BOOCH, RUMBAUGH, JACOBSON, 2006, p. 7).

Devem-se escolher bem os modelos a serem utilizados. Os modelos corretos esclarecerão problemas de desenvolvimento mais complexos, possibilitando conclusões que não seriam possíveis de outra forma. Os modelos inapropriados causarão confusões que poderiam ser evitadas.

Na orientação a objetos, todas as coisas de que o mundo é formado, como exemplos, o cliente, um relatório, um computador, etc., são denominadas de objetos. Seres humanos costumam agrupar os objetos provavelmente para tentar um melhor gerenciamento e entendimento das coisas. É bem mais fácil entender a ideia “empresa” do que entender todas as empresas que existem. Na orientação a objetos, cada ideia é denominada de classe. Segundo Bezerra (2007, p. 7), “uma classe é uma descrição dos atributos e serviços comuns a um grupo de objetos”.

### **2.2 LINGUAGEM DE MODELAGEM UNIFICADA (UML)**

A UML “é uma linguagem padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software” (BOOCH, RUMBAUGH, JACOBSON, 2006, p. 14). É adequada para a modelagem de sistemas de todos os tipos. É uma linguagem muito expressiva, envolvendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas.

A UML não está restrita à modelagem de software. Pode-se utilizá-la para modelar o fluxo de trabalho, a estrutura e o comportamento de sistemas os mais diversos e projetos de *hardware* (BOOCH, RUMBAUGH, JACOBSON, 2006).

### 2.3 PROCESSO UNIFICADO

Segundo Wazlawick (2011), o processo unificado se fundamenta em três valores:

- a) É dirigido por casos de uso: o planejamento do desenvolvimento é feito em função dos casos de uso identificados, tratando-se prioritariamente os mais complexos;
- b) É centrado na arquitetura: o processo de desenvolvimento prioriza a construção de uma arquitetura de sistema que permita a realização dos requisitos. Essa arquitetura baseia-se na identificação de uma estrutura de classes, produzida a partir de um modelo conceitual;
- c) É interativo e incremental: a cada ciclo de trabalho realizado, novas características são adicionadas à arquitetura do sistema, deixando-a mais completa e mais próxima do sistema final. (WAZLAWICK, 2011, p. 4).

No processo unificado, existem quatro grandes fases: concepção, elaboração, construção e transição.

A fase de concepção procura levantar os requisitos, estudar a viabilidade do projeto e a compreensão do sistema de forma mais abrangente. Os resultados são: um documento de requisitos e riscos, uma lista de casos de uso e um cronograma de desenvolvimento.

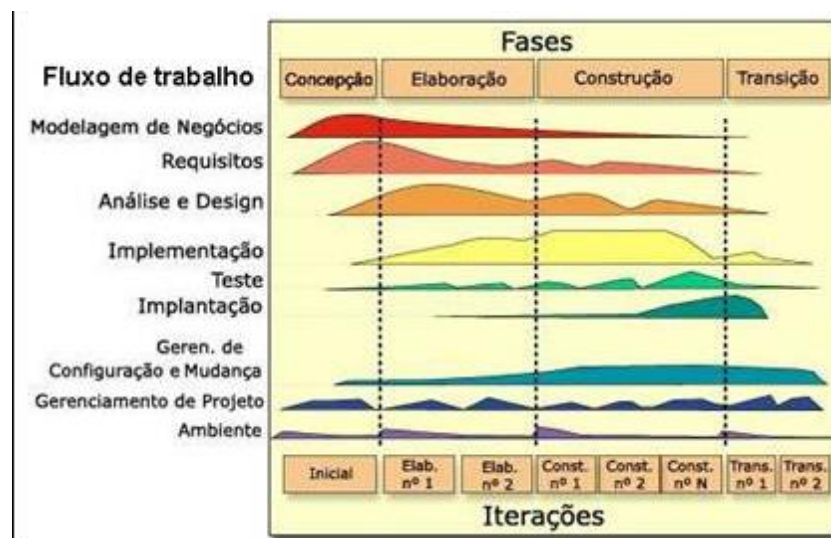
A fase de elaboração apresenta a maior parte da análise e projeto; a construção a maior parte da implementação e testes. É durante os ciclos interativos que acontece a análise detalhada do sistema, a modelagem e o projeto do sistema. Na fase de transição o sistema será implantado. A Figura 1 mostra o ciclo de vida no processo unificado em suas 4 grandes fases.



**Figura 1 - Ciclo de vida de software baseado em ciclos iterativos**

Fonte: Wazlawick (2011).

A Figura 2 é a representação clássica da distribuição das atividades de desenvolvimento de sistemas e sua ênfase nas diferentes fases da implementação mais conhecida do processo unificado.



**Figura 2 - Exemplo das diferentes ênfases do processo unificado**

Fonte: Adaptada de BOOCH, RUMBAUGH, JACOBSON (2006).

## 2.4 SISTEMAS DE INFORMAÇÃO GERENCIAL E DE APOIO À DECISÃO

Neste capítulo será abordado o que são os Sistemas de Informação Gerencial, os Sistemas de Apoio à Decisão e sua arquitetura e suas principais funções ou características.



### **2.4.1 Sistemas de Informação Gerencial**

Para O'Brien "um SIG gera produtos de informação que apoiam muitas das necessidades de tomada de decisão da administração" (O'BRIEN, 2001, p. 250).

Os relatórios gerados por esses sistemas buscam atender as necessidades de informação que os gerentes necessitam para o gerenciamento das empresas. Esses produtos predefinidos de informação atendem as necessidades dos tomadores de decisão dos níveis operacional e tático, obtendo um bom resultado quanto às informações para o gerenciamento das empresas. Os gerentes de vendas, por exemplo, recorrem muito a relatórios de análise de vendas para comparar os desempenhos dos vendedores na empresa.

### **2.4.2 Sistemas de Apoio à Decisão (SAD)**

Segundo O'Brien (2001, p. 253), "São sistemas de informação computadorizados que fornecem aos gerentes apoio interativo de informações durante o processo da tomada de decisão".

Estes sistemas computadorizados fornecem as informações necessárias para o planejamento e organização eficazes. Os SAD oferecem aos administradores uma análise sobre circunstâncias futuras, caminhos a serem seguidos para a tomada de uma decisão. Segundo Gordon (2006), os SAD possuem alguns benefícios:

- Uma melhora no processo da tomada de decisão;
- Um maior número de alternativas para uma decisão;
- A capacidade de implementar análises *ad hoc* ou aleatórias;
- Resposta mais rápida às situações previstas;
- Uma comunicação aprimorada;
- Trabalho de equipe mais eficaz;
- Melhor controle;
- Economia de tempo e de custos.

### **2.4.3 Arquitetura dos Sistemas de Apoio à Decisão**

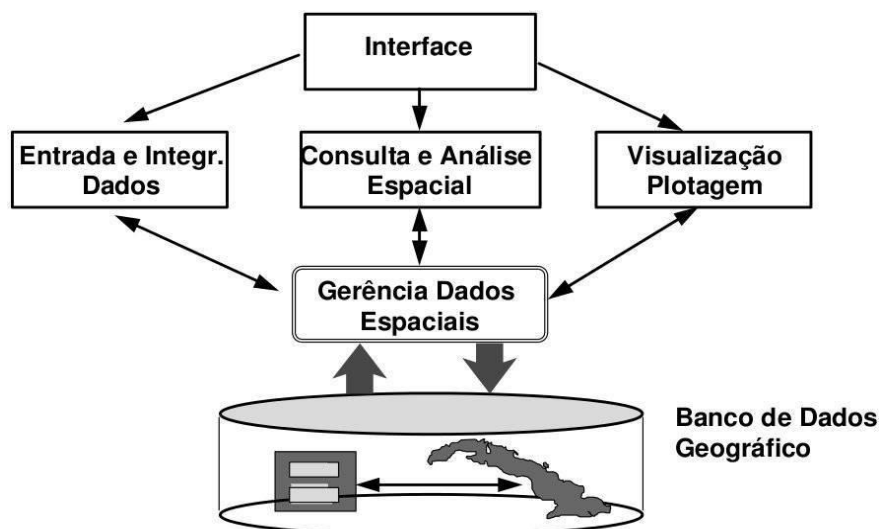
Para Gordon (2006), “Um sistema de apoio à decisão completo consiste em quatro componentes principais: Um banco de dados, uma base de conhecimento, uma base de modelos e uma interface com o usuário” (Gordon, 2006, p. 260).

O banco de dados fornece acesso às informações relativas às decisões. Os dados ajudam na avaliação e validação dos modelos usados nas previsões. A base de conhecimentos fornece informações sobre relacionamentos altamente complexos entre dados que um banco de dados tem dificuldades em representar. Uma base de modelos inclui uma série de ferramentas analíticas para a construção de modelos de processos e atividades de um negócio.

Após isso, inclui-se uma interface de usuário robusta, que permita aos usuários controlar as ferramentas inclusas no sistema para suas respectivas análises. “Um SAD deve ser projetado para suportar a maior liberdade que os usuários experimentam manipulando dados e processando informações” (Gordon, 2006, p. 260).

O administrador deve ter a oportunidade de avaliar o impacto de decisões alternativas em um sistema de apoio à decisão. O SAD deve possuir a capacidade de prever possíveis efeitos que uma decisão pode provocar.

A Figura 3 denota um exemplo de como funciona a arquitetura dos sistemas de apoio à decisão.



**Figura 3 - Exemplo de Arquitetura dos Sistemas de Apoio à Decisão**

Fonte: RIBEIRO (2013)

## 2.5 ASTAH\* COMMUNITY

*Astah\* Community* é o sucessor do *Jude Community*. É uma ferramenta gratuita de modelagem do sistema com suporte para a UML 2.x (ASTAH, 2014). Além da modelagem do diagrama, a ferramenta oferece ajustes de alinhamento e tamanho dos diagramas e exportação das imagens dos diagramas.

É apresentada em várias versões:

- *Astah\* Community*: versão gratuita que suporta todos os diagramas UML básicos;
- *Astah\* Professional*: possui suporte para UML, diagramas de entidade e relacionamento, diagrama de fluxo de dados, CRUD (criar, consultar, atualizar e excluir funcionalidades definidas por cada acesso), fluxograma e mapas mentais;
- *Astah\* UMLPad*: é o primeiro software UML do mundo escrito exclusivamente para o *Ipad*. Possui suporte para diagramas UML, sendo possível integrar ao *Astah\* Professional*;
- *Astah\* SysML*: é uma ferramenta de design de sistemas que apoia o projeto, análise e verificação de sistemas complexos. *SysML* é o padrão da indústria para engenharia de sistemas grandes e complexos;
- *Astah\* GSN (Goal Structuring Notation)*: GSN é uma notação gráfica desenvolvida para a especificação de casos de falta de segurança nos sistemas. Essa ferramenta ajuda na garantia de segurança do sistema;
- *Astah\* Share*: Ferramenta acessada via navegador onde os diagramas ficam hospedados em servidores e podem ser compartilhados com outros usuários do grupo de trabalho.

A figura 4 apresenta a tela inicial da ferramenta *Astah\**, com suas respectivas funções explicitadas.

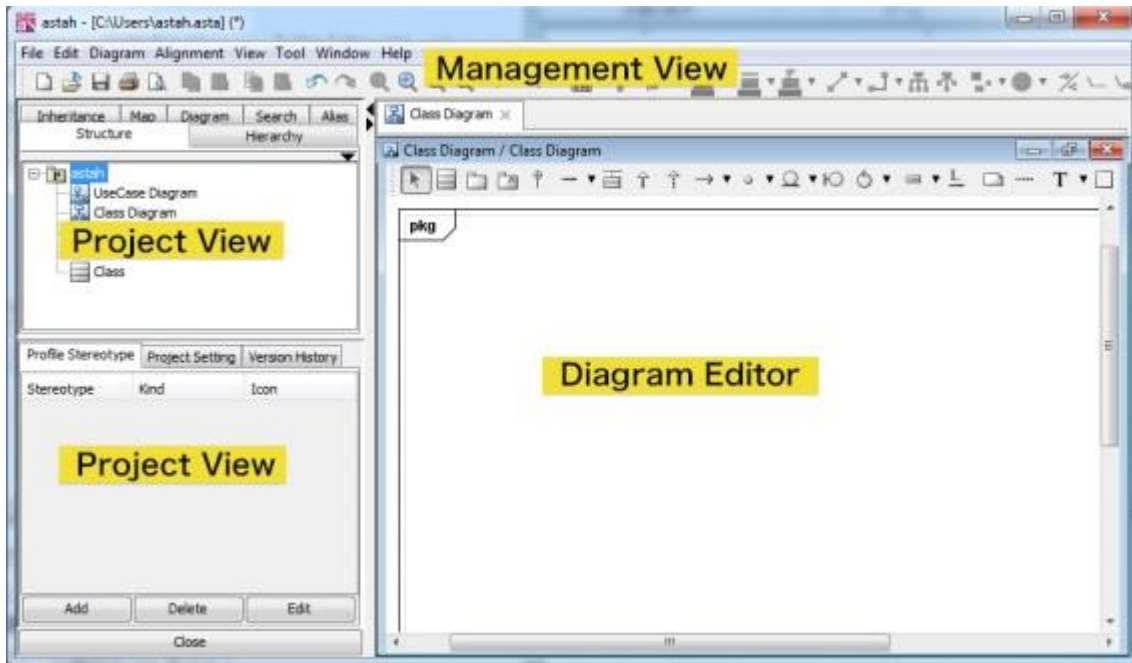


Figura 4 - Tela inicial ferramenta *Astah\**

Fonte: ASTAH (2014)

A IDE da ferramenta *Astah\* Professional*, assim como a ferramenta *Astah\* Community*, está dividida em três partes principais, como pode ser vista na Figura 4 acima:

- **Management View** (Visão gerencial): É utilizado para as operações básicas do *Astah\**. Contém a barra de ferramentas onde são encontrados as funções do programa, como a operação de um arquivo ou edição do mesmo.
- **Project View** (Visão do projeto e propriedades): A tela na parte superior à esquerda fornece uma visão geral de todo o projeto, mostrando por exemplo os diagramas existentes no projeto. Na parte inferior, possui guias de propriedades que são usadas para exibir e editar propriedades de elementos dos modelos.
- **Diagram Editor** (Editor de diagramas): É usado para edição de diagramas e modelos. Vários diagramas podem ser abertos simultaneamente. Basta usar as guias na parte superior para alternar entre os diagramas.

## 2.6 SPRING TOOL SUITE

É uma IDE para desenvolvimento Java, porém suporta várias outras linguagens a partir de *plug-ins* como a plataforma *Android*. Foi construído em java e segue o modelo *open source* de desenvolvimento de software. É um ambiente de desenvolvimento multiplataforma, onde pode-se escrever, compilar, depurar e gerar instaladores. A IDE *Spring Tool Suite* facilita o trabalho por possuir grande conjunto de bibliotecas, módulos e APIs (*Application Programming Interface*).

Sendo feito em Java, a ferramenta *Spring Tool Suite* pode ser executada em qualquer sistema operacional que suporte Java, estando disponível para *Linux*, *Windows* e *Mac OS*.

Na figura 5, na área à esquerda estão os elementos que são criados com a ferramenta. Na parte superior ao centro está o editor de códigos. Na parte inferior central estão os resultados adquiridos do editor.

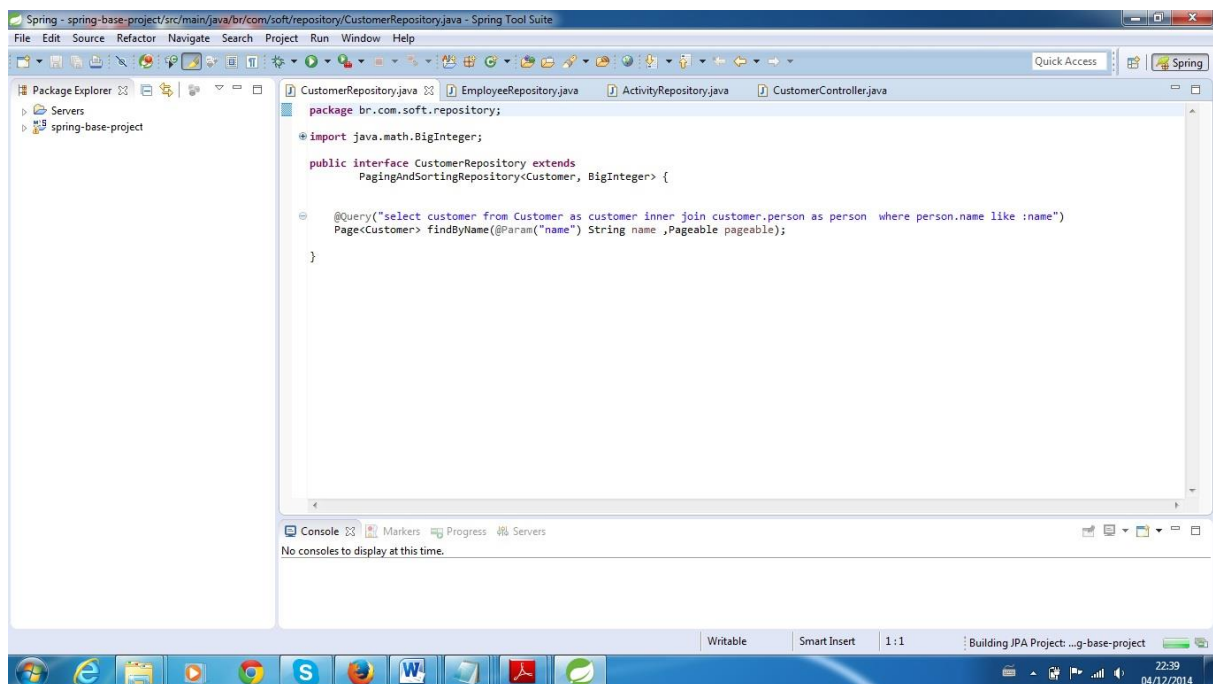


Figura 5 - Tela Spring Tool Suite

Fonte: Autoria própria

## 2.7 POSTGRESQL

*PostgreSQL* é um sistema de banco de dados objeto-relacional com código-fonte aberto (POSTGRESQL, 2014). Funciona nos principais sistemas operacionais, incluindo *Linux*, *Mac OS X*, *Solaris* e *Windows*. Possui suporte total para chaves estrangeiras, *joins*, *views*, *triggers* e *stored procedures*.

