

Rosivaldo Freitas de Souza Filho

SISTEMA DE GERENCIAMENTO DE IGREJAS EM CÉLULAS EM MODELO MDA

Palmas

2016

Rosivaldo Freitas de Souza Filho

SISTEMA DE GERENCIAMENTO DE IGREJAS EM CÉLULAS EM MODELO MDA

Trabalho de Conclusão de Curso submetido à coordenação do curso de CST Sistemas Para Internet - Campus Palmas, como requisito parcial para obtenção do grau de Tecnólogo em Sistemas Para Internet.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
TOCANTINS CAMPUS PALMAS

CST Sistemas Para Internet

Orientador: Prof. Esp. Vinicius Istofel Oliveira

Palmas

2016

Rosivaldo Freitas de Souza Filho

SISTEMA DE GERENCIAMENTO DE IGREJAS EM CÉLULAS EM MODELO MDA

Trabalho de Conclusão de Curso submetido à coordenação do curso de CST Sistemas Para Internet - Campus Palmas, como requisito parcial para obtenção do grau de Tecnólogo em Sistemas Para Internet.

Trabalho aprovado. Palmas, 19 de Setembro de 2016:

Prof. Esp. Vinicius Istofel Oliveira
Orientador

Prof. Me. Marinaldo Oliveira Santos
Examinador Interno

**Prof. Me. Francisco Willians Makoto
P Hirano**
Examinador Interno

Palmas
2016

Agradecimentos

Agradeço a Deus, por ter me concedido saúde, força pra seguir em frente em todas as fases de construção desse trabalho, desde o momento em que esse projeto era apenas um sonho.

Agradeço muito à minha família, Rosivaldo Souza, Carmen Souza e Renan Souza, por me encorajarem e por terem me apoiado a vida toda em todos os desafios e lutas, por terem participado direta e indiretamente dos processos de concepção e criação desse projeto, sonharam junto, planejaram junto, e estiveram sempre dispostos a me ajudar nos momentos que eu mais precisei, e não me deixaram desistir mesmo quando eu achava que não podia mais.

Agradeço ao corpo docente do Instituto Federal pelo empenho e dedicação em transmitir o conhecimento, que é o bem mais valioso que podemos adquirir, desenvolver e compartilhar como profissionais em qualquer área.

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

Resumo

Assim como toda instituição constituída por hierarquias de trabalho e prestação de contas, as igrejas, em uma visão geral, assim como qualquer empresa, necessitam de métodos para sua própria gestão e organização. No modelo celular de igrejas, uma célula é um grupo que se reúne semanalmente em um dia e horário determinado. As células são parte de um todo que é a igreja, elas são interdependentes. Contidas em uma estrutura hierárquica distribuída geograficamente em Setores, Áreas, Distritos e Regiões.

O objetivo deste trabalho é demonstrar como foi desenvolvido um sistema web utilizado como uma forma de auxiliar a organização e gestão desse modelo organizacional de igrejas, mais especificamente o chamado *Modelo de Discipulado Apostólico (MDA)*, apresentando a metodologia utilizada em cada etapa dos processos de concepção e desenvolvimento do trabalho, assim como as tecnologias e ferramentas utilizadas na construção da aplicação.

Palavras-chaves: Igreja, Células, MDA, Sistema, Web.

Abstract

As every institution made up of work hierarchies and accountability, the churches, in an overview, as well as any company, they need methods for their own management and organization. In cellular model of churches, a cell is a group that I meet weekly one day and certain time. The cells are part of a whole which is the church, they are interdependent. Contained in a hierarchical structure distributed geographically in Sectors, Areas, Districts and Regions.

The purpose of this study is to demonstrate how we developed a web system used as a way to help the organization and management of this organizational model churches, specifically the so-called *Apostolic Discipleship Model (MDA)*, presenting the methodology used in each step of the design process and development work, as well as the technologies and tools used in the implementation of construction.

Key-words: Churches, Cells, MDA, System, Web.

Lista de ilustrações

Figura 1 – Modelo Estrutural MDA	17
Figura 2 – Modelo do contexto Celular	18
Figura 3 – Padrão MVC (BAUTAZAR, 2006)	22
Figura 4 – Exemplo de modelo ER	24
Figura 5 – Diagrama de Casos de Uso	39
Figura 6 – Diagrama de Classes	61
Figura 7 – Exemplo de View no sistema - Dashboard	64
Figura 8 – Organização do Código	68
Figura 9 – Processo de Autenticação - <i>Spring Security</i> (WEISSMANN, 2012, p.235)	69
Figura 10 – Relatório de Célula em PDF	73
Figura 11 – Relatório de Célula	73
Figura 12 – Dashboard Responsivo	80
Figura 13 – Teste de Validação	82
Figura 14 – Teste de Validação	83
Figura 15 – Serviço Virtual Privado	88
Figura 16 – Informações do Sistema em Produção	89
Figura 17 – Tela de Login	90
Figura 18 – Dashboard	90
Figura 19 – Lista de Membros	90
Figura 20 – Tela de Detalhamento de Membro	91
Figura 21 – Tela de Cadastro de Membro	91
Figura 22 – Tela de Lista de Células	91
Figura 23 – Tela de Detalhamento de Célula	92
Figura 24 – Tela de Reuniões de Célula	92
Figura 25 – Tela de Cadastro de Reunião de Célula	92
Figura 26 – Tela de Lista de Setores	93
Figura 27 – Tela de Edição de Setores	93
Figura 28 – Tela de Lista de Reuniões de Treinamento de Lideres (TADEL)	93
Figura 29 – Tela de Edição de TADEL	94
Figura 30 – Tela de Lista de Cultos	94
Figura 31 – Tela de Edição de Culto	94
Figura 32 – Tela de Lista de discipulados	95
Figura 33 – Tela de Detalhamento de Discipulado	95
Figura 34 – Tela de Nova Reunião de Discipulado	95
Figura 35 – Tela de Trilho de Liderança	96
Figura 36 – Tela de Logout	96

Figura 37 – Formulário de Célula Atual Preenchido à mão	98
---	----

Lista de tabelas

Tabela 1 – Requisito Funcional - Autenticar Usuário	34
Tabela 2 – Requisito Funcional - Manter Membros	34
Tabela 3 – Requisito Funcional - Manter Células	35
Tabela 4 – Requisito Funcional - Manter Setores	35
Tabela 5 – Requisito Funcional - Manter Reunião de Célula	36
Tabela 6 – Requisito Funcional - Manter Reunião de Treinamento de Lideres (Tadel)	36
Tabela 7 – Requisito Funcional - Manter Culto de Celebração	36
Tabela 8 – Requisito Funcional - Consultar Relatório de Célula	37
Tabela 9 – Requisito Funcional - Manter Discipulado	37
Tabela 10 – Requisitos Não Funcionais	38
Tabela 11 – Caso de Uso - Manter Membros	40
Tabela 12 – Caso de Uso - Manter Células	43
Tabela 13 – Caso de Uso - Manter REUNIÃO DE CÉLULA	45
Tabela 14 – Caso de Uso - Manter DISCIPULADO	48
Tabela 15 – Caso de Uso - MANTER TADEL	50
Tabela 16 – Caso de Uso - MANTER CULTO DE CELEBRAÇÃO	52
Tabela 17 – Caso de Uso - MANTER FREQUÊNCIA DO CULTO DE CELEBRAÇÃO	54
Tabela 18 – Caso de Uso - MANTER FREQUÊNCIA DO TADEL	56
Tabela 19 – Caso de Uso - Consultar Membros	57
Tabela 20 – Caso de Uso - Consultar Células	58
Tabela 21 – Caso de Uso - Consultar Discipulado	59
Tabela 22 – Caso de Uso - Emitir Relatório	60

Lista de abreviaturas e siglas

UML	Unified Modeling Language
JSF	JavaServer Faces
API	Application Programming Interface
HTML	Hyper Text Markup Language
JPA	Java Persistence API
XHTML	Xtensible Hypertext Markup Language
CSS	Cascading Style Sheets
JDK	Java Developer Kit
JRE	Java Runtime Enviroment
JVM	Java Virtual Machine
JAR	Java ARchive
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and XML
MVC	Model View Controller
POJO	Plain Old Java Objects
HQL	Hibernate Query Language
SQL	Structured Query Language
JSP	Java Server Pages
VPS	Servidor Virtual Privado
SSH	Secure Shell

Sumário

1	INTRODUÇÃO	14
1.1	Justificativa	15
1.2	Objetivos	15
1.2.1	Objetivo Geral	15
1.2.2	Objetivos Específicos	15
2	ESTRUTURA ORGANIZACIONAL	16
2.1	Modelo Celular	16
2.2	Modelo Mda	17
2.2.1	Estrutura	17
2.2.2	Célula	18
2.2.3	Discipulado	18
3	PREPARAÇÃO DA PESQUISA	19
3.1	Tipo de Pesquisa	19
3.2	Metodologia	19
3.2.1	Levantamento Bibliográfico	19
3.2.2	Levantamento de Requisitos	19
3.2.3	Análise e Projeto	20
3.2.4	Desenvolvimento	20
3.2.5	Testes	20
3.2.6	Implantação	20
4	APORTE TEÓRICO	21
4.1	Engenharia de Software	21
4.1.1	Padrão de Arquitetura MVC	21
4.2	UML	22
4.3	Banco de Dados	22
4.3.1	Modelo Conceitual	23
4.3.2	Modelo Lógico	23
4.3.3	Modelo Físico	23
4.3.4	Abordagem Entidade Relacionamento	24
4.4	Web Design Responsivo	24
4.4.1	Tipos de Medidas em CSS	25
4.5	Tecnologias	25
4.5.1	Java	25