Possui funcionalidades sofisticadas como MVCC (*Multi Version Concurrency Control*) ou controle de concorrência de multi versão, onde os processos de leitura não bloqueiam processos de escrita e vice-versa, replicação assíncrona, transações aninhadas (pontos de salvamento), sofisticado planejador de consultas / otimizador, suporta o armazenamento de grandes objetos binários, incluindo sons, imagens ou vídeos. Também suporta conjunto de caracteres internacionais, codificação de caracteres de vários *bytes*. Pode gerenciar uma enorme quantidade de dados e acomodar um grande número de usuários simultâneos.

Outras características avançadas que se pode destacar é a herança de tabelas e o sistema de regras, que permite que se criem regras que identifiquem operações específicas para uma determinada tabela ou exibição.

## 2.8 APACHE TOMCAT

O *Tomcat* (TOMCAT, 2014) é um servidor *web* java, ou seja, um *container* de *servlets*. Implementa as tecnologias *Java Servlet* e *JavaServer Pages*. Não implementa um container EJB, mas abrange parte da especificação Java EE como tecnologias como *servlet* e *JSP*, tecnologias de apoio relacionadas e segurança, como *JNDI Resources* (API para acesso a diretórios) e *JDBC DataSources*.

Pode atuar também como servidor *web*, ou pode funcionar integrado a um servidor *web* dedicado como o *apache* ou o *IIS* (*Internet Information Services*). Como servidor *web*, ele aceita um servidor *web* HTTP puramente em Java. O servidor inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se manualmente arquivos de configuração formatados em XML.

## 2.9 SCRUM

Segundo Schwaber (2013), o *Scrum* é uma metodologia ágil de trabalho onde é usada para estabelecer conjuntos de regras e práticas de gestão para conseguir o sucesso de um projeto. Essa metodologia foi criada em meados de 1990 por Ken Schwaber e Jeff Sutherland, que descrevem que o *Scrum* é um *framework* do qual pessoas podem tratar e resolver problemas complexos e adaptativos. Consiste ainda em times associados em papéis, eventos, artefatos e regras no qual cada componente serve a um propósito específico e é essencial para o uso e sucesso do *Scrum*.

Ainda segundo Schwaber (2013) o Scrum deixa clara a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que você possa melhorá-las. É fundamentado nas teorias empíricas de controle de processo. Essa teoria afirma que os fatos são baseados somente em experiências vividas e presenciadas, nos conhecimentos adquiridos no dia-a-dia.

O empirismo é apoiado em três pilares: transparência, inspeção e adaptação.

- a) **Transparência:** Garante que todos os fatores e aspectos que possam vir a afetar o resultado final de um determinado processo estejam visíveis e seja do conhecimento de todos os envolvidos. Esta transparência requer aspectos definidos por um padrão para que os observadores compartilhem um mesmo entendimento. Nos projetos que utilizam o Scrum, as informações são atualizadas em tempo real e permanecem visíveis para todos os envolvidos através do quadro de tarefas.
- b) **Inspeção:** Os usuários devem inspecionar os artefatos Scrum e o progresso em direção a detectar variações. Esta inspeção não deve ser tão frequente a ponto de atrapalhar a própria execução das tarefas. São benéficas quando feitas por especialistas no trabalho e de forma correta.
- c) **Adaptação:** Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, o processo ou material deve ser corrigido. O ajuste deve ser feito o mais breve possível para não impactar em mais desvios.

No Scrum, quem trabalha nos processos é o time inteiro, não existindo distinção de cargos (SCHWABER, 2013). Existem apenas três papéis em uma equipe: *Product Owner*, Equipe de desenvolvimento e Scrum Master.

Segundo Ken Schwaber (2013) que é um dos criadores da metodologia, o *product owner* é o dono do produto. Sua maior responsabilidade é gerenciar o *backlog* do produto (lista onde se encontram os requisitos do sistema). O gerenciamento do *backlog* inclui:

- Expressar claramente os itens do *backlog* do produto;
- Ordenar os itens do *backlog* do produto;
- Garantir o valor do trabalho realizado pela equipe de desenvolvimento;
- Garantir que o *backlog* seja visível, transparente, claro para todos;
- Garantir que a equipe entenda todos os itens do *backlog*.

Apenas essa pessoa é a responsável por essa tarefa. É sua função também entender o produto e passar uma visão clara sobre os objetivos para a equipe.

A equipe de desenvolvimento é formada por profissionais que realizam o trabalho de entregar uma versão utilizável e incremental do produto. São estruturados e autorizados para organizar e gerenciar seu próprio trabalho. As equipes têm as seguintes características:

- Eles são auto-organizados;
- Equipes de desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias para criar o incremento do produto;
- A responsabilidade pertence a equipe de desenvolvimento mesmo que os integrantes tenham habilidades especializadas e área de especialização;
- Não possui sub-times dedicados a domínios específicos de conhecimento, como teste ou análise.

O tamanho ideal da equipe é algo que deve ser tomado muito cuidado, pois se a equipe for muito pequena, a produtividade da equipe diminui acarretando em atrasos no projeto. Caso a equipe seja muito grande o empirismo fica difícil de ser gerenciado, precisando de muita coordenação.

Já Schwaber (2013) descreve o Scrum Master como o responsável por garantir que o Scrum seja entendido e aplicado. Mas ele não é gerente de projetos. É um mediador entre o *product owner*, a equipe de desenvolvimento e a organização dos projetos. Sua responsabilidade é manter foco no processo ajudando aqueles que estão fora da equipe a entenderem o projeto.



O Scrum Master auxilia o *product owner* de várias formas. Encontra técnicas para o gerenciamento do *backlog* do produto, comunica a visão, objetivo e itens do *backlog* do produto para a equipe, ensina a criar itens do *backlog* de forma correta, facilita os eventos do projeto conforme necessário.

O Scrum Master também auxilia a equipe de desenvolvimento. Treina o gerenciamento próprio da equipe de desenvolvimento, lidera a equipe no desenvolvimento de produtos valiosos, remove impedimentos da equipe quanto ao avanço do projeto, treina a equipe em ambientes onde o Scrum não é totalmente adotado ou compreendido.

## 2.10 EVENTOS SCRUM

Em cada evento no Scrum há chance de se examinar e adaptar algo. Para que se tenha transparência e uma boa inspeção é que esses eventos são projetados.

### 2.10.1 Sprint

É o coração do Scrum e nada mais é que um período de até um mês, onde uma versão utilizável do produto é criada. Uma *sprint* nova inicia-se após a conclusão da *sprint* anterior. É feita uma reunião de planejamento para definir o trabalho a ser realizado pela *sprint*.

Durante a *sprint* não são feitas mudanças que coloquem em risco o objetivo da *sprint*, as metas de qualidades não mudam, o escopo fica mais claro. “Cada *sprint* tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto” (Schwaber, 2013).

Quanto ao cancelamento de uma *sprint*, ela pode ser cancelada antes do *time-boxed* (tempo estipulado para uma *sprint*) terminar. Mesmo podendo ser influenciado pela equipe de desenvolvimento ou scrum master, apenas o *product owner* pode cancelar uma *sprint*. O cancelamento de uma *sprint* consome recursos, já que é feita outra reunião entre a equipe de desenvolvimento para o planejamento de uma nova *sprint*.

Ao final de uma *sprint*, é feita uma reunião para revisão da *sprint* e inspecionar o *backlog* do produto. Durante a reunião, a equipe colabora sobre o que

foi feito. O Scrum Master garante que a reunião ocorra de forma adequada. O resultado da reunião é um *backlog* do produto revisado (Schwaber, 2013).

### **2.10.2 Reunião Diária**

É um evento rápido para que a equipe de desenvolvimento sincronize as atividades e crie um plano para as próximas 24 horas. Essa reunião é utilizada pela equipe de desenvolvimento para verificar se o progresso irá completar o *backlog* da *sprint*. As reuniões diárias melhoram a comunicação, eliminam outras reuniões, identificam e removem impeditivos para o projeto, promovem tomadas de decisões rápidas. Quem assegura para que essa reunião aconteça entre a equipe de desenvolvimento é o Scrum Master.

## **2.11 ARTEFATOS DO SCRUM**

Os artefatos do Scrum representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação.

### **2.11.1 Backlog do Produto**

“É uma lista ordenada de tudo que deve ser necessário no produto e é uma origem única dos requisitos para qualquer mudança a ser feita no produto” (Schwaber, 2013). O responsável pelo *backlog* é o *product owner*.

O *Backlog* do Produto muda constantemente para verificar e adequar-se as necessidades do projeto, pois nunca está completo. No início são apenas estabelecidos os requisitos mais conhecidos e melhor entendidos.

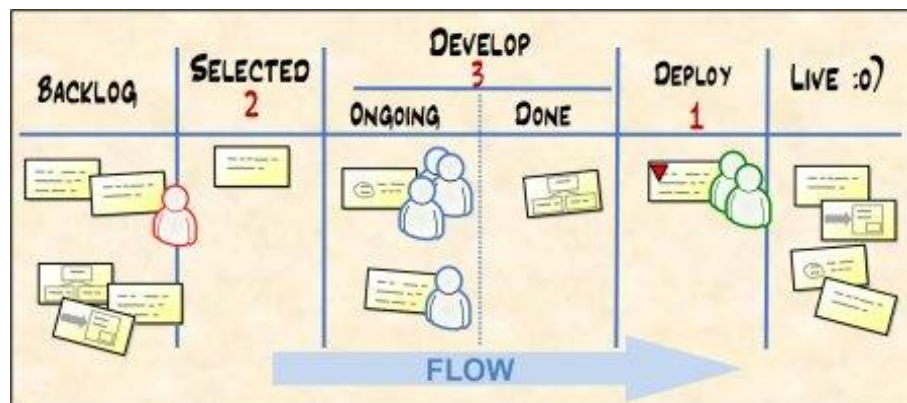
É formado por todas as características, funções, requisitos, melhorias e correções sobre as mudanças a serem feitas em futuras versões. Os itens do *Backlog* do Produto possuem os atributos de descrição, ordem, estimativa e valor (Schwaber, 2013).

## 2.12 KANBAN

O termo *kanban* significa “cartão”, pois ele faz uso de cartões (*post-its*) para indicar o andamento do fluxo dos processos nas empresas. Foi inicialmente usado em empresas japonesas, sendo a Toyota a primeira a utilizar esse método devido à necessidade de um bom funcionamento de produção em série.

É uma ferramenta ágil de gerenciamento de mudança e não é uma ferramenta para gerenciamento de projetos. Vem sendo utilizado por várias áreas, com objetivo de controlar melhor as tarefas diárias. Metodologias como o Scrum utilizam-se do quadro do *kanban*, pois o mesmo oferece uma visão mais ampla do trabalho a ser feito e o status do projeto em andamento. Essa técnica permite de forma rápida analisar questões como o que já foi feito, quanto tempo pode levar um processo, o que pode ser feito para retomar uma fase bloqueada, entre outros.

Outro fato importante é que não existe um *Kanban* padrão. Ele pode ser construído da forma que a empresa achar conveniente, ter quantidade de linhas e colunas que acharem necessárias para garantir a visibilidade dos processos. A figura 6 abaixo apresenta um exemplo de um quadro *kanban*.



**Figura 6 - Exemplo de um quadro kanban**

Fonte: DEVMEDIA (2014)

A figura 6 mostra que o quadro foi dividido em seis colunas com várias tarefas de acordo com o seu desenvolvimento. Desse modo, fica transparente, por exemplo, todas as tarefas que estão acumuladas, quais estão para serem feitas e as que já foram terminadas.

Ainda existem alguns princípios básicos da ferramenta:

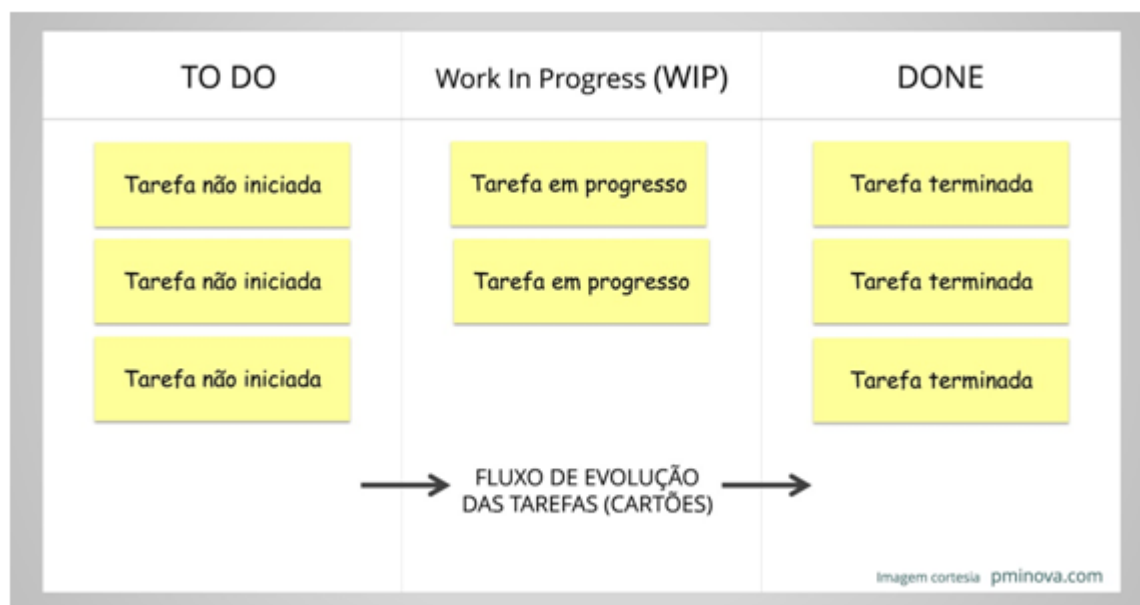
- Visualização da cadeia de valor, enxergando as fases do produto;
- Desenvolvimento adaptativo, entregando o que tem mais valor antes;
- Permite medição, controle e melhoria contínua restringindo o processo em torno de seus estágios.

A figura 7 mostra outro exemplo de um quadro *kanban*.

O *Scrum* e o *Kanban* possuem algumas semelhanças entre eles. Ambos utilizam controle de cronograma, sendo cada um de uma forma. O *Scrum* baseia-se em iterações de tempo fixo para combinar as atividades e no *Kanban* a equipe é que decide como planejar, melhorar e entregar o processo.

Ambos limitam atividades em andamento. No *Scrum*, são definidos quantos itens serão feitos a cada iteração. Já no *Kanban* são definidos quantos itens podem estar em uma coluna ao mesmo tempo. Também caracterizam-se pela entrega de softwares o mais rápido possível e sucessivamente, entregando partes do software funcionando para que o cliente veja como está indo o trabalho.

Possuem base em equipes auto organizáveis, o trabalho é dividido em partes, usam a transparência dos dados para a melhoria do processo.



**Figura 7 - Exemplo de um quadro kanban mais simplificado**

Fonte: DEVMEDIA (2014)

## 2.13 EXTREME PROGRAMMING (XP)

É um método de desenvolvimento de software criado em 1997 não prescritivo e que procura fundamentar suas práticas por um conjunto de valores (DEVMEDIA, 2014). O objetivo principal do XP é levar as boas práticas na engenharia de software como o teste, pois procurar por erros gera muita perda de tempo, sendo que sempre estar testando melhora-se o código perdendo menos tempo. Possui muitos princípios em comum com o Scrum.

O XP já foi utilizado por grandes empresas como Chrysler no seu sistema de folha de pagamento e a Ford Motors Company VCAPS System que utilizava métodos tradicionais e apresentava problemas. Após utilizar a metodologia XP resolveu seu problema.

O XP descarta muita documentação, burocracia, processos pesados, entre outros. É um processo leve que tem o foco em pessoas. Tem muito de gerencial, mas está muito mais centrado em princípios técnicos. É um muito mais complicado para se implementar do que o Scrum, sendo necessário um plano de transição adotando as técnicas do XP aos poucos, ao invés de colocar em prática todas as técnicas.

Essa metodologia possui vários pontos em comum com o Scrum, como os princípios de métodos ágeis e a filosofia enxuta. Os valores do XP são a simplicidade, o *feedback*, a comunicação, a coragem e o respeito.