4.5.1.1	Java EE	26
4.5.2	JSF	27
4.5.3	Primefaces	27
4.5.4	Spring Security	28
4.5.5	PostgreSQL	29
4.5.6	JPA	29
4.5.7	Hibernate	30
4.5.8	Twitter Bootstrap	31
4.6	Ferramentas	31
4.6.1	Apache Tomcat	31
4.6.2	Astah	31
4.6.3	Ambiente de Desenvolvimento Netbeans	32
4.6.4	iText	33
4.6.5	JRebel	33
5	REQUISITOS DO SISTEMA	34
5.1	Requisitos Funcionais	34
5.2	Requisitos Não Funcionais	38
5.3	Diagrama de Casos de Uso	39
5.4	Detalhamento dos Casos de Uso	40
5.5	Diagrama de Classes	61
5.5.1	Descrição do Diagrama de Classes	62
6	DESENVOLVIMENTO	63
6.1	Implementando Padrão MVC	63
6.1.1	Model	63
6.1.2	View	64
6.1.3	Controller	66
6.2	Organização do Código Fonte	68
6.3	Autenticação e Autorização com Spring Security	69
6.3.1	Filtro de Acesso	69
6.3.2	Gerenciador de Autenticação	70
6.3.3	Provedor de Autenticação	70
6.4	Relatório de Célula	72
6.5	Interface Web	75
6.5.1	Template com Facelets	75
6.5.2	Layout	78
6.5.3	Integração JSF/Bootstrap	79
6.5.4	Interface Web-responsiva	80
6.6	Testes	81

6.6.1	Testes de Rotinas	81
6.6.2	Testes de Verificação e Validação	82
7	CONCLUSÃO E TRABALHOS FUTUROS	84
7.1	Conclusão	84
7.2	Trabalhos Futuros	84
	REFERÊNCIAS	85
	APÊNDICES	87
	APÊNDICE A – IMPLANTACAO	88
A.1	Servidor	88
A.2	Serviços	88
A.3	Aplicação em Produção	89
	APÊNDICE B – TELAS DO SISTEMA	90
C –	ANEXO A - FORMULÁRIO DE CÉLULA	98

1 Introdução

O modelo de Igreja Em Célula, surgiu no início dos anos 1960 pelo pastor coreano David Yonggi Cho, fundador da Igreja do Evangelho Pleno de Yoido, e aperfeiçoada duas décadas depois pelo pastor colombiano *César Castellanos*, criador da Missão Carismática Internacional, que sistematizou o método e o renomeou para G12 ([ALEIXO, 2014](#)).

No Brasil, o modelo organizacional de igrejas em célula chamado *Modelo de Discipulado Apostólico (MDA)* criado em 1999 pelo pastor *Abe Huber* trabalha com uma estrutura de hierarquia de liderança e supervisão, onde cada nível da hierarquia conta com um Líder (supervisor) que precisa prestar contas com o Supervisor do nível superior a respeito dos membros que estão debaixo de sua cobertura ([HUBER, 1999](#)).

A Igreja da Paz no Brasil é regida pelo modelo MDA, porém existem alguns problemas relacionados com a forma de prestação de contas de cada nível da hierarquia, um deles é o fato de todos os relatórios serem preenchidos e analisados de forma manual, com tabelas impressas onde cada supervisor deve preencher as informações pertinentes ao seu nível.

Consequentemente esse modo de trabalho leva a inconsistência de dados, provocados por muitos fatores, um deles é a dependência de uma boa caligrafia de quem preencha os relatórios. Outra consequência é a redundância de informações provocada pela falta de validação eficaz em dados repetidos, já que as tabelas estão impressas ou em tabelas do Excel onde mesmo que seja possível usar um filtro, ainda há possibilidades de encontrarmos erros de digitação. Existe uma demora no processo de coleta e análise de dados, justamente pelo fato das tabelas serem impressas é necessário que haja um encontro entre os supervisores para a entrega desse relatório, gerando uma considerável latência entre o preenchimento do relatório e a análise final dos dados.

A solução proposta para esses problemas é um sistema Web que propicie o gerenciamento de dados da igreja e posteriormente gere relatórios que auxiliam de forma rápida e eficiente as tomadas de decisões, reduzindo tempo empregado no processo, diminuindo o consumo de recursos como o papel e contornando os problemas relacionados com a escrita manual.

1.1 Justificativa

As tomadas de decisões em relação a mudanças de estratégias na gestão das células de dos demais níveis acima como Setores e Áreas são realizadas tendo como base as informações obtidas através das formulários que são entregues pelos supervisores aos líderes de célula.

Há uma grande necessidade por parte dos líderes e supervisores de obter as informações a respeito das reuniões de células de uma forma mais rápida e consistente, ou seja, de uma forma que contorne os problemas advindos de preenchimentos manuais de formulários. Ao utilizar uma solução automatizada como a que foi desenvolvida, é possível obter os resultados dos processamentos dessas informações em tempo real, podendo utilizar os resultados para gerar relatórios que auxiliam na visualização do desenvolvimento quantitativo e qualitativo da igreja.

A automação desses processos beneficia diretamente os envolvidos com o processo de manipulação das informações da igreja, facilitando a coleta, transferência e consulta de dados, diminuindo drasticamente o problemas de inconsistência de informações, tornando os relatórios mais precisos e rápidos.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral desenvolver um sistema web que atenda as necessidades de gestão organizacional de igrejas em células com o modelo celular MDA.