As práticas do XP são as seguintes (DEVMEDIA, 2014):

- **Versões Pequenas:** São *releases* pequenos e frequentes implantadas no cliente, sendo que as funcionalidades mais importantes são desenvolvidas primeiro para serem entregues ao cliente. Auxilia muito no processo de aceitação por parte do cliente.
- **Jogo de Planejamento:** No início da semana, desenvolvedores e cliente reúnem-se para definir as prioridades do projeto. Nessa reunião o cliente identifica as prioridades e os desenvolvedores as estimam. O cliente assim fica sabendo o que está acontecendo e o que irá acontecer no projeto.
- **Teste:** Os testes se tornam a caracterização da programação, mostrando o que está e o que não está certo. Envolve a presença do

cliente no desenvolvimento e na validação dos testes. O teste proporciona confiança ao sistema, pois podemos saber imediatamente se algo acarretou em algum erro no sistema.

- **Programação em pares:** Trata-se de duas pessoas trabalhando em uma máquina, onde um programa e outro faz críticas ou sugestões. Cada dupla sempre troca de lugar. Esse método melhora a comunicação e o aprendizado. Com isso tem-se como resultado um projeto com maior qualidade e produtividade.
- **Projeto Simples:** É um princípio do XP. São projetos flexíveis a mudanças. Geralmente projetos flexíveis possuem custo alto. Mas com a metodologia XP não se torna caro, pois se utiliza de ciclos curtos. No XP o projeto simples é aquele que passa em todos os testes, tem o menor número possível de classes e métodos, entre outros.
- **Refatoração:** A refatoração significa melhorar o código sem alterar sua funcionalidade. Permite a melhoria contínua da programação com mínimo de erros e mantendo a compatibilidade com o código existente.
- **Propriedade Coletiva:** O código fonte não tem dono e todos podem modificar o código a qualquer momento. É uma forma de evitar problemas como a troca de pessoas na equipe.
- **Integração Contínua:** Todo código pronto deve ser integrado imediatamente ao projeto, fazendo todos os testes antes e depois da integração.
- **Cliente presente:** Clientes devem estar presentes para inscreverem testes de aceitação e definirem prioridades.
- **Semana de 40 horas:** Trabalho com qualidade e ritmo saudável sem horas extras. Horas extras apenas quando tiver produtividade. Se existir muitas horas extras pode ser um sinal de que algo está errado.
- **Padrões de codificação:** Como todos da equipe fazem o que for necessário nos códigos, a melhor maneira é que se defina um padrão de codificação no início dos projetos.
- **Metáfora:** Facilita a comunicação com o cliente, entendendo a realidade dele. É uma linguagem comum que todos devem possuir.

Serve para traduzir as palavras de um cliente para um significado que ele espera dentro do projeto.

- **Reunião diária:** Todos fazem uma reunião em pé rápida para debater o que foi feito no dia anterior, o que será feito no dia e se há algum impedimento.

Logo abaixo, a figura 8 demonstra as práticas do XP.

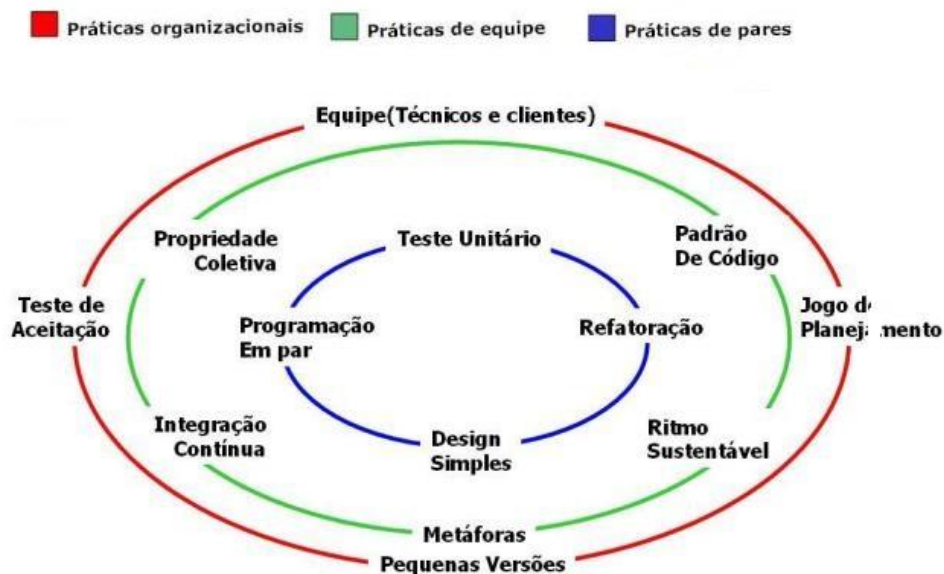


Figura 8 - Práticas XP

Fonte: DEVMEDIA (2014)

## 2.14 LINGUAGEM JAVA

É uma linguagem de programação orientada a objeto criada na década de 90 pela empresa Sun Microsystems. A linguagem Java é compilada para um *bytecode* que é executado por uma máquina virtual.

Programas Java consistem em partes chamadas classes. Dentro das classes estão os métodos que realizam tarefas e retornam informações quando as tarefas são concluídas. Podem-se criar suas próprias classes para formar algum programa Java. Mas a própria linguagem possui extensas bibliotecas de classes, conhecidas como Java APIs (DEITEL, DEITEL, 2010).

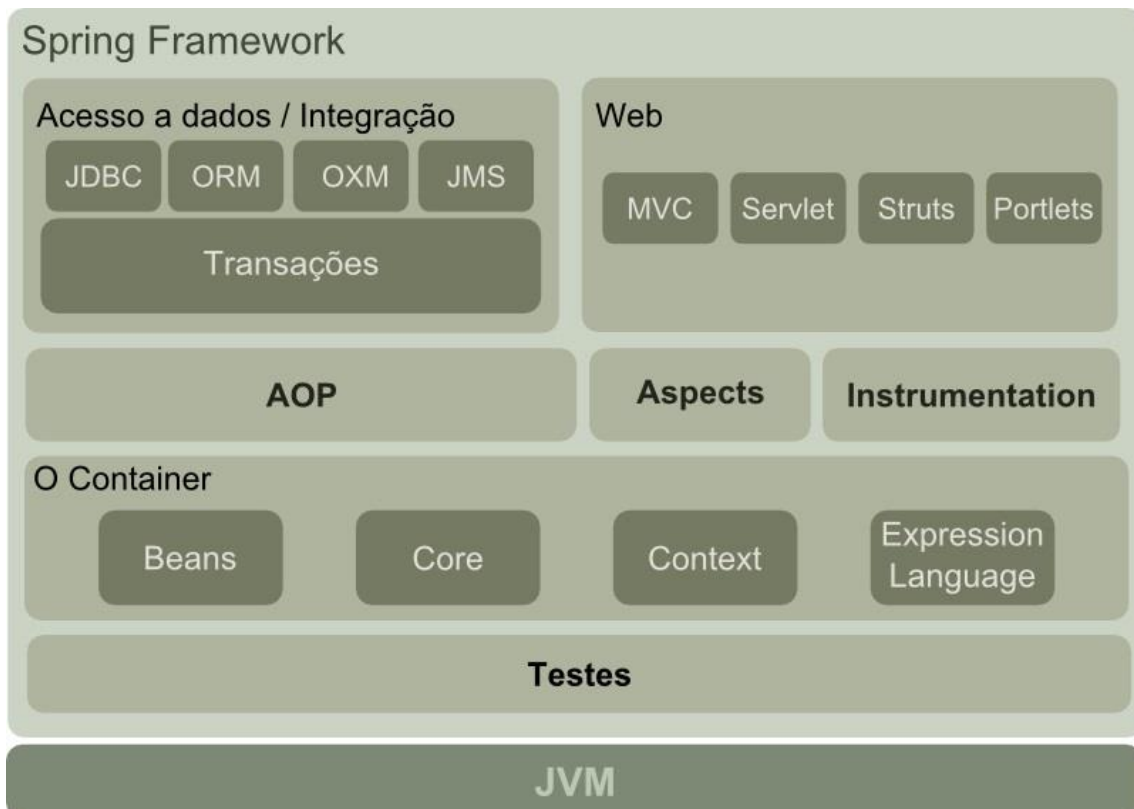
Além de ser uma linguagem orientada a objeto, o Java possui outras características como portabilidade independente de plataforma, recursos de rede, segurança, sintaxe parecida com C/C++, facilidade de internacionalização, facilidade

para criação de programas distribuídos e multitarefa, possui vasto conjunto de bibliotecas, desalocação de memória automática por processo, carga dinâmica de código. Não podemos esquecer que possui licença gratuita.

## 2.15 SPRING

“*Spring Framework* é um *framework* voltado para o desenvolvimento de aplicações corporativas para a plataforma Java, baseado nos conceitos de inversão de controle e injeção de dependências” (WEISSMANN, 2012, p. 3). Foi criado por Rod Johnson para amenizar os problemas que ele encontrava ao utilizar a tecnologia de *Enterprise JavaBeans*.

É formado por seis componentes: o *container* de inversão de controle, suporte a AOP, instrumentação, acesso a dados/integração, suíte de testes e *web* (WEISSMANN, 2012). A figura 9 mostra os componentes que compõe a *Spring*:



**Figura 9 - Componentes básicos do Spring**

Fonte: WEISSMANN (2012)



**O Container** – Os módulos *core* e *beans* são o núcleo do *framework* onde são implementados o suporte à inversão de controle e injeção de dependências. No módulo contexto encontra-se a implementação do *ApplicationContext*. O módulo *Expression Language* fornece uma linguagem muito parecida com a EL que é utilizado com JSP, mas nesse caso é voltado para a configuração do container.

**AOP e Aspects** – É um *framework* que implementa a programação orientada a aspectos. Fornece acesso a recursos até então disponíveis apenas em servidores de aplicações pesados com *WebLogic*, *WebSphere*, *JBoss*, entre outros.

**Instrumentação de Código** – Facilita a vida do suporte dando facilidades na implementação de JMX. Esta tecnologia permite acompanhar tudo o que acontece com o sistema, gerando diversas estatísticas.

**Acesso a dados e integração** – Oferece suporte às tecnologias como JDBC, ORMs como *Hibernate*, *iBatis*, JPA, JDO e OXM. Além disso, oferece uma nova hierarquia de exceções mais interessante do que as oferecidas originalmente pelas tecnologias que são abstraídas.

### 2.15.1 Spring MVC

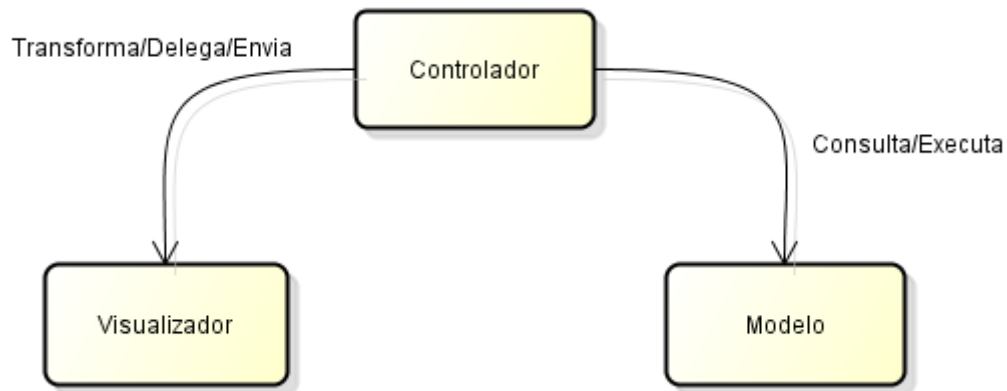
O *Spring MVC* é um *framework* moderno que usa recursos atuais da linguagem e do *container Spring*. No início, Spring era um container leve com o objetivo de fornecer serviços para a aplicação como, por exemplo, o gerenciamento de objetos ou transação. Inicialmente não foi criado para desenvolvimento *web*, mas como o *Struts* estava sendo considerado ultrapassado, começou um método próprio para programar.

O *Struts* foi lançado no ano 2000, sendo um dos primeiros *frameworks MVC* com a ideia de se criar um controlador reutilizável entre projetos. O objetivo era tornar mais fácil a criação de aplicações *web* com a linguagem Java. Atualmente é visto como um *framework* que demanda muito trabalho, por ter sido criado há muito tempo, quando ainda não existiam muitas das facilidades do Java.

O *Spring MVC* é também a aplicação de todas as boas práticas de projeto, o que o torna uma alternativa bem viável na criação de novas aplicações *web*. É o padrão de projeto mais popular adotado pelos *frameworks* de desenvolvimento de aplicações *web*. Sua estratégia permite isolar, em teoria, as camadas de negócio

(modelo) e visualização através de uma camada intermediária que é o controlador (WEISSMANN, 2012).

A figura 10 mostra o padrão adotado pelo MVC.



**Figura 10 - Padrão MVC**

Fonte: WEISSMANN (2012)

“O modelo diz respeito à toda parte do sistema responsável pela lógica do negócio e seus componentes auxiliares, como persistência, cacheamento e integração com outros sistemas” (WEISSMANN, 2012, p. 132).

A visualização é a parte visível ao usuário final. Quando olhamos para uma janela ou página HTML, estamos olhando para o resultado final dessa camada (WEISSMANN, 2012).

O controlador gerencia a interação entre o modelo e o visualizador. Quando o usuário clica em um link o controlador é acionado transformando os parâmetros de entrada para um formato compatível com a interface disponibilizada pela camada de negócio (modelo) e o resultado é recebido pelo controlador, modificado quando necessário e envia à camada de visualização (WEISSMANN, 2012).

### 2.15.2 Dispatcher Servlet

É o componente responsável por gerenciar o funcionamento do *Spring MVC*. É o padrão *Front Controller* usado na escrita de frameworks voltados para a criação de aplicações *web*. Tem como objetivo fornecer um ponto de entrada central para todas as requisições direcionadas a aplicação. O *Dispatcher Servlet* é responsável

por interpretar esses requisitos e decidir qual o componente responsável por seu processamento e retorno para o usuário (WEISSMANN, 2012).

A figura 11 mostra uma visão global de como esse padrão age dentro do *Spring MVC*.

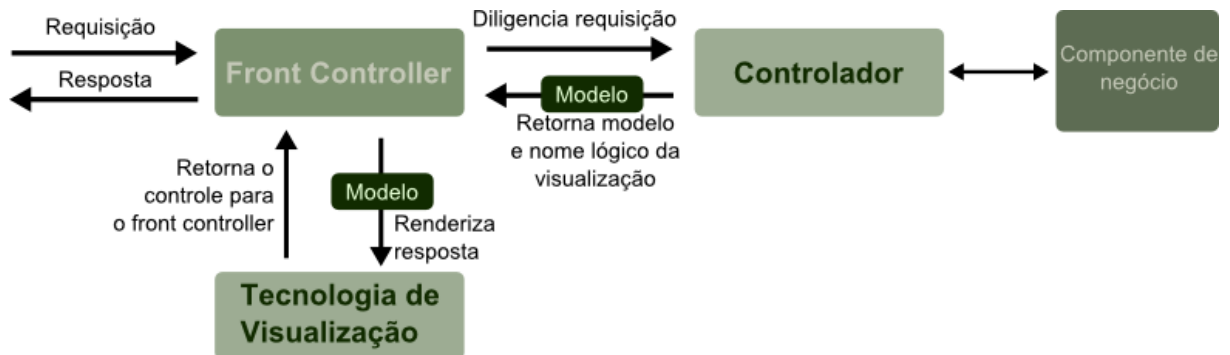


Figura 11 - Padrão *Front Controller* no contexto do *Spring MVC*

Fonte: WEISSMANN (2012)

Toda chamada ao servidor inicia-se com uma requisição e termina como uma resposta enviada ao cliente. A imagem a seguir mostra os cinco eventos que ocorrem durante o caminho de requisição e resposta.

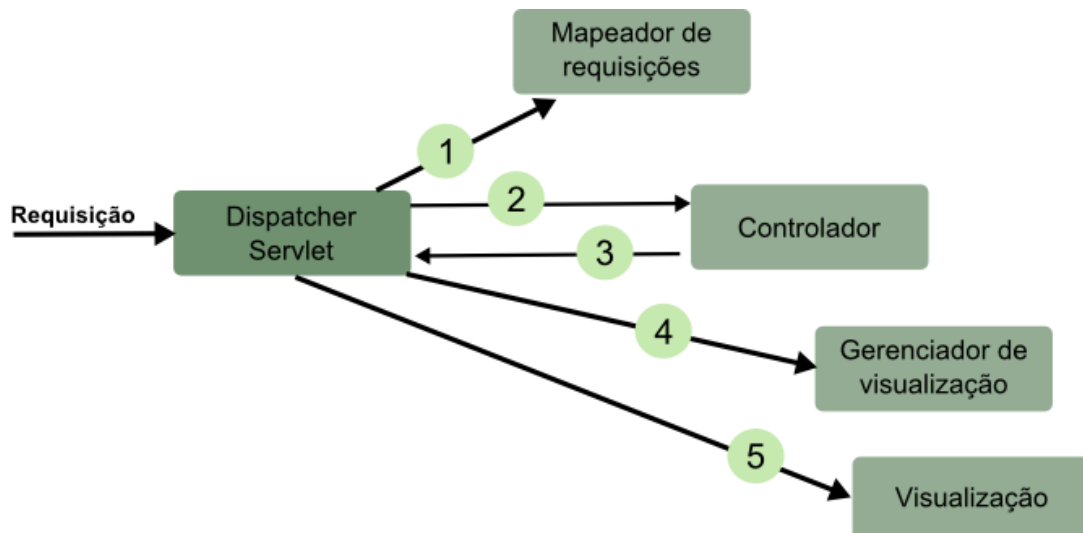


Figura 12 - O ciclo de vida de uma requisição

Fonte: WEISSMANN (2012)

Tudo começa quando uma requisição chega ao *Dispatcher Servlet*. Ela terá sua assinatura analisada e enviada ao mapeador de requisições que é o componente responsável por descobrir qual controlador deve ser acionado. Após

isso, o controlador é executado e o *template* de visualização a ser renderizado como resposta ao usuário (*view*) é retornado ao *Dispatcher Servlet*. Após isso, o gerenciador de visualização com base no nome da *view*, retorna ao *Dispatcher Servlet* qual é o elemento de visualização será renderizada de volta para o usuário.