1.2.2 Objetivos Específicos

- Avaliar a incoerência de informação na gestão organizacional da igreja;
- Construir uma interface web-responsiva;
- Gerar relatório dinâmico de fácil manuseio e compreensão.
- Resolver o problema de inconsistência de dados da *Igreja da Paz de Miracema TO*

2 Estrutura organizacional

Estrutura organizacional é o instrumento administrativo resultante da identificação, análise, ordenação e agrupamento das atividades e dos recursos das empresas.([OLIVEIRA, 2006](#))

A estrutura organizacional é definida como um instrumento gerencial utilizado para atingir os objetivos organizacionais, resultado do processo de divisão do trabalho e da definição de meios para coordená-lo.([SEIFFERT, 2007](#))

2.1 Modelo Celular

O modelo de Igreja Em Célula, surgiu no início dos anos 1960 pelo pastor coreano *David Yonggi Cho*, fundador da *Igreja do Evangelho Pleno de Yoido*, e aperfeiçoada duas décadas depois pelo pastor colombiano *César Castellanos*, criador da Missão Carismática Internacional, que sistematizou o método e o renomeou para G12 ([ALEIXO, 2014](#)).

No universo eclesial, quando se fala em *Modelo Celular* está sendo referenciado uma estrutura organizacional de igrejas em célula. As igrejas em Célula utilizam diversos tipos de modelos organizacionais para estruturar e coordenar as células, cada um com características e particularidades que os diferem uns dos outros, mas o objetivo principal de cada modelo é ser uma ferramenta de gestão que dê suporte as atividades administrativas da igreja.

2.2 Modelo Mda

2.2.1 Estrutura

No Brasil, o modelo organizacional de igrejas em célula chamado *Modelo de Discipulado Apostólico (MDA)* foi criado em 1999 pelo pastor *Abe Huber*, trabalha com uma estrutura de hierarquia de liderança e supervisão, onde cada nível da hierarquia conta com um Líder (supervisor) que precisa prestar contas com o Supervisor do nível superior a respeito dos membros que estão debaixo de sua cobertura (HUBER, 1999).

Estruturalmente uma célula é formada por um grupo de membros, um grupo definido de células compõem um Setor, por sua vez um grupo de Setores compõem uma Área, grupos de Áreas formam um Distrito, e por fim, Grupos de Distritos formam uma Região. Na figura 2 temos uma ilustração de um modelo de uma igreja em células definida com o nível hierárquico de Área.

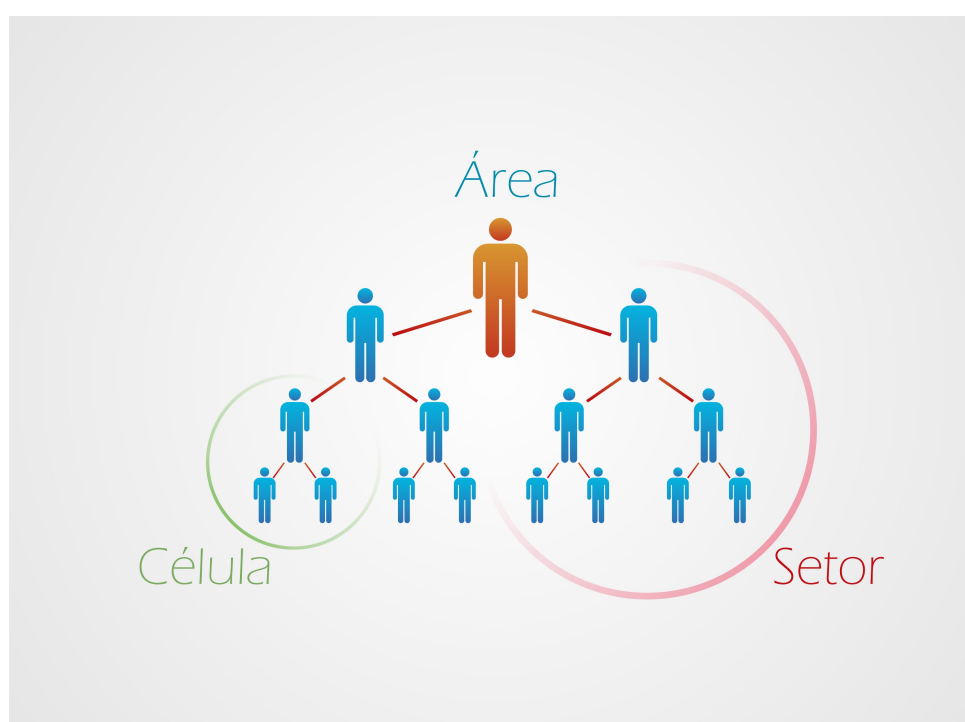


Figura 1 – Modelo Estrutural MDA

2.2.2 Célula

Célula é definida como um grupo de membros de uma igreja que se reúne semanalmente em um dia e horário determinado na residência de um de seus membros. Cada célula possui um membro com a função de líder que tem a incumbência de organizar as reuniões e relatar ao seu supervisor as informações a respeito das atividades da célula.