### 2.15.3 Spring Security

Para WEISSMANN (2012), “o *Spring Security* é um *framework* de controle de acesso, que nos permite definir de forma declarativa quem acessará o que em nossos sistemas”. O controle de acesso é parte fundamental na segurança de qualquer projeto, mas segurança não é apenas isso. Há vários problemas a serem tratados que o *Spring Security* não resolve.

O controle de acesso pode ser aplicado nas requisições que chegam à aplicação e na invocação de métodos dos *beans* gerenciados pelo contexto do *Spring* (WEISSMANN, 2012).

Segundo WEISSMANN (2012), todo controle de acesso possui dois procedimentos: autenticação e autorização. A autenticação visa garantir se o usuário é realmente o correto, enquanto a autorização verifica se este usuário possui permissão necessária para executar determinada tarefa no sistema.

O processo de autenticação inicia-se com o usuário acessando com seu *login* e sua senha ou dados biométricos. O sistema autenticador gera uma assinatura com as credenciais e verifica a existência do usuário no banco de dados. Após vem o processo de autorização, onde é visto se as credenciais passadas ao sistema têm as devidas permissões de acesso ou quais são as permissões de acesso para aquele usuário.

O *Spring security* possui alguns módulos:

- ***Spring-security-core***: Aqui estão implementados os mecanismos de autorização e autenticação, além de todas as interfaces básicas que são reaproveitadas pelos demais módulos e componentes;
- ***Spring-security-config***: *Namespace* próprio que reduz a quantidade de configuração que o programador precisa digitar;

- **Spring-security-web:** Nesse módulo encontra-se presente o código responsável por lidar com a interceptação de requisições a projetos *web*;
- **Spring-security-taglibs:** Possui uma biblioteca de *tags* que nos permite definir quais áreas das páginas serão acessíveis aos usuários do sistema conforme suas permissões;
- **Spring-security-remoting:** Faz integração com o *Spring Remoting*;
- **Spring-security-ldap:** Suporte a autenticação a partir de servidores LDAP;
- **Spring-security-cas:** suporte a autenticação/autorização por servidores de *single sign n CAS*;
- **Spring-security-openid:** Suporte a *OpenID*.

## 2.16 BOOTSTRAP

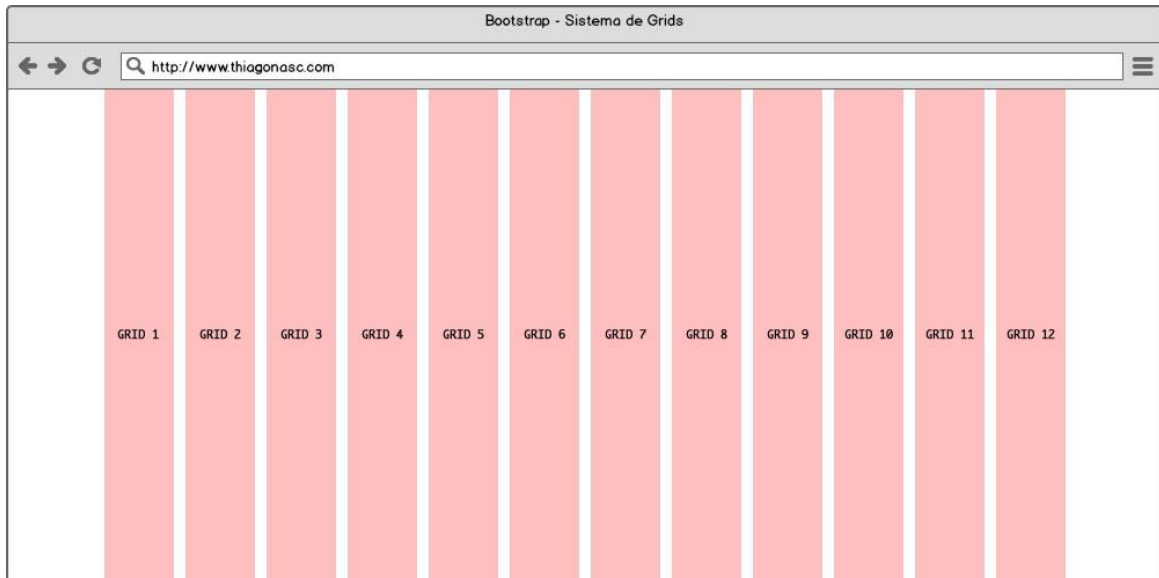
Desenvolvido pela equipe do Twitter, o Bootstrap é um framework front-end, ou seja, é um conjunto de ferramentas criadas para facilitar o desenvolvimento de sites e sistemas web. É compatível com HTML5 e CSS3, possibilitando a criação de layouts responsivos e o uso de *grids*. O objetivo principal é consumir o menor tempo possível no desenvolvimento de um website.

Como toda ferramenta possui vantagens e desvantagens. Como vantagens podemos citar que possui documentação detalhada e de entendimento fácil, possibilita a criação de layouts responsivos, possui inúmeros componentes possibilitando a criação de qualquer página web, por manter padrões, facilita a criação e edição de layouts e funciona em todos os navegadores atualmente.

Como desvantagens podemos citar que há um padrão de desenvolvimento que o código deverá seguir, deve fazer ajustes visuais para que não fique o tema padrão do bootstrap utilizado por muitos usuários.

Possui uma estrutura descomplicada, contendo em seu pacote três tipos de arquivos (CSS, JavaScript e Fonts). Toda a estrutura do CSS já vem definida, bastando procurar o componente adequado na documentação do Bootstrap e adicionar o código.

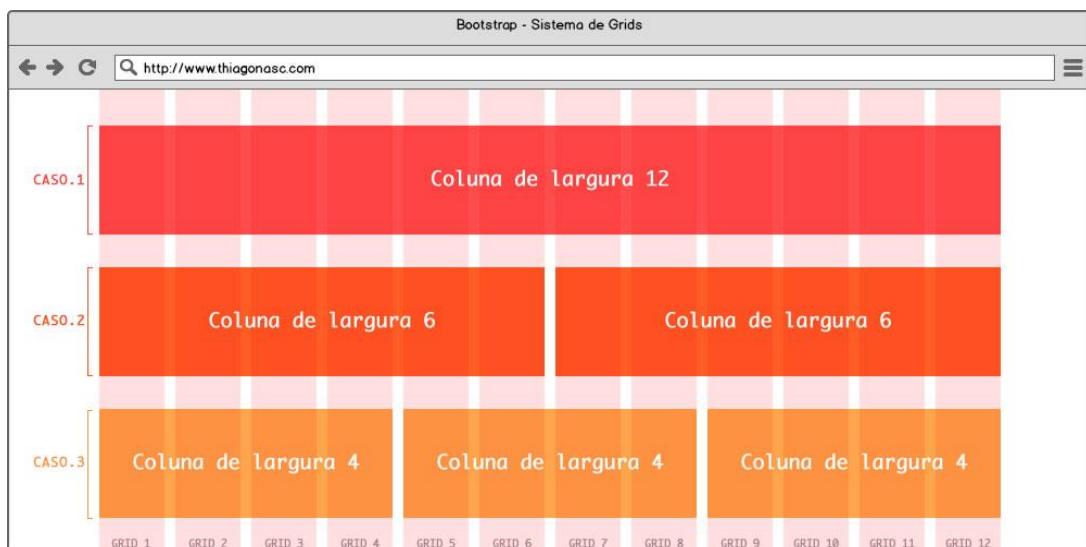
O uso dos grids é fundamental para um projeto com o Bootstrap. O sistema de grids possibilita a divisão em até 12 colunas de larguras iguais, como mostra a figura 13.



**Figura 13 - Sistema de Grids Bootstrap**

Fonte: <http://thiagonasc.com/desenvolvimento-web/desenvolvendo-com-bootstrap-3-um-framework-front-end-que-vale-a-pena>

O uso de grids possibilita mudar o visual de um sistema de maneira fácil e rápida, alterando apenas o valor da largura das colunas, como evidencia a figura 14.



**Figura 14 - Exemplo de Sistema de Grids Bootstrap**

Fonte: <http://thiagonasc.com/desenvolvimento-web/desenvolvendo-com-bootstrap-3-um-framework-front-end-que-vale-a-pena>

## 2.17 ANGULARJS

É um *framework JavaScript open-source*, mantido pelo Google, que auxilia na execução de *single-page applications* (aplicações de uma única página). Seu objetivo é aumentar aplicativos que podem ser acessados por um navegador web, sob o padrão *model-view-controller* (MVC), em um esforço para facilitar tanto o desenvolvimento quanto o teste dos aplicativos.

A biblioteca lê o HTML que contém *tags* especiais e então executa a diretiva na qual esta *tag* pertence, e faz a ligação entre a apresentação e seu modelo, representado por variáveis *JavaScript* comuns. O *framework* adapta e estende o HTML tradicional para uma melhor experiência com conteúdo dinâmico, com ligação direta e bidirecional dos dados que permite sincronização automática de *models* e *views*.

## 2.18 METODOLOGIA

Aqui serão descritas as principais etapas da metodologia para a realização da análise e desenvolvimento do projeto. Antes de iniciar a análise do sistema, foi estudado o funcionamento de uma academia com especialistas na área. Nesses estudos foi possível conhecer os aspectos relacionados a uma academia, tais como o ambiente para os exercícios, equipamentos utilizados e o gerenciamento da academia num todo.

No início, foram necessárias algumas visitas a algumas academias existentes para sanar dúvidas referentes ao projeto e verificar possíveis necessidades de gerenciamento a serem atendidas pelo projeto. Essas academias possuíam profissionais formados atuando na assistência aos alunos e os proprietários também eram profissionais da área, facilitando assim a coleta de dados.

Considerando que o projeto do sistema é amplo, houve necessidade de definir uma primeira visão geral do sistema, mas os seus requisitos serão complementados e revistos à medida que o projeto tiver andamento. Apesar de em uma primeira iteração terem sido levantados e definidos todos os requisitos considerados essenciais, várias revisões irão ocorrer à medida que o projeto evoluir e as reuniões com os interessados para definir escopos acontecerem.

Para auxiliar a coleta de dados, alguns questionamentos foram feitos. O questionário foi aplicado a um profissional experiente no ramo. Com isso, foi possível conhecer os procedimentos de uma academia, quais os cuidados existentes com os frequentadores, principais necessidades e preocupações e as ações tomadas em diferentes situações que acontecem dentro de uma academia. Logo abaixo, estão alguns dos vários questionamentos feitos ao profissional.

<b>Pergunta</b>	<b>Resposta</b>
Quantos funcionários possuem atualmente na academia?	Quatro funcionários. Um é atendente e os outros três são instrutores.
Quantos alunos possui matriculados na academia?	320 alunos.
Quais os valores pagos por aluno?	Depende da frequência semanal e da atividade que o mesmo frequenta. A atividade pode ser aula de dança, artes marciais, musculação, etc.
São efetuadas vendas de outros produtos? Há controle de entrada e saída desses produtos?	Sim. Há controle, mas é manualmente. Geralmente é anotado no papel o que foi vendido e os respectivos valores.
E quanto aos cadastros dos alunos, como é feito?	Existem fichas individuais, onde são preenchidas com os dados pessoais no início, quando o aluno chega até a academia para começar as aulas.
Existe algum histórico armazenado quanto a evolução do aluno?	Não existe, são anotadas algumas observações apenas nas fichas quando necessário.
São feitas avaliações físicas periódicas dos frequentadores?	Não são feitas por falta de recursos para armazenar os dados.
Como é feito o controle do pagamento das mensalidades dos alunos?	É anotado nas fichas de cada aluno. É complicado as vezes controlar o pagamento de todos, até mesmo para em caso do não pagamento, o aluno não pode utilizar os serviços que a academia



	oferece.
E o recebimento das mensalidades é feito de que forma?	Na própria academia.
Por que não são feitos boletos bancários?	Falta de relacionamento e convênios com os bancos e falta de sistema para gerenciar a empresa.
É feito algum controle quanto a contas a pagar e receber?	É controlado no papel e caneta, mas sem a devida a atenção. Não é um controle 100% correto.
Há convênios com alguma empresa?	Há com uma empresa, onde os funcionários possuem um valor de desconto nas mensalidades.
E a entrada de pessoas na academia possui algum controle?	Existe uma recepcionista na entrada que procura controlar o acesso de pessoas dentro da academia. Precisa de algo do gênero para melhorar esse controle de acesso.

**Quadro 1 – Questionário**

Para o desenvolvimento do projeto, algumas fases foram necessárias. Inicialmente elaborou-se uma visão geral do sistema proposto com seus principais componentes. Foram levantadas as necessidades administrativas, financeiras e gerenciais das academias.

Nesta fase decidiu-se que o sistema seria modelado com base nos requisitos levantados inicialmente, mas à medida que fossem implementados, os requisitos seriam revistos. Essa modelagem foi feita no trabalho de estágio. Com base nessas decisões foi estabelecido um cronograma para que a documentação do trabalho e a modelagem fossem realizadas simultaneamente. Também se chegou à ideia de que utilizar a metodologia Scrum para o desenvolvimento daria um salto na qualidade e gerenciamento do projeto.

Após isso, veio a fase de construção dos diagramas da UML, com base nas necessidades e interesses levantados em entrevistas e visitas feitas aos clientes. As principais tarefas desta fase são:

- Definir os casos de uso para o sistema;

- Definir o diagrama de classes, com seus atributos, operações e relacionamentos;
- Definir os requisitos funcionais e os requisitos não funcionais dos casos de uso;
- Desenvolver o projeto do banco de dados, com a definição das tabelas, campos, chaves e os relacionamentos;
- Gerar os *scripts* para criar o banco para PostgreSQL 9.3.

Depois, foi feita a preparação do ambiente para a implementação do projeto. A definição das tecnologias que seriam utilizadas foi realizada na fase de planejamento. As tecnologias foram preparadas, instaladas e configuradas para que a implementação ocorresse.

A implementação do sistema será realizada em partes. Inicialmente serão feitos os cadastros de clientes, funcionários e atividades. Juntando a isso, ainda será feito a matrícula do aluno nas atividades que desejar no próprio cadastro do cliente. O capítulo 4 tem um exemplo de codificação e também algumas das telas que o sistema possui, afim de mostrar um esboço de como ficaria o projeto.

Vale ressaltar também que como foi utilizada a metodologia do Scrum para o projeto, várias conversas ocorreram com o futuro cliente. Como é um projeto em que o autor está trabalhando sozinho, pelo menos agora no início, não foi necessário e nem foi possível dividir as tarefas, de acordo com as regras do Scrum, mas uma pessoa pode fazer todas as funções, sem a necessidade de outras. Isso seria possível se tivesse um Time de Desenvolvimento, o *Product Owner* e o Scrum Master.

A primeira conversa ou reunião que ocorreu entre as partes interessadas, ou seja, os potenciais clientes e o autor desse projeto foram ainda no início quando apenas tinha-se um esboço de ideia de software para gerenciamento de academia. Nesse dia, foram levantados alguns dados relevantes da academia, como o funcionamento, a ordem dos processos feita dentro da academia, as modalidades de atividades físicas oferecida aos clientes, etc. Com isso, descobriu-se várias necessidades que os potenciais clientes tinham ficando mas fácil de escrever uma visão geral do sistema.

Após a construção da visão geral do sistema bem detalhada e divisão dos possíveis casos de uso, o autor voltou a ter uma conversa com um dos interessados, já que as necessidades dos envolvidos eram semelhantes. Foi mostrado o levantamento feito do projeto para verificar se estava dentro do pretendido. Claro que houve alguns pequenos ajustes, mas estava basicamente tudo certo, sendo que com isso, foi passado para o próximo passo.

O próximo passo coube ao responsável pela elaboração do projeto, pois com a visão geral e os casos de usos aprovados, é necessário construir outros diagramas e tabelas. Os primeiros a serem feitos foram os requisitos funcionais e os requisitos não funcionais e também o detalhamento dos casos de uso. Aproveitando isso, foi elaborado os requisitos suplementares, os conceitos, as consultas e relatórios que o sistema viria a ter e o Diagrama de Classes.

Depois de concluído a parte de análise e modelagem do projeto, com as tecnologias que seriam utilizadas já definidas, partiu-se para o início da programação do projeto. Foi optado por fazer de início alguns dos cadastros. Procurou-se seguir a metodologia no Scrum, sendo que a cada passo importante dado era feita uma avaliação do que foi feito juntamente com o cliente para ver se ficou de acordo com o que ele precisava, sendo alterado ou modificado quando necessário. Como a parte de cadastros não possui muitas diferenças para se implementar, mudando apenas alguns atributos, as dificuldades encontradas não foram tantas.

Nesse relatório não será mostrado todo sistema já implementado e pronto, pela questão do tempo escasso, mas já há um esboço de qual é a real ideia do projeto.

### **3 MODELAGEM E IMPLEMENTAÇÃO DO SISTEMA**

Este capítulo apresenta a modelagem e a implementação do sistema para gerenciamento de uma academia, incluindo o controle das avaliações físicas, das atividades realizadas e também dos produtos consumidos pelos clientes da academia.

#### **3.1 SISTEMA**

O sistema contará com cadastro de clientes e colaboradores, permitirá o controle de convênios com empresas às quais poderão ser ofertados planos para as diferentes atividades oferecidas pela academia. Será criado um cadastro de atividades oferecidas (musculação, pilates, danças e outros), assim como um plano de acompanhamento de peso e massa corporal. O sistema possibilitará acompanhamento de avaliação física completa com resultados apresentados em relatórios e gráficos para a melhor compreensão dos clientes, assim como a criação de turmas para cursos ou outras atividades conjuntas.

Como funcionalidades adicionais, cita-se que será desenvolvido um controle financeiro levando em consideração os convênios com as empresas, assim como, um plano de pagamento (mensal, trimestral ou de acordo com o convênio). Serão emitidos boletos bancários (enviados por e-mail) de acordo com a necessidade do cliente e serão aceitos pagamentos via transferência bancária e cartão de crédito. Haverá opção de cobrança de multas ou bloqueios para alunos inadimplentes.

O sistema efetuará o controle de cheques recebidos/emitidos, de produtos vendidos pela academia e dívidas a pagar pela mesma, gerará relatórios do sistema financeiro, de acordo com o número de alunos que realizam as atividades, assim como os pagamentos em dias e as dívidas pendentes e pagas pela academia.

Com relação às matrículas, ao ser efetuado o cadastro é escolhido o plano de atividades que o aluno realizará na academia. Nesse plano é definida a forma de pagamento e o aluno recebe um login e senha para o acesso ao site da academia. Nesse site o mesmo terá a possibilidade de verificar a sua situação do pagamento da mensalidade ou atividades efetuadas e acesso ao seu plano de acompanhamento caso realize alguma atividade física ou atividades

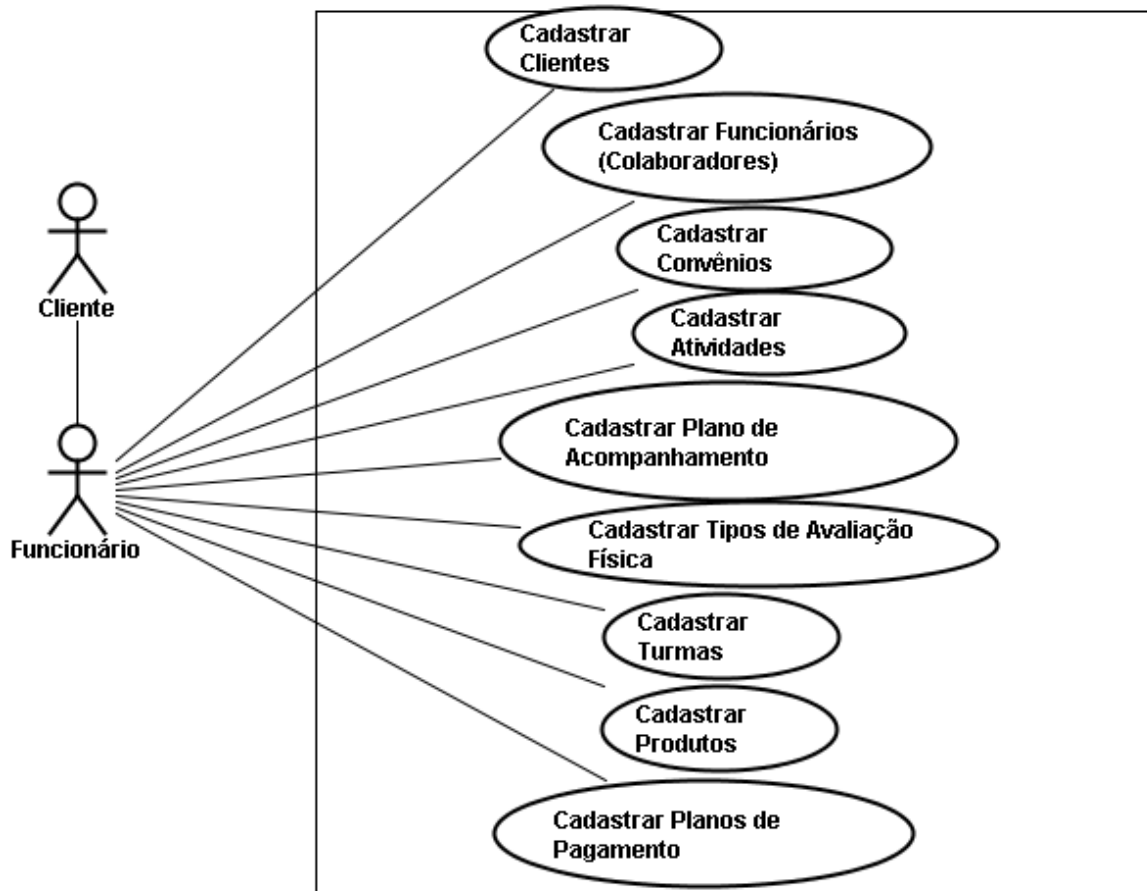
complementares. O cliente também receberá e-mail contendo novidades oferecidas pela academia.

O sistema terá uma agenda de atividades de acordo com o cadastro do funcionário. Cada funcionário receberá permissões de acesso assim como de alteração no sistema.

### 3.2 CASOS DE USO

As Figuras 15 e 16 apresentam os casos de uso referentes aos cadastros e controles aos quais o sistema será responsável.

A Figura 15 apresenta os casos de uso referente aos cadastros.



**Figura 15 - Diagrama de Casos de Uso dos Cadastros**

Fonte: Autoria Própria

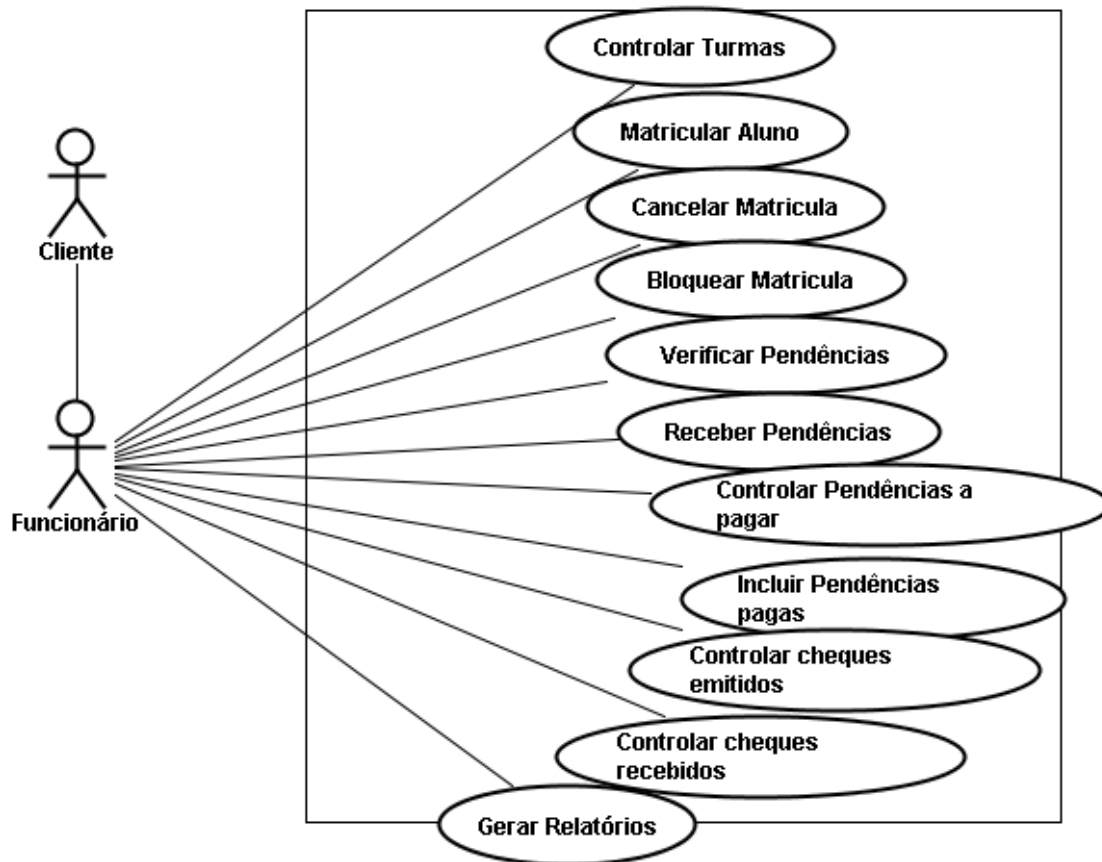
Os casos de uso dos cadastros são realizados pelos atores cliente e funcionário. O funcionário responsável pela função coleta os dados necessários do

cliente para o cadastro e lança no sistema para que possa ser feita a matrícula do mesmo em uma das atividades que a academia disponibiliza aos alunos.

<b>Nome</b>	<b>Atores</b>	<b>Descrição</b>	<b>Referências Cruzadas</b>
Cadastrar Clientes	Cliente, Funcionário	O sistema deve efetuar o cadastro de clientes. O cliente é identificado através de um CPF válido.	F1
Cadastrar Funcionários	Funcionário	O sistema deve efetuar o cadastro de funcionários.	F2
Cadastrar Convênios (Clientes)	Cliente, Funcionário	O sistema irá cadastrar convênios previamente estabelecidos com outras empresas, instituições e outros.	F3, F2
Cadastrar Atividades	Administrador	O sistema deve registrar os tipos de atividades oferecidas pela academia.	F4
Cadastrar Plano de acompanhamento	Funcionário	O sistema deve efetuar o cadastro dos tipos de planos oferecidos.	F5
Cadastrar Tipos de Avaliação física	Funcionário	O sistema deve efetuar o cadastro dos tipos de avaliações físicas disponíveis para os clientes.	F6
Cadastrar Turmas	Funcionário	O sistema deve efetuar o cadastro de turmas para cursos conjuntos.	F1, F2, F3, F7
Cadastrar Produtos	Funcionário	O sistema deve efetuar o cadastro de produtos oferecidos aos clientes.	F9
Cadastrar Planos de pagamento	Funcionário	O sistema deve cadastrar os diferentes planos oferecidos.	F10

**Quadro 2 - Casos de uso dos cadastros**

A Figura 16 apresenta os casos de uso referentes aos controles diversos do sistema.



**Figura 16 - Diagrama de Casos de Uso dos Demais Controles**

Fonte: Autoria Própria

Os casos de uso para os demais controles são realizados pelo ator funcionário. Todas as funcionalidades do sistema são acessadas e realizadas a partir de um *login*. Só pode acessar o sistema os funcionários cadastrados e que informem um *login* e senha corretos. Alguns controles como o financeiro e o bloqueio e cancelamento de matrículas pode ser feito apenas pelos administradores, ou com autorização do mesmo.

Nome	Atores	Descrição	Referências Cruzadas
Controlar turmas	Funcionário, Administrador	O sistema deve efetuar o controle de turmas por atividades.	F4, F7, F18
Matricular Aluno	Funcionário	O sistema deve efetuar a matrícula do aluno, informando o	F1, F3, F4, F5, F6, F7, F10

		plano de atividades desejado, assim como se possui algum convênio.	
Cancelar Matrícula	Funcionário, Cliente	O sistema deve permitir ao aluno o cancelamento de sua matrícula.	F10, F11
Bloquear Matrícula	Funcionário	O sistema irá bloquear a matrícula em caso de atraso de pagamento e só liberando quando pago o débito com a devida multa calculada.	F10, F12
Verificar Pendências	Funcionário	O sistema efetua a verificação de pendências a serem recebidas.	F17
Receber Pendências	Funcionário, Administrador	O sistema deve efetuar o recebimento das pendências.	F11, F12
Controlar Pendências à pagar	Administrador	O sistema efetua o controle das pendências a serem pagas.	F13
Incluir pendências pagas	Administrador	O sistema efetua a inclusão das pendências que foram pagas.	F13, F14
Controlar Cheques emitidos	Administrador	O sistema efetua o controle de cheques que foram emitidos.	F14, F15
Controlar Cheques recebidos	Administrador, Funcionário	O sistema efetua o controle de cheques que foram recebidos.	F12, F16
Gerar Relatórios	Administrador, Funcionário	O sistema irá gerar relatório resgatando informações previamente cadastradas no sistema.	F5, F11, F12, F13, F17

**Quadro 3 - Casos de uso dos demais controles**



### 3.3 CASOS DE USO COMPLETO

Este capítulo apresenta o detalhamento dos casos de uso levantados para o sistema.

#### 3.3.1 Níveis de Detalhamento de um Caso de Uso

Nos quadros 4 e 5 a seguir, são apresentados os detalhamentos de algumas das funcionalidades mais complexas e importantes do sistema, lembrando que foram feitas as expansões de todos os casos de uso para a modelagem, as quais são apresentadas no Apêndice A.

<b>Caso de Uso:</b> Controlar turmas
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema deve efetuar o controle de turmas por atividades.
<b>Pós-condições:</b> O sistema informa se há vagas na respectiva turma informada.
<b>Requisitos Correlacionados:</b> F7
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> <ol style="list-style-type: none"> <li>1. O cliente chega à academia para fazer seu cadastro e sua matrícula em umas das atividades existentes na academia.</li> <li>2. O funcionário verifica se há vagas na turma escolhida pelo cliente.</li> <li>3. [IN] O funcionário insere no sistema os dados do aluno e a atividade escolhida pelo mesmo.</li> <li>4. [OUT] O funcionário finaliza o cadastro.</li> </ol>
<b>Tratamento de Exceções:</b> <b>2a Não existe vaga para o horário que o aluno tem preferência</b> <b>2a.1[OUT]</b> O sistema informa que não existe vagas para aquele horário <b>2a.2</b> Retorna ao item 2 do fluxo principal

Quadro 4 - Caso de uso detalhado Controlar turmas

<b>Caso de Uso:</b> Verificar Pendências
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema efetua a verificação de pendências a serem recebidas.
<b>Pós-condições:</b> O sistema informa através de uma mensagem ao usuário as pendências existentes

<b>Requisitos Correlacionados:</b> F11
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> <ol style="list-style-type: none"> <li>1. O funcionário verifica as pendências existentes no sistema.</li> <li>2. É executada uma cobrança aos respectivos devedores.</li> </ol>
<b>Tratamento de Exceções:</b> <b>2a Quando é feito a cobrança e a dívida é renegociada</b> <b>2a.1</b> Verifica com o gerente ou proprietário se há possibilidade de fazer algo diferenciado. <b>2a.2</b> Retorna ao item 2 do fluxo principal.

**Quadro 5 - Caso de uso detalhado Verificar pendências**

### 3.4 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

O primeiro passo realizado na modelagem foi o levantamento dos requisitos funcionais e não funcionais do sistema, que representam as funcionalidades a serem implementadas pelo sistema. Foi utilizada a letra “F” juntamente com um número sequencial como padrão para identificação, para simbolizar que se trata de um requisito funcional.

Os quadros 6 e 7 listam os requisitos funcionais considerados mais importantes na modelagem e seus respectivos requisitos não funcionais, sendo que no Apêndice B são apresentados os demais requisitos funcionais e não funcionais.

<b>F8</b> Controlar turmas	<b>Oculto ( )</b>			
<b>Descrição:</b> O sistema deve efetuar o controle de turmas por atividades. Verificar se ainda possui vagas em alguma turma que o aluno esteja interessado em matricular-se.				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF8.1 Identificação da Turma	O sistema irá fazer a identificação da turma por meio de um código previamente cadastrado.	Usabilidade	( X )	( X )
NF8.2 Interface	O controle de turmas deve ser feito utilizando somente uma tela.	Desempenho	( )	( X )

NF8.3 Controle de vagas	Cada turma terá um limite de alunos previamente estipulado.	Controle	( X )	( )
-------------------------------	--	----------	-------	-----

**Quadro 6 - Requisito Controlar turmas**

<b>F11 Verificar pendências</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema efetua a verificação de pendências a serem recebidas.				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF11.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade.	Segurança	( X )	( X )
NF11.2 Verificação de pendências (Receber)	Verifica através do código da matrícula do aluno as pendências do mesmo se houver.	Usabilidade	( )	( X )

**Quadro 7 - Requisito Verificar pendências**

### 3.5 DIAGRAMA DE CLASSES

Os diagramas de classes são utilizados para fazer a modelagem da visão estática de um sistema. Assim oferece suporte para os requisitos funcionais da modelagem de um sistema.

Este diagrama possui as classes responsáveis por parte dos cadastros e armazenamento de dados para o gerenciamento do sistema, sendo utilizada a ferramenta Astah Community para sua devida construção. A Figura 17 apresenta o diagrama de classes da modelagem do sistema proposto.