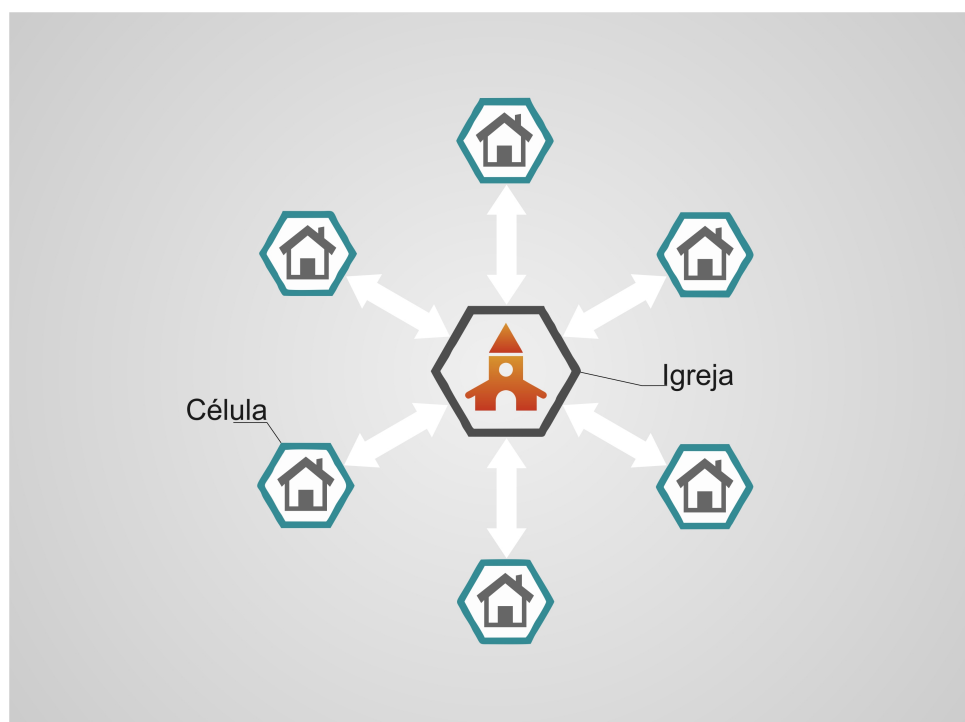


Figura 2 – Modelo do contexto Celular

2.2.3 Discipulado

O Discipulado é a base do modelo MDA e a sua principal característica é definir que cada membro se torne responsável por outro membro, independente da célula que eles façam parte. Cada Discípulo presta contas a seu Líder de Célula a respeito das reuniões que esse discípulo teve com seu discipulador, posteriormente esse líder de célula informa em seu relatório semanal se as reuniões de discipulado da sua célula estão acontecendo, para que no final do mês se tenha uma noção do índice regularidade das reuniões de discipulado na igreja, esse é um dos índices utilizados para medir a qualidade do trabalho de uma igreja em Células no modelo MDA.

3 Preparação da pesquisa

3.1 Tipo de Pesquisa

Este trabalho consiste em uma pesquisa de natureza aplicada e caráter exploratório que visa a compreensão e o aperfeiçoamento de processos já existentes para a produção de conhecimento tecnológico suficiente para o desenvolvimento de ferramentas que deem suporte ao cliente no que se refere a gestão de sua estrutura organizacional.

3.2 Metodologia

Os passos tomados para a construção deste trabalho foram os seguintes:

- Levantamento Bibliográfico
- Levantamento de Requisitos
- Análise e Projeto
- Desenvolvimento
- Testes
- Implantação

3.2.1 Levantamento Bibliográfico

Para este projeto foram feitas pesquisas em livros, artigos científicos e sites relacionados ao desenvolvimento de aplicações web e documentos que descrevam formalmente a política de trabalho de igrejas em células no modelo MDA. Concomitantemente foram realizadas pesquisas a respeito de sistemas similares para se ter melhor embasamento prático a respeito das funcionalidades e usabilidade já utilizada nos mesmos.

3.2.2 Levantamento de Requisitos

Foram realizadas entrevistas com os funcionários da igreja a fim de levantar os requisitos do sistema para melhor entendimento de suas regras de negócio. Foram realizadas reuniões com os *stakeholders* responsáveis pela secretaria da igreja, onde foram coletados os requisitos do sistema através de questionários, conversas direcionadas e observações aos processos de trabalho.

Foram analisados sistemas computacionais já em uso assim como procedimentos e métodos utilizados no processo de preenchimento, entrega e análise de relatórios de *Células* e *Setores*, deixando de fora os níveis de *Área*, *Distrito* e *Região*, pois a igreja em questão ainda não se enquadra nesses níveis de estrutura.

No final dessa fase do processo foi confeccionada uma documentação com os requisitos funcionais e não funcionais e as descrições de cada caso de uso do sistema.

3.2.3 Análise e Projeto

Nesta fase foi realizada a modelagem do sistema, que inclui o diagrama de casos de usos e diagrama de classes para que possam ser melhor visualizadas as características e necessidades do sistema, o final desse processo resultou na confecção de uma documentação de software que foi a base de orientação para a fase de desenvolvimento.