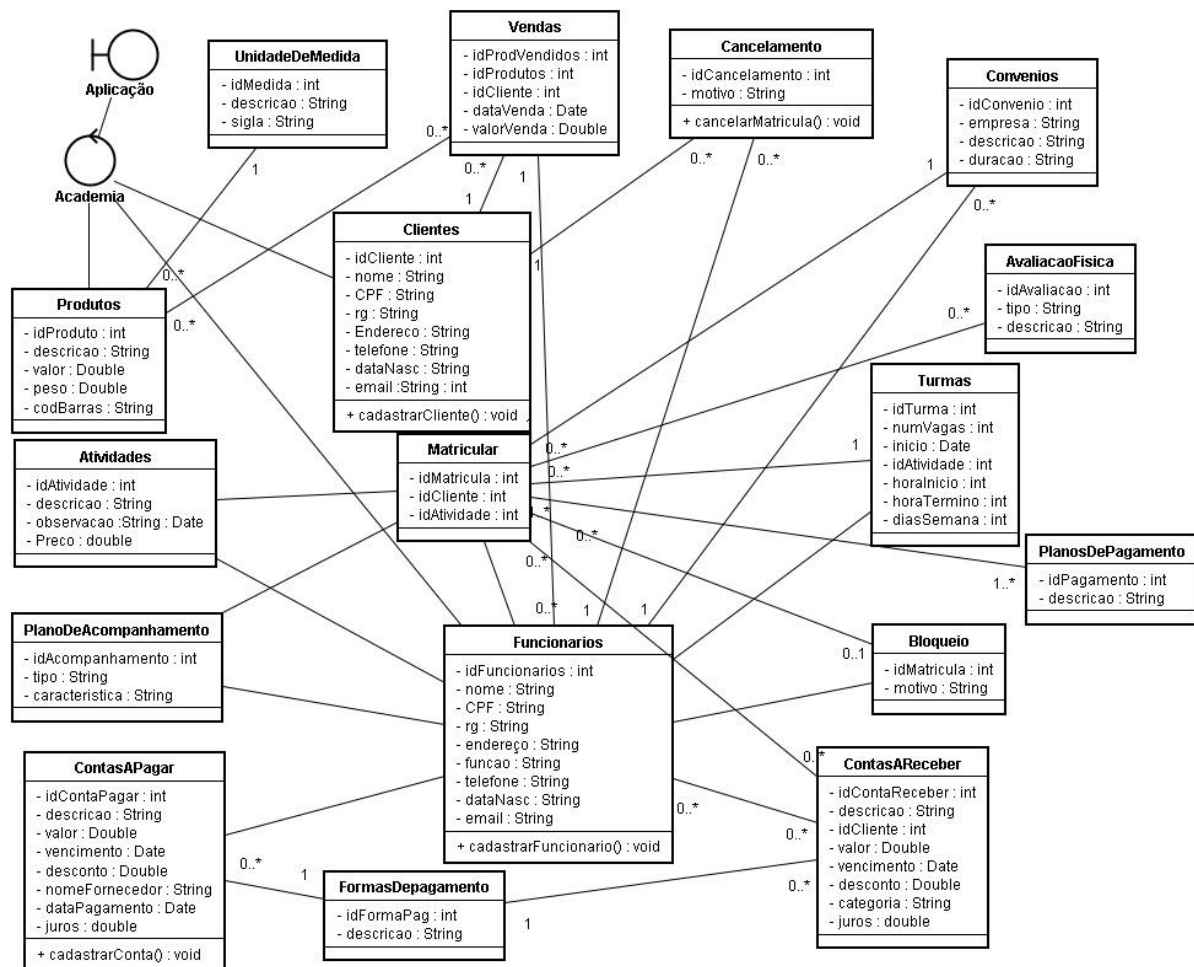


Figura 17 - Diagrama de Classes

Fonte: Autoria Própria

### 3.6 REQUISITOS DE CONCEITOS

Após os requisitos do sistema serem identificados foi possível identificar o conjunto de conceitos do sistema. No Quadro 8 são apresentados os conceitos, sendo que as letras “I”, “A”, “E” e “C” significam inclusão, alteração, exclusão e consulta, respectivamente. A coluna observação apresenta as restrições (exceções) para que não ocorram inconsistências nas funcionalidades descritas. Na coluna referências cruzadas estão as referências entre as funcionalidades e caso haja algumas alterações em algumas delas será possível identificar as afetadas, podendo verificar se problemas foram gerados.

<b>Conceito</b>	<b>I</b>	<b>A</b>	<b>E</b>	<b>C</b>	<b>Observação</b>	<b>Referências Cruzadas</b>
Cliente	X	X	X	X	Os clientes já cadastrados só podem ser excluídos do sistema após certo período de inatividade, sendo esse período configurável.	F1
Funcionário	X	X	X	X	Os funcionários devem seguir uma agenda definida anteriormente.	F2
Convênios	X	X	X	X	Um convênio pode ser alterado, consultado ou excluído a hora que for necessário, sendo feita automaticamente a correção da mensalidade.	F3
Atividades	X	X	X	X	Novas atividades podem ser incluídas e outras já existentes podem ser modificadas ou consultadas.	F4
Planos de acompanhamento	X	X	X	X	O plano de acompanhamento não pode ser excluído, apenas alterado.	F5
Avaliações físicas	X	X		X	As avaliações físicas podem ser alteradas ou consultadas.	F6
Turmas	X	X	X	X	As turmas já fechadas não podem ser alteradas.	F7
Produtos	X	X	X	X	Os produtos não podem ser excluídos, apenas alterados.	F9
Planos de pagamento	X	X		X	Não podem ser excluídos, apenas alterados.	F10
Matrícula	X	X	X	X	Uma matrícula não pode ser excluída se houverem pendências financeiras.	F17
Cancelamento	X			X	Deve ser feito uma consulta se não há nenhum tipo de débito pendente com a academia	F11, F18

					antes de um cancelamento.	
Pagamentos	X			X	Um pagamento não pode ser alterado ou excluído.	F13, F14, F15
Recebimentos	X			X	Um recebimento não pode ser alterado ou excluído.	F12, F16
Bloqueios	X	X	X	X	Um bloqueio só deve ser liberado após não haver mais pendências.	F19

**Quadro 8 - Requisitos de conceitos**

### 3.7 REQUISITOS SUPLEMENTARES

Os requisitos suplementares são todo tipo de restrição tecnológica ou lógica que se aplica ao sistema como um todo e não apenas a funções individuais.

No Quadro 9 estão os requisitos suplementares encontrados na modelagem desse sistema.

Nome	Restrição	Categoria	Desejável	Permanente
S1 Interface gráfica do sistema	O sistema deve possuir uma interface amigável, fácil e rápida para facilitar e agilizar o tempo do serviço.	Interface	(X)	( )
S2 Tipos de banco de dados	Deve ser maleável de tal forma que se houver necessidade de troca de sistema no futuro, possa aproveitar o mesmo banco.	Persistência	(X)	( )
S3 Permissões dos usuários	Os usuários devem ter acesso restrito ao sistema, para acesso a funcionalidades mais restritas necessita-se de autorização do administrador.	Segurança	( )	(X)
S4 Restrições de cadastro	O sistema não deve aceitar o cadastro faltando dados obrigatórios.	Segurança	( )	(X)

S5 Restrição de matrícula	O sistema não deve aceitar matrícula se o aluno possuir algum débito.	Segurança	( )	(X)
S6 Restrição de cadastro	O sistema não deve permitir o cadastro do aluno em atividades que ocorrem simultaneamente.	Usabilidade	(X)	(X)
S7 Restrição de turmas	O sistema não deve permitir que haja duas turmas realizando a mesma atividade simultaneamente.	Usabilidade	(X)	(X)

**Quadro 9 - Requisitos Suplementares**

### 3.8 CONSULTAS / RELATÓRIOS

Outro artefato importante da modelagem são os relatórios que podem ser gerados pelo sistema. Possuem menor complexidade sendo consideradas simples, pois apenas consultam dados já armazenados no sistema, baixando o risco de conter erros. O processo unificado orienta que estas funcionalidades sejam implementadas por apresentarem baixo risco de problemas ao projeto.

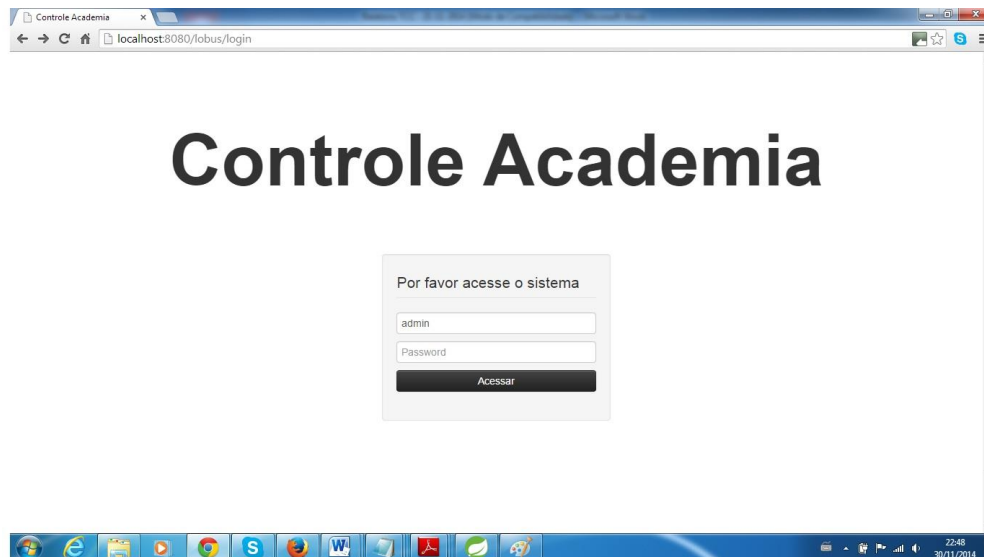
O Quadro 10 apresenta os relatórios a serem gerados pelo sistema juntamente com os requisitos que serão referências.

Nome	Referências Cruzadas
Relatório de alunos ativos	F1, F17
Relatório de pagamentos (alunos)	F1, F12, F16
Relatório de alunos matriculados	F11, F17
Relatório de acompanhamento físico.	F6, F17
Relatório de contas a receber	F11, F16, F17
Relatório de contas a pagar	F13, F15, F17
Relatório de alunos pendentes	F1, F11, F17
Relatório de pagamentos/recebimentos (academia)	F12, F14, F15, F16, F17

**Quadro 10 - Consultas / Relatórios**

### 3.9 APRESENTAÇÃO DO SISTEMA

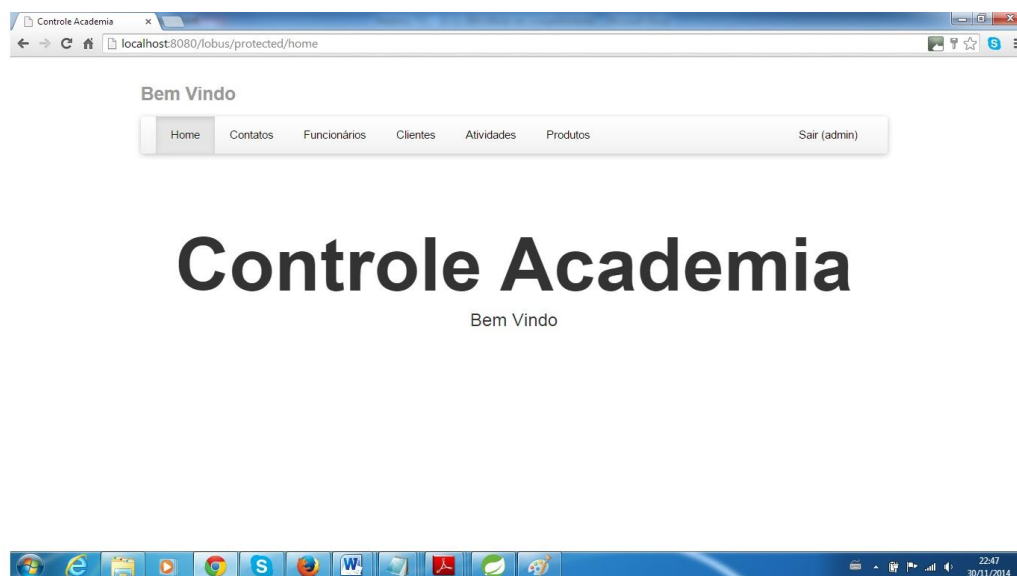
A figura 18 apresenta a tela de acesso do sistema, que é a primeira tela que o usuário visualizará ao acessar o endereço referente à aplicação. Nessa tela é solicitado o e-mail do usuário e senha previamente cadastrados.



**Figura 18 - Tela de Login do Sistema**

Fonte: Autoria própria

Se os dados informados na tela de acesso ao sistema estiverem corretos, o usuário será direcionado a tela principal do sistema, como mostra a figura 19.



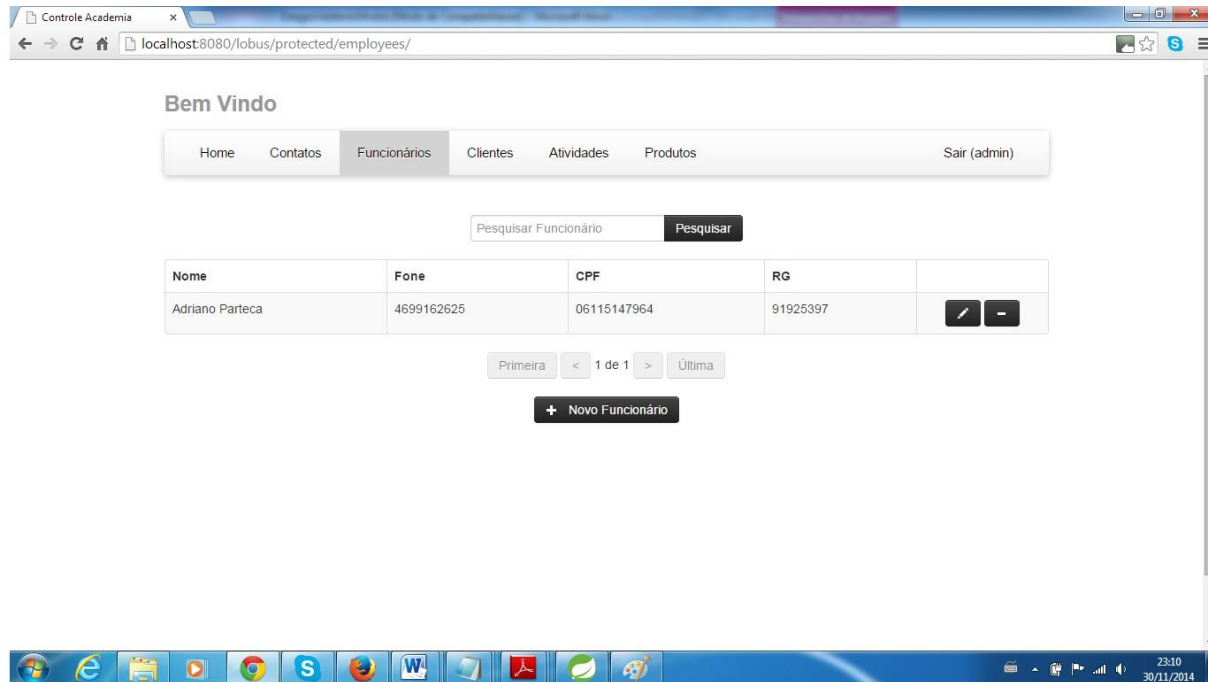
**Figura 19 - Tela Principal do Sistema**

Fonte: Autoria própria



Na figura 19 está a tela principal do sistema. Nessa tela, o usuário é capaz de visualizar na parte superior o menu por meio do qual terá acesso as telas de cadastros de funcionários, clientes e atividades.

A figura 20 apresenta a tela do cadastro de funcionários.



**Figura 20 - Tela de Cadastro de Funcionários**

Fonte: Autoria própria

A figura acima mostra que é possível serem cadastrados vários funcionários para utilizarem o sistema. Para cadastrar um novo usuário basta clicar no botão “Novo Funcionário”. Há ainda a opção para editar ou apagar um funcionário cadastrado. Também há um campo de pesquisa para facilitar, caso seja necessário.

A figura 21 mostra os campos que aparecem para serem preenchidos após clicar no botão para adicionar um funcionário. Nota-se que os campos são todos obrigatórios, para ter-se um melhor controle cadastral dos funcionários existentes.

A figura 22 apresenta a tela do cadastro de clientes. Seu *layout* é semelhante à tela de cadastro de funcionários. O que mudará apenas serão os campos a serem preenchidos quando quiser cadastrar um novo cliente, como mostra a figura 23. Além disso, há possibilidade de matricular o aluno na atividade que deseja na hora que fizer o cadastro.

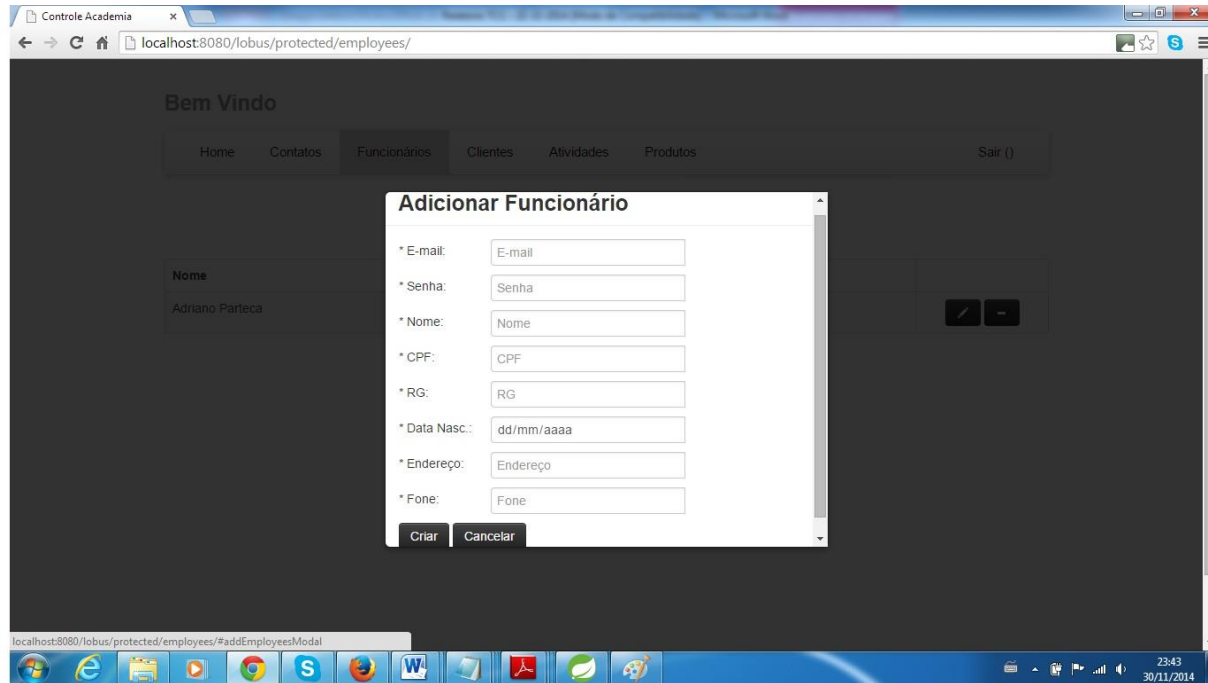


Figura 21 - Tela de Inclusão de Funcionários

Fonte: Autoria própria

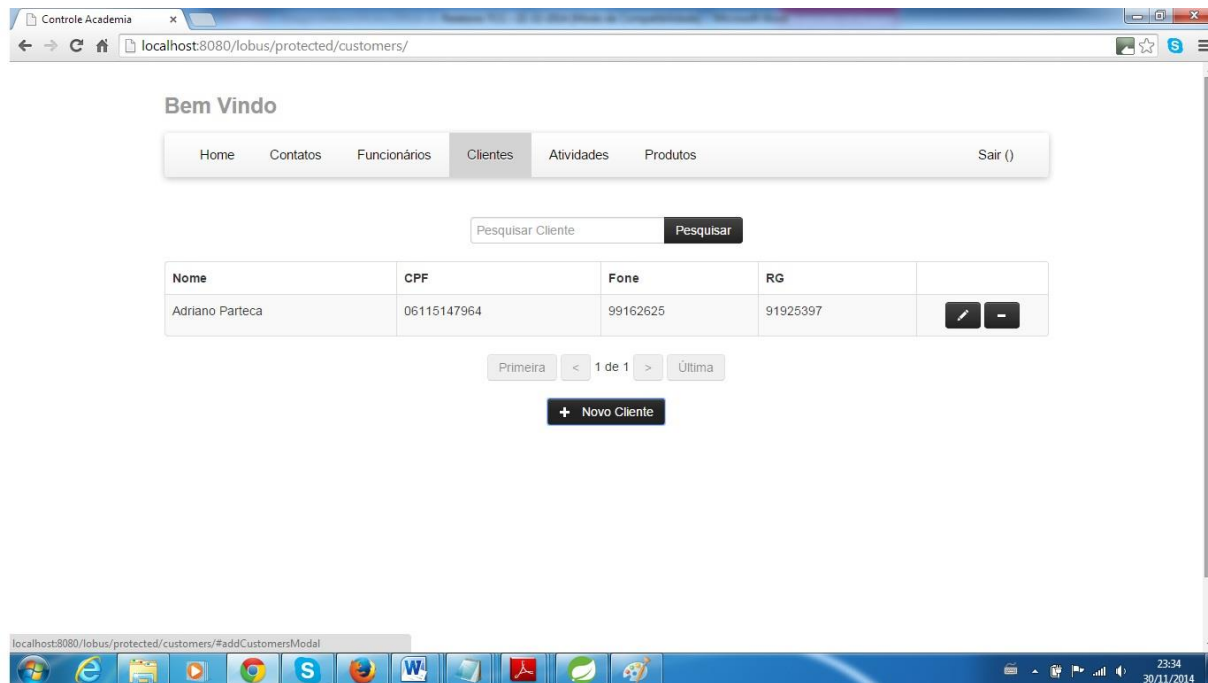


Figura 22 - Tela de Cadastro de Clientes

Fonte: Autoria própria

The screenshot shows a web browser window with the URL `localhost:8080/lobus/protected/customers/`. The page title is "Controle Academia". A modal window titled "Adicionar Cliente" is open, containing the following fields:

- \* Nome:
- CPF:
- RG:
- Data Nasc.:
- E-mail:
- Endereço:
- Fone:
- Atividade:

Buttons "Criar" and "Cancelar" are located at the bottom of the modal. The background shows a sidebar with "Home" and "Contatos" and a main area with "Sair ()".

**Figura 23 - Tela de Inclusão de clientes**

Fonte: Autoria própria

Agora partiremos para o módulo financeiro. A figura 24 indica a tela do cadastro de contas a receber. Já a figura 25 mostra a tela referente ao cadastro de contas a pagar.

The screenshot shows a web application window with the title "Novo Recebimento". The form contains the following fields:

- Descrição:
- Categoria:
- Data vencimento:
- Valor:
- Cliente:
- Data recebimento:
- Descontos / Taxas:
- Juros / Multa:
- Valor recebido:

Buttons "Criar" and "Cancelar" are located at the bottom of the form.

**Figura 24 - Tela de Cadastro de Contas a Receber**

Fonte: Autoria própria

**Novo Pagamento**

Descrição:

Categoria:

Data vencimento:

Valor:

Fornecedor:

---

Data Pagamento:

Descontos / Taxas:

Juros / Multa:

Valor pago:

**Figura 25 - Tela de Cadastro de Contas a Pagar**

Fonte: Autoria própria

### 3.10 IMPLEMENTAÇÃO DO SISTEMA

Neste item, está exemplificada a implementação do sistema. Para o desenvolvimento do projeto foi utilizada a ferramenta de desenvolvimento *Spring Tool Suite*.

No arquivo “context.xml” é possível encontrar uma configuração simples de *datasource* que será referenciado pelo Spring. Essa fonte de dados já fará *pool* de conexões e também validará a conexão para que não existam conexões inativas no *pool* o que ocasionaria erros. A figura 26 mostra o código do “context.xml”.

A figura 27 mostra o código responsável por fazer a pesquisa no banco de dados através do nome. A figura 28 também mostra uma forma de consulta ao banco de dados.



```

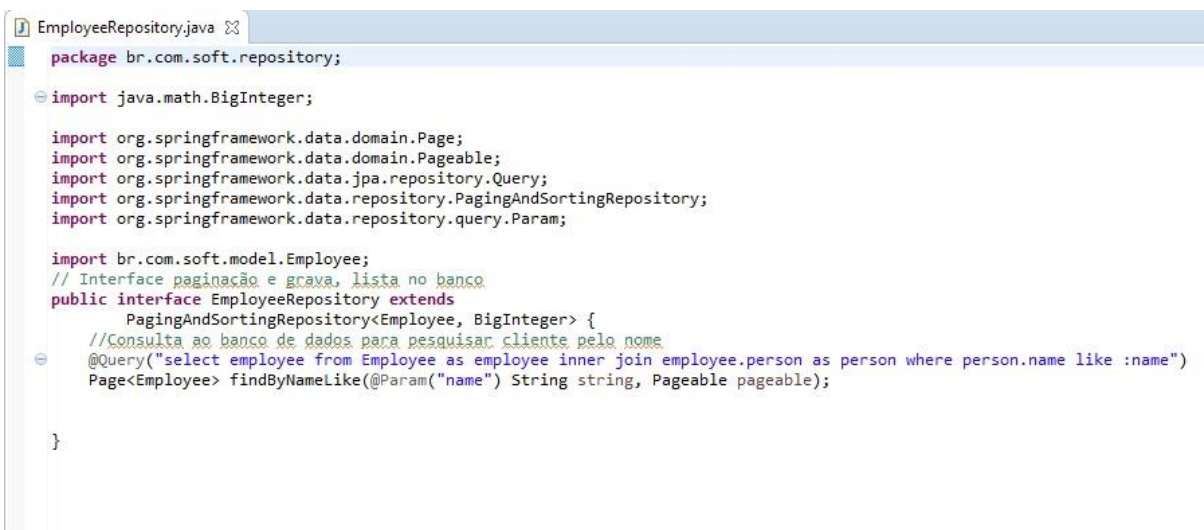
context.xml
<?xml version='1.0' encoding='utf-8'?>
<Context>
  <WatchedResource>WEB-INF/web.xml</WatchedResource>

  <Resource
    factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
    name="jdbc/tomcatDataSource"
    auth="Container"
    type="javax.sql.DataSource"
    initialSize="1"
    maxActive="20"
    maxIdle="3"
    minIdle="1"
    maxWait="5000"
    username="postgres"
    password="123"
    driverClassName="org.postgresql.Driver"
    validationQuery="SELECT 'OK'"
    testWhileIdle="true"
    testOnBorrow="true"
    numTestsPerEvictionRun="5"
    timeBetweenEvictionRunsMillis="30000"
    minEvictableIdleTimeMillis="60000"
    url="jdbc:postgresql://localhost:5432/academia" />
</Context>

```

**Figura 26 - Arquivo Context.xml**

Fonte: Autoria Própria



```

EmployeeRepository.java
package br.com.soft.repository;

import java.math.BigInteger;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;

import br.com.soft.model.Employee;
// Interface paginação e grava, lista no banco
public interface EmployeeRepository extends
    PagingAndSortingRepository<Employee, BigInteger> {
    //Consulta ao banco de dados para pesquisar cliente pelo nome
    @Query("select employee from Employee as employee inner join employee.person as person where person.name like :name")
    Page<Employee> findByNameLike(@Param("name") String string, Pageable pageable);
}

```

**Figura 27 - Consulta ao Banco de Dados Pelo Nome do Cliente**

Fonte: Autoria Própria

```

CustomerRepository.java
package br.com.soft.repository;

import java.math.BigInteger;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;

import br.com.soft.model.Customer;

public interface CustomerRepository extends
    PagingAndSortingRepository<Customer, BigInteger> {

    @Query("select customer from Customer as customer inner join customer.person as person where person.name like :name")
    Page<Customer> findByName(@Param("name") String name ,Pageable pageable);
}

```

**Figura 28 - Consulta ao Banco de Dados**

Fonte: Autoria Própria

A figura 29 e 30 mostram o código de configuração do Spring Data.

```

context.xml  LoginController.java  ActivityRepository.java  UserRepository.java
package br.com.soft.repository;

import org.springframework.data.repository.CrudRepository;

import br.com.soft.model.User;

public interface UserRepository extends CrudRepository<User, Integer> {
    User findByEmail(String email);
}

```

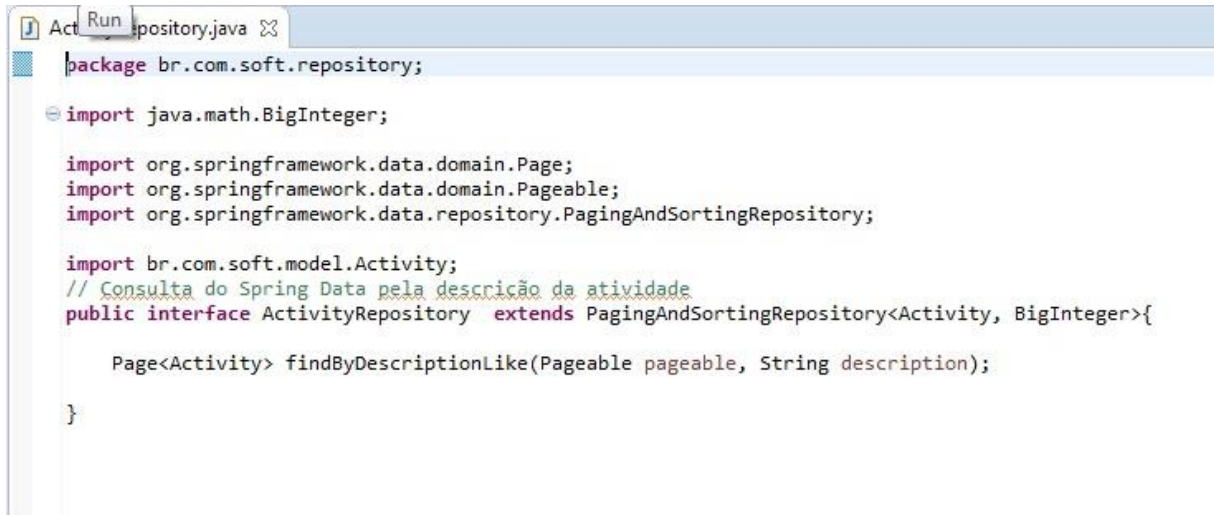
**Figura 29 - Consulta Spring Data**

Fonte: Autoria Própria

A ideia do Spring Data é que o usuário crie a interface e o Spring Data se encarregue do resto. A figura acima mostra que após o *login* do usuário, realizado pelo Spring Security, existe um interceptador que fará a consulta dos dados do usuário pelo *e-mail* no banco de dados. É uma consulta simples para buscar as informações necessárias.

A figura 30 mostra o código do repositório que fará as consultas das atividades cadastradas na academia. A interface *ContactRepository* está herdando de outra interface do Spring, a "*PagingAndSortingRepository*". Essa nova interface

também herda da interface “*CrudRepository*” só que ela conta com uma função a mais, que é a paginação.



```
Act Run repository.java ✕
package br.com.soft.repository;

import java.math.BigInteger;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.repository.PagingAndSortingRepository;

import br.com.soft.model.Activity;
// Consulta do Spring Data pela descrição da atividade
public interface ActivityRepository extends PagingAndSortingRepository<Activity, BigInteger>{

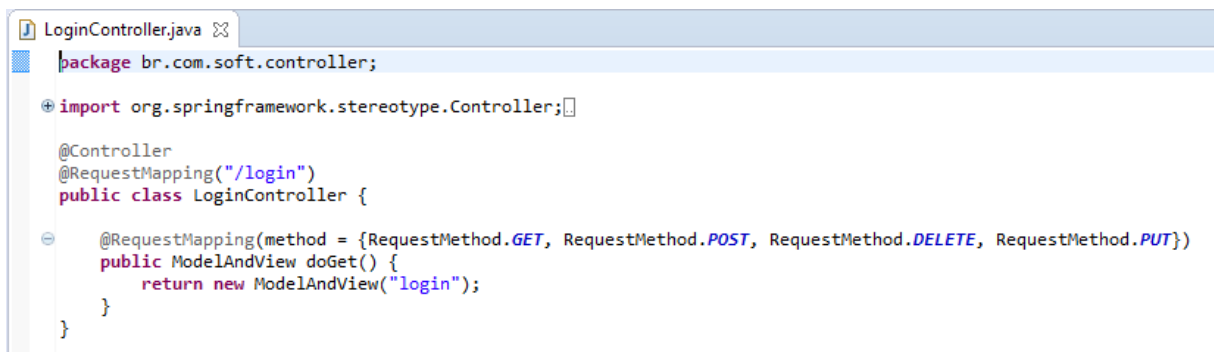
    Page<Activity> findByDescriptionLike(Pageable pageable, String description);

}
```

**Figura 30 - Consulta do Spring Data com Paginação**

Fonte: Autoria Própria

A figura 31 mostra o método chamado quando o *login* é realizado.



```
LoginController.java ✕
package br.com.soft.controller;

import org.springframework.stereotype.Controller;

@Controller
@RequestMapping("/login")
public class LoginController {

    @RequestMapping(method = {RequestMethod.GET, RequestMethod.POST, RequestMethod.DELETE, RequestMethod.PUT})
    public ModelAndView doGet() {
        return new ModelAndView("login");
    }

}
```

**Figura 31 - Método de Login**

Fonte: Autoria própria

## 4 CONCLUSÃO

Neste trabalho foi possível identificar por meio da metodologia do Scrum as principais funcionalidades que um software para gerenciamento de uma academia precisa por meio de um conjunto de diagramas e tabelas que estão dispostos nos princípios da UML. Também foi possível apresentar os métodos para a programação de parte do projeto.

A metodologia Scrum foi de suma importância para que se elaborasse o projeto, sendo que sua metodologia ajudou a alavancar os dados necessários. A principal ferramenta utilizada para auxiliar a análise e a modelagem do sistema foi a ferramenta Astah Community, tendo suma importância durante o processo, pois foi a responsável pela criação dos diagramas de casos de uso e diagrama de classes, facilitando a identificação das classes e tabelas. Com isso, tornou-se mais fácil a identificação de possíveis falhas e exceções no projeto, diminuindo um possível retrabalho e aumento nos custos.

A partir deste projeto, espera-se ser possível a construção de um software que resolverá os problemas levantados na visão geral do sistema, melhorando a qualidade no atendimento dos clientes e inovando para buscar os resultados almejados pelos clientes após começarem a fazer atividades físicas.