3.2.4 Desenvolvimento

Nesta fase foi feita a codificação da aplicação, partindo dos modelos da fase de projeto, visando concretizar todos os pontos diagramados na modelagem do sistema. Foi utilizada a linguagem de programação Java, tendo como modelo de processos o *modelo de Desenvolvimento Incremental e Iterativo*, esse modelo trabalha com metodologia ágil e necessita de um contato mais dinâmico e interativo com um *stakeholders* especialista. O modelo incremental iterativo terá atividades simultâneas e intercaladas (SOMMERVILLE, 2011).

3.2.5 Testes

Foram realizados testes unitários e de rotinas no paralelamente a fase de desenvolvimento, e testes de certificação e validação juntamente com o *feedback* dos *stakeholders* especialistas, pois a cada funcionalidade pronta foi avaliada pelo cliente para que se prosseguisse para o desenvolvimento da próxima funcionalidade se tudo estivesse de acordo com as especificações ou então eram realizadas as correções necessárias para que o sistema correspondesse ao desejado, para que o resultado não estivesse fora do que já havia sido especificado na fase de modelagem.

3.2.6 Implantação

Nesta fase a aplicação foi colocada em produção, em um servidor web em nuvem para ser acessado pelos *stakeholders* principais, ou seja, os líderes de Célula, o secretário e o supervisor de Setor.

4 Aporte Teórico

4.1 Engenharia de Software

"Engenharia de software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de maneira eficiente em máquinas reais".(PRESSMAN, 2011, p. 39)

Engenharia de software: (1) A aplicação de uma abordagem sistemática, disciplinada e qualificável no desenvolvimento, na operação e na manutenção de software; Isto é, a aplicação de engenharia ao software. (2) O estudo de abordagens como definido em (1). (PRESSMAN, 2011, p. 39).

O autor menciona ainda que a Engenharia de Software é uma tecnologia em camadas sendo elas: Ferramentas, Métodos e Processos, sendo que a base para a engenharia de software é a camada de processos onde se define a metodologia a ser utilizada para se ter o controle do gerenciamento de projetos de software. Os Métodos da engenharia fornecem as informações técnicas para desenvolver software, nessa camada encontramos tarefas como: análise de requisitos, modelagem de projeto, construção de programa, testes e suporte. Na camada de Ferramentas encontra-se o suporte automatizado ou semi-automatizado para o processo e para os métodos. (PRESSMAN, 2011)

4.1.1 Padrão de Arquitetura MVC

MVC (Model View Controller) é um padrão de arquitetura de software que visa separar a lógica de negócio da interface com o usuário, assim os programadores podem mudar facilmente cada parte, sem afetar as outras.

Segundo (CORDEIRO, 2012) No padrão MVC, o modelo(Model) representa as informações (os dados) e as regras de negócio, a visão(View) contém elemento de interface com o usuário, como textos, formulários, e o controle(Controller) gerencia a comunicação entre o modelo e a visão.

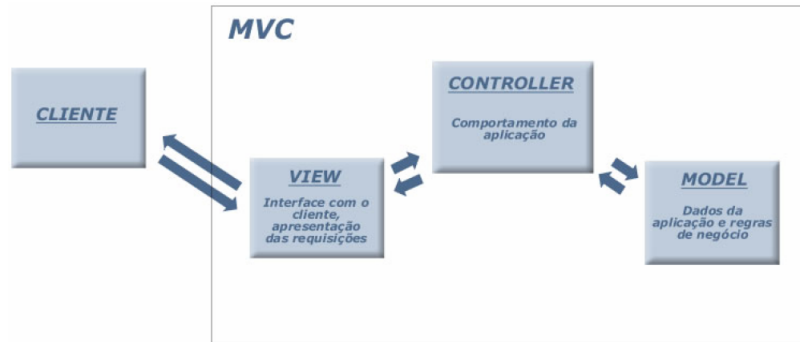


Figura 3 – Padrão MVC (BAUTAZAR, 2006)

4.2 UML

A UML (Unified Modeling Language), que significa Linguagem Unificada de Modelagem é uma linguagem padrão para modelagem orientada a objetos. Ela surgiu da fusão de três grandes métodos, do BOOCH, OMT (Rumbaugh) e OOSE (Jacobson).

UML (Unified Modeling Language) 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 GRADY, 2012, p. 14).

Trata-se de uma linguagem visual para modelar sistemas orientados a objetos, ou seja, é uma linguagem visual que utiliza gráficos para a representação de sistemas complexos. Através dela podem ser representados graficamente diversas perspectivas de um mesmo sistema.

Cada elemento gráfico possui uma sintaxe, ou seja, uma forma predeterminada de desenhar o elemento, e uma semântica que define o seu significado e o objetivo de sua utilização. Outra característica da UML é o fato dela ser extensível, que significa que podem ser agregados a ela diversos elementos de extensão, o que permite que a UML seja adaptada às características específicas de cada projeto de desenvolvimento.