## REFERÊNCIAS

APACHE TOMCAT – Disponível em <<http://tomcat.apache.org/>> - Acesso em 08/10/2014.

ASTAH. Disponível em <<http://astah.net>> Acesso em: 03/03/2014.

BEZERRA, E. **Princípios de Análise e Projetos de Sistemas com UML**. 2. ed. Rio de Janeiro: ELSEVIER, 2007.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, Ivar. **UML – Guia do Usuário**. 2. Ed. Rio de Janeiro – RJ: Campus; Elsevier, 2006.

DEITEL, PAUL J.; DEITEL, HARVEY - **Java, Como Programar**. 8. Ed. São Paulo – SP: Pearson, 2005.

DEVMEDIA. Disponível em <<http://www.devmedia.com.br>> Acesso em: 26/10/2014.

FAPESP. Disponível em: <<http://revistapesquisa.fapesp.br/2013/01/21/praticar-atividade-fisica-ajuda-a-evitar-agravamento-de-cardiopatia>>. Acesso em 05 jan. 2014.

GORDON, STEVEN R. **Sistemas de Informação - Uma Abordagem Gerencial**. 3. ed. Rio de Janeiro: LTC, 2006.

Juntando as Peças. Disponível em: <<http://luiz.oeducador.com.br/?author=1&paged=3>>. Acesso em 12 Jan. 2014.

O'BRIEN, JAMES A. **Sistemas de Informação e As Decisões Gerenciais na Era da Internet**. 2. ed. São Paulo – SP: Saraiva, 2004.

POSTGRESQL – Disponível em <<http://www.postgresql.org/about/>> - Acesso em 27/08/2014.

RIBEIRO, JOÃO ARAUJO. **Arquitetura de Sistemas de Informação Geográfica**. Disponível em [http://www.geomatica.eng.uerj.br/docentes/araujo/\\_export/s5/arquitetura\\_de\\_sistemas\\_de\\_informacao\\_geografica](http://www.geomatica.eng.uerj.br/docentes/araujo/_export/s5/arquitetura_de_sistemas_de_informacao_geografica)>. Acesso em: 10 jun. 2013.

SPRING MVC - <http://www.caelum.com.br/apostila-java-web/spring-mvc/#11-1-porque-precisamos-de-frameworks-mvc> - Acesso em 12/11/2014

SCHWABER, KEN – **Guia do Scrum**. 2013.

WAZLAWICK, RAUL SIDNEI. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. Rio de Janeiro – RJ: Elsevier, 2011.

WEISSMANN, HENRIQUE L. **Vire o Jogo com Spring Framework**. São Paulo – SP: Casa do Código, 2012.

## APÊNDICE A - NÍVEIS DE DETALHAMENTO DE UM CASO DE USO

<b>Caso de Uso:</b> Receber Pendências
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema deve efetuar o recebimento das pendências.
<b>Pós-condições:</b> Após o recebimento da mesma, deve-se baixar a pendência no sistema.
<b>Requisitos Correlacionados:</b> F11
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> <ol style="list-style-type: none"> <li>1. O cliente chega à academia para efetuar o pagamento de sua dívida.</li> <li>2. O funcionário atende o cliente e verifica o valor da dívida, juros, etc.</li> <li>3. [IN] O funcionário faz o lançamento do pagamento no sistema.</li> <li>4. [OUT] É feito a emissão de um comprovante de pagamento.</li> <li>5. O cliente pode utilizar os benefícios da academia se desejar novamente sem nenhum problema.</li> </ol>
<b>Tratamento de Exceções:</b> <b>2a Se o título já estiver protestado</b> <b>2a.1</b> Orienta o cliente a procurar um cartório de registros de títulos para fazer o acerto. <b>2a.2</b> Retorna ao item 2 do fluxo principal.

**Quadro 11 - Caso de uso detalhado Receber pendências**

<b>Caso de Uso:</b> Controlar Pendências a pagar
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema efetua o controle das pendências a serem pagas.
<b>Pós-condições:</b> O sistema informa através de uma mensagem ao usuário as pendências existentes.
<b>Requisitos Correlacionados:</b> F13
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> <ol style="list-style-type: none"> <li>1. O funcionário verifica no sistema as pendências a serem pagas.</li> <li>2. O funcionário ou responsável efetua o pagamento da mesma.</li> <li>3. [IN] O funcionário faz o lançamento do pagamento no sistema.</li> <li>4. [OUT] É realizada a baixa do sistema da conta paga.</li> </ol>
<b>Tratamento de Exceções:</b>

**2a Se o título já estiver protestado**

**2a.1** Procurar a empresa a qual é devido para renegociação.

**2a.2** Retorna ao item 2 do fluxo principal.

**Quadro 12 - Caso de uso detalhado Controlar pendências a pagar**

<b>Caso de Uso:</b> Incluir Pendências pagas
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema efetua a inclusão das pendências que foram pagas.
<b>Pós-condições:</b> Após o pagamento da mesma, deve-se baixar a pendência do sistema.
<b>Requisitos Correlacionados:</b> F13
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> 1 [IN] O funcionário faz a inclusão do pagamento no sistema. 2 [OUT] É feito a baixa do sistema da conta paga.
<b>Tratamento de Exceções:</b> Não há exceções a serem tratadas

**Quadro 13 - Caso de uso detalhado Incluir pendências pagas**

<b>Caso de Uso:</b> Controlar Cheques emitidos
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema efetua o controle de cheques que foram emitidos.
<b>Pós-condições:</b> Após a emissão, é feito o lançamento no sistema do cheque que foi emitido.
<b>Requisitos Correlacionados:</b> F15
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> 1. O funcionário verifica que há algo a ser pago e faz a emissão de um cheque. 2. [IN] O funcionário ou responsável faz o lançamento do cheque emitido no sistema. 3. [OUT] É emitido um comprovante com o valor da saída.
<b>Tratamento de Exceções:</b> <b>2a Emissão de cheques pré-datados</b> <b>2a.1</b> Verificar antes se não haverá problema quanto à falta de saldo na conta para debitar o cheque. <b>2a.2</b> Retorna ao item 1 do fluxo principal.

**Quadro 14 - Caso de uso detalhado Controlar cheques emitidos**

<b>Caso de Uso:</b> Controlar Cheques Recebidos
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema efetua o controle de cheques que foram recebidos.
<b>Pós-condições:</b> Após o recebimento, é feito o lançamento no sistema do cheque que foi recebido.
<b>Requisitos Correlacionados:</b> F16
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<p><b>Fluxo Principal:</b></p> <ol style="list-style-type: none"> <li>1. O aluno chegar à academia para efetuar o pagamento de sua mensalidade com o cheque em mãos.</li> <li>2. O Funcionário recebe o pagamento.</li> <li>3. [IN] O funcionário ou responsável faz o lançamento do cheque recebido no sistema.</li> <li>4. [OUT] É emitido um comprovante de pagamento.</li> </ol>
<p><b>Tratamento de Exceções:</b></p> <p><b>2a Recebimento de cheques pré-datados</b></p> <p><b>2a.1</b> Fazer uma consulta com o gerente ou responsável antes de recebê-lo.</p> <p><b>2a.2</b> Retorna ao item 2 do fluxo principal.</p> <p><b>3a Recebimentos de cheques com origem duvidosa ou de terceiros</b></p> <p><b>3a.1</b> Fazer uma consulta com o gerente ou responsável antes de recebê-lo.</p> <p><b>3a.2</b> Retorna ao item 2 do fluxo principal.</p>

**Quadro 15 - Caso de uso detalhado Controlar cheques recebidos**

<b>Caso de Uso:</b> Matricular Aluno
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema deve efetuar a matrícula do aluno, informando o plano de atividades desejado, assim como se possui algum convênio.
<b>Pós-condições:</b> É emitido um documento no qual o aluno deve assinar para comprovar a matrícula.
<b>Requisitos Correlacionados:</b> F17
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<p><b>Fluxo Principal:</b></p> <ol style="list-style-type: none"> <li>1. O aluno já cadastrado chega à academia para efetuar sua matrícula.</li> <li>2. O Funcionário confere os dados do aluno e vê em que atividades deseja matricular-</li> </ol>

<p>se.</p> <p>3. [IN] O funcionário efetua a matrícula do aluno na atividade que escolheu.</p> <p>4. [OUT] É emitido um documento como comprovante da matrícula.</p>
<p><b>Tratamento de Exceções:</b></p> <p><b>2a Se o aluno possui algum tipo de pendência financeira com a academia</b></p> <p><b>2a.1</b> Fazer uma consulta com o gerente ou responsável antes de matriculá-lo.</p> <p><b>2a.2</b> Retorna ao passo 3 do fluxo principal.</p>

**Quadro 16 - Caso de uso detalhado Matricular aluno**

<b>Caso de Uso:</b> Cancelar Matrícula
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema deve permitir ao aluno o cancelamento de sua matrícula.
<b>Pós-condições:</b> É emitido um documento no qual o aluno deve assinar para comprovar o cancelamento da matrícula.
<b>Requisitos Correlacionados:</b> F17, F18
<b>Interessados:</b> Funcionário
<b>Variações tecnológicas:</b> Não possui
<p><b>Fluxo Principal:</b></p> <ol style="list-style-type: none"> <li>1. O aluno já cadastrado e matriculado chega à academia para efetuar o cancelamento da matrícula.</li> <li>2. O Funcionário confere os dados do aluno, vê em que atividade(s) está matriculado e confere se não há nenhuma pendência a ser paga pelo mesmo.</li> <li>3. [IN] O funcionário efetua a baixa da matrícula do aluno.</li> <li>4. [OUT] É emitido um documento como comprovante do cancelamento da matrícula.</li> </ol>
<p><b>Tratamento de Exceções:</b></p> <p><b>2a Se o aluno possuir algum tipo de pendência</b></p> <p><b>2a.1</b> O aluno deve pagar seu débito para poder ser efetuado o cancelamento da matrícula.</p> <p><b>2a.2</b> Retorna ao item 2 do fluxo principal.</p>

**Quadro 17 - Caso de uso detalhado Cancelar matrícula**

<b>Caso de Uso:</b> Bloquear Matrícula
<b>Atores:</b> Funcionário
<b>Precondições:</b> O sistema deve permitir o bloqueio de uma matrícula caso haja alguma irregularidade.
<b>Pós-condições:</b> É emitido um aviso para o responsável de que a matrícula do aluno está bloqueada.

<b>Requisitos Correlacionados:</b> F17, F19
<b>Interessados:</b> Funcionário, alunos
<b>Variações tecnológicas:</b> Não possui
<b>Fluxo Principal:</b> <ol style="list-style-type: none"> <li>1. O aluno já cadastrado e matriculado chega a academia para realizar suas atividades.</li> <li>2. O Funcionário confere os dados do aluno e constata que há irregularidades no cadastro do aluno.</li> <li>3. [OUT] É emitido um aviso de que o aluno está impossibilitado de realizar as atividades que normalmente realizava.</li> </ol>
<b>Tratamento de Exceções:</b> <b>2a Se o aluno resolver todas as pendências com a academia</b> <b>2a.1</b> O aluno é liberado para realizar suas atividades normalmente.

**Quadro 18 - Caso de uso detalhado Bloquear matrícula**

## APÊNDICE B - REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

<b>F12 Receber Pendências</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema deve efetuar o recebimento das pendências				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF12.1 Controle de acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade	Usabilidade	( X )	( X )

**Quadro 19 - Requisito Receber pendências**

<b>F13 Controlar pendências à pagar</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema efetua o controle das pendências a serem pagas				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF13.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade.	Segurança	( X )	( X )
NF13.2 Verificação de pendências a serem pagas	Verifica através da data as contas à vencer que deverão ser pagas no dia corrente.	Usabilidade	( )	( X )
NF13.3 Mensagem de aviso	O sistema deverá emitir uma mensagem de aviso das contas que estarão vencidas ou que vencerão no dia corrente.	Usabilidade	( )	( X )

**Quadro 20 - Requisito Controlar pendências à pagar**

<b>F14 Incluir pendências pagas</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema efetua a inclusão das pendências que foram pagas				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>



NF14.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade.	Segurança	( X )	( X )
------------------------------	---	-----------	-------	-------

**Quadro 21 - Requisito Incluir pendências pagas**

<b>F15 Controlar Cheques emitidos</b>	<b>Oculto ( )</b>			
<b>Descrição:</b> O sistema efetua o controle de cheques que foram emitidos				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF15.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade.	Segurança	( X )	( X )
NF15.2 Verificação de cheques que serão debitados	Verifica através da data do sistema se possui algum cheque emitido que poderá ser debitado na conta bancária.	Usabilidade	( )	( X )

**Quadro 22 - Requisito Controlar cheques emitidos**

<b>F16 Controlar Cheques recebidos</b>	<b>Oculto ( )</b>			
<b>Descrição:</b> O sistema efetua o controle de cheques que foram recebidos				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF16.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela atividade.	Segurança	( X )	( X )
NF16.2 Verificação de cheques que foram recebidos	Verifica através da data do sistema se possui algum cheque recebido que poderá ser creditado na conta bancária	Usabilidade	( )	( X )

**Quadro 23 - Requisito Controlar cheques recebidos**

<b>F17 Matricular Aluno</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema deve efetuar a matrícula do aluno, informando o plano de atividades desejado, assim como se possui algum convênio.				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF17.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela matrícula de alunos.	Segurança	( X )	( X )
NF17.2 Identificar aluno pelo código do cadastro	O software irá buscar pelo código do cliente anteriormente cadastrado as informações do aluno.	Interface	( )	( X )
NF17.3 Verificar possíveis pendências	O software verifica pelo código do aluno se existe alguma pendência do mesmo.	Usabilidade	( )	( X )
NF17.4 Identificação do plano de atividade	O plano será identificado por um código previamente cadastrado.	Usabilidade	( )	( X )
NF17.5 Identificação de convênios	O convenio será identificado por um código previamente cadastrado.	Usabilidade	( )	( X )

**Quadro 24 - Requisito Matricular aluno**

<b>F18 Cancelar matrícula</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema deve permitir ao aluno o cancelamento de sua matrícula				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF18.1 Controle de Acesso	A função só poderá ser acessada por um usuário com perfil de administrador ou funcionário responsável pela matrícula de	Segurança	( X )	( X )

	alunos.			
NF18.2 Identificar aluno pelo código do cadastro	O software irá buscar pelo código do aluno anteriormente cadastrado as informações do aluno.	Usabilidade	( )	( X )
NF18.3 Verificar possíveis pendencias	O software verifica pelo código do aluno se existe alguma pendência do mesmo não tornando possível o cancelamento se houver.	Usabilidade	( )	( X )

**Quadro 25 - Requisito Cancelar matrícula**

<b>F19 Bloquear matrícula</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema irá bloquear a matrícula em caso de atraso de pagamento e só liberando quando pago o débito com a devida multa calculada.				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF19.1 Realizar bloqueio	O sistema realizará o bloqueio do cliente caso possua alguma pendência.	Usabilidade	( )	( X )
NF19.2 Identificar aluno pelo código do cadastro	O software irá buscar pelo código do aluno anteriormente cadastrado as informações do aluno.	Usabilidade	( )	( X )
NF19.3	O sistema irá atribuir uma multa correspondente ao débito e ao tempo de atraso do mesmo	Usabilidade	( )	( X )

**Quadro 26 - Requisito Bloquear matrícula**

<b>F20 Gerar Relatórios</b>		<b>Oculto ( )</b>		
<b>Descrição:</b> O sistema irá gerar relatório resgatando informações previamente cadastradas no sistema				
<b>Requisitos Não Funcionais</b>				
<b>Nome</b>	<b>Restrição</b>	<b>Categoria</b>	<b>Desejável</b>	<b>Permanente</b>
NF20.1 Gera relatório de	O sistema	Usabilidade	( )	( X )

acompanhamento físico	realizará uma busca das informações previamente cadastradas no acompanhamento físico verificando as mudanças	e		
NF20.2 Gera relatório de contas a receber	O sistema realizará uma busca das informações previamente cadastradas no controle financeiro verificando as contas a receber.	Usabilidade	( )	( X )
NF20.3 Gera relatório de contas a pagar	O sistema realizará uma busca das informações previamente cadastradas no controle financeiro verificando as contas a pagar.	Usabilidade	( )	( X )
NF20.4 Gera relatório de alunos matriculados	O sistema realizará uma busca das informações previamente cadastradas na	Usabilidade	( )	( X )

	matrícula de alunos verificando os alunos matriculados.			
NF20.5 Gera relatório de alunos ativos	O sistema realizará uma busca das informações previamente cadastradas na matrícula de alunos verificando os alunos que estão relacionados com alguma atividade.	Usabilidade	( )	( X )
NF20.6 Gera relatório pagamentos(alunos)	O sistema realizará uma busca das informações previamente cadastradas no controle financeiro verificando os pagamentos efetuados pelos alunos.	Usabilidade	( )	( X )
NF20.7 Gera relatório pagamentos/recebimentos(academia)	O sistema realizará uma busca das informações previamente	Usabilidade	( )	( X )

	<p>cadastradas no controle financeiro fazendo um comparativo das entradas e saídas do mês.</p>			
<p>NF20.8 Gera relatório de alunos pendentes</p>	<p>O sistema realizará uma busca das informações previamente cadastradas no bloqueio de alunos verificando quais alunos estão pendentes.</p>	<p>Usabilidade e</p>	<p>( )</p>	<p>( X )</p>

**Quadro 27 - Requisito Gerar relatórios**