A UML não depende de qual linguagem de programação ou processo de desenvolvimento será utilizado no decorrer do desenvolvimento do software, isso torna essa linguagem mais abrangente, pois diferentes sistemas de software requerem diferentes abordagens de desenvolvimento. (BEZERRA, 2002, p. 14-15)

4.3 Banco de Dados

Banco de dados é "um conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários" (HEUSER, 1998)

Quando se trata de banco de dados, existem sistemas especializados na persistência de dados que oferecem recursos sofisticados e eficientes para esse tipo de objetivo. Esses sistemas são conhecidos como Sistemas Gerenciadores de Bancos de Dados - SGBD. (K19, 2015a)

”sistema de gerência de banco de dados (SGBD) é um software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados”(HEUSER, 1998)

Para descrever o tipo de informações que serão gravadas em um banco de dados utiliza-se um *Modelo de Banco de Dados*, que representa a estrutura formal de um banco de dados. Para construir um modelo de dados, usa-se uma linguagem de modelagem de dados. Linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens textuais ou linguagens gráficas.(HEUSER, 1998)

”No projeto de banco de dados, normalmente são considerados dois níveis de abstração de modelo de dados, o do modelo conceitual e o do modelo lógico.” (HEUSER, 1998)

4.3.1 Modelo Conceitual

Um modelo conceitual de um banco de dados é uma descrição de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD. (HEUSER, 1998)

4.3.2 Modelo Lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado. (HEUSER, 1998)

4.3.3 Modelo Físico

Eles são usados por profissionais que fazem sintonia de banco de dados, procurando otimizar a performance. As linguagens e notações para o modelo físico não são padronizadas e variam de produto a produto. A tendência em produtos mais modernos é esconder o modelo físico do usuário e transferir a tarefa de otimização ao próprio SGBD.(HEUSER, 1998)

4.3.4 Abordagem Entidade Relacionamento

A abordagem entidade-relacionamento (ER) é uma técnica de modelagem de dados que foi criada em 1976 por *Peter Chen*. Ela pode ser considerada como um padrão de fato para modelagem conceitual. Nesta técnica, o modelo de dados é representado através de um modelo entidade-relacionamento (*modelo ER*). Usualmente, um modelo ER é representado graficamente, através de um diagrama entidade relacionamento (*DER*). Mesmo as técnicas de modelagem orientada a objetos que têm surgido nos últimos anos baseiam-se nos conceitos da abordagem ER. ([HEUSER, 1998](#))

Os principais conceitos da abordagem entidade relacionamento são os seguintes:

- Entidade: Conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados.
- Relacionamento: Conjunto de associações entre entidades. Em um DER, um relacionamento é representado através de um losango, ligado por linhas aos retângulos representativos das entidades que participam do relacionamento.
- Cardinalidade: número mínimo e máximo de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento



Figura 4 – Exemplo de modelo ER

4.4 Web Design Responsivo

”web design responsivo, que é aquele web design que responde a quaisquer dispositivos/resoluções e, devido a uma série de características técnicas bem específicas, é bem apresentado em qualquer um deles.” ([ZEMEL, 2012](#), p. 17)

Para conseguir desenvolver um design responsivo para a web, as tecnologias principais (ou modos de aplicar essas tecnologias, se preferir) estão envolvidas: ([ZEMEL, 2012](#))

- Layout Fluído: prima pela não especificação de medidas fixas no layout do projeto, torna possível que haja uma adaptação “natural” e automática do que se apresenta na tela. Portanto, seja qual for a resolução do dispositivo que fez o acesso, evitamos

barras de rolagem inconvenientes e/ou conteúdos "cortados", não exibidos em sua completude.(ZEMEL, 2012)

- técnicas variadas, para fazer com que os assets (recursos como imagens, vídeos, etc) do site também sejam flexíveis para garantir que a experiência do visitante prime pela excelência, independentemente do dispositivo que esteja usando.(ZEMEL, 2012)
- Média Queries: torna possível ocultar, fazer aparecer e reposicionar elementos e interações conforme a resolução atual que esteja sendo usada no momento da visitação. Afinal, o site não precisa (e, na verdade, não deve) ter, exatamente, a mesma aparência e disposição de seus elementos em qualquer resolução.(ZEMEL, 2012)

4.4.1 Tipos de Medidas em CSS

- Pixel: É a unidade de medida fixa mais usada nas CSS. Comumente falando, um pixel é um ponto indivisível na tela de exibição de um dispositivo.(ZEMEL, 2012)
- Ponto (point): são tradicionalmente utilizados para CSS de impressão. Um ponto é igual a 1/72 polegadas. Assim como os pixels, pontos são unidades de tamanho fixo.(ZEMEL, 2012)
- Sem (em): O "em" é uma unidade escalável. Quando se trata do tamanho da fonte, 1em é igual ao tamanho atual da fonte do elemento-pai. Ems são escaláveis por natureza.(ZEMEL, 2012)
- Porcentagem(%): A unidade por cento é muito parecida coma unidade "em" mas possui algumas diferenças fundamentais. Em primeiro lugar, o atual tamanho da fonte é igual a 100%(ou seja 12pt = 100%). Durante o uso da unidade por cento, o texto permanece totalmente escalável para dispositivos móveis.(ZEMEL, 2012)

4.5 Tecnologias

4.5.1 Java

Para esse projeto foi escolhido a linguagem de desenvolvimento Java por se tratar de uma linguagem de alto nível, orientada a objetos e multiplataforma.

"A linguagem de programação Java representa uma linguagem simples, orientada a objetos, multithread, interpretada, neutra de arquitetura, portátil, robusta, segura e que oferece alto desempenho"(MENDES, 2009, p. 17). Também complementa apontando o fado de que a tecnologia Java é composta de uma linguagem de programação e de uma plataforma (API e a máquina virtual).

A linguagem Java é considerada simples porque permite o desenvolvimento de sistemas em diferentes sistemas operacionais e arquiteturas de hardware, sem que o programador tenha que se preocupar com detalhes de infra-estrutura. Dessa forma, o programador consegue desempenhar seu trabalho de uma forma mais produtiva e eficiente. (MENDES, 2009, p. 17)

Para desenvolvimento desse projeto se torna necessário a utilização do Java Development Kit (JDK) do Java SE (Standard Edition) que é um conjunto de ferramentas para desenvolvedores somado à Java Runtime Environment (JRE) que é o ambiente de execução Java, formado pela JVM e bibliotecas.

4.5.1.1 Java EE

”O Java EE (Java Enterprise Edition) consiste de uma série de especificações bem detalhadas, dando uma receita de como deve ser implementado um software que faz cada um desses serviços de infraestrutura.” (CAELUM, 2016)

A Java EE (Java Platform, Enterprise Edition) é uma plataforma padrão para desenvolver aplicações Java de grande porte e/ou para a internet, que inclui bibliotecas e funcionalidades para implementar software Java distribuído, baseado em componentes modulares que executam em servidores de aplicações e que suportam escalabilidade, segurança, integridade e outros requisitos de aplicações corporativas ou de grande porte. (FARIA, 2013, p. 14)

Segundo (FARIA, 2013) a plataforma Java EE possui uma série de especificações (tecnologias) com objetivos distintos, por isso é considerada uma plataforma guarda-chuva. Entre as especificações da Java EE, as mais conhecidas são:

- **Servlets:** são componentes Java executados no servidor para gerar conteúdo dinâmico para a web, como HTML e XML. (FARIA, 2013)
- **JSP (*JavaServer Pages*):** uma especialização de Servlets que permite que aplicações web desenvolvidas em Java sejam mais fáceis de manter. É similar às tecnologias como ASP e PHP, porém mais robusta por ter todas as facilidades da plataforma Java. (FARIA, 2013)
- **JSF (*JavaServer Faces*):** é um framework web baseado em Java que tem como objetivo simplificar o desenvolvimento de interfaces (telas) de sistemas para a web, através de um modelo de componentes reutilizáveis. A proposta é que os sistemas sejam desenvolvidos com a mesma facilidade e produtividade que se desenvolve sistemas desktop (até mesmo com ferramentas que suportam clicar-e-arrastar componentes). (FARIA, 2013)

- **JPA (*Java Persistence API*):** é uma API padrão do Java para persistência de dados, que usa um conceito de mapeamento objeto-relacional. Essa tecnologia traz alta produtividade para o desenvolvimento de sistemas que necessitam de integração com banco de dados. Só para citar, essa API possibilita que você desenvolva aplicações usando banco de dados sem precisar escrever uma linha sequer de SQL.([FARIA, 2013](#))
- **EJB (*Enterprise Java Beans*):** são componentes que executam em servidores de aplicação e possuem como principais objetivos, fornecer facilidade e produtividade no desenvolvimento de componentes distribuídos, transacionados, seguros e portáteis.([FARIA, 2013](#))

4.5.2 JSF

Segundo ([FARIA, 2013](#)) Java Server Faces (JSF) é uma tecnologia para desenvolvimento web baseado no padrão de projeto MVC (Model View Controller), o que torna o desenvolvimento de sistemas menos complicado e mais coeso.

Java Server Faces, também conhecido como JSF, é uma tecnologia para desenvolvimento web que utiliza um modelo de interfaces gráficas baseado em eventos. Esta tecnologia foi definida pelo JCP (Java Community Process), o que a torna um padrão de desenvolvimento e facilita o trabalho dos fornecedores de ferramentas, ao criarem produtos que valorizem a produtividade no desenvolvimento de interfaces visuais. ([FARIA, 2013](#), p. 51)

4.5.3 Primefaces

“PrimeFaces é uma bibliotecas de componentes ricos baseados em JavaServer Faces. A suíte de componentes inclui diversos campos de entrada, botões, tabelas de dados, árvores, gráficos, diálogos, etc.” ([FARIA, 2013](#), p. 180)

O *Primefaces* possui uma vasta quantidade de componentes otimizados para dispositivos móveis “*mobilekit*”, “*push framework*” e uma galeria de temas que podem ser baixados gratuitamente no site oficial, ou podem ser compradas nas versões *Pre-muim*.([PRIMEFACES, 2016](#))

A biblioteca conta com componentes de formulários como *autocomplete*, *textArea*, *Calendar*, *Sliders*, *InputMask*, componentes de apresentação de dados como *dataGrid*, *dataTable*, *treeTable*, *timeLine*, *mindMap*, componentes de mensagens como *Growl* e *Messages*, e outros como gráficos, menus, painéis, diálogos, componentes de *download* e *upload* de arquivos, entre outros. ([PRIMEFACES, 2016](#))

Ainda apresenta um kit de interface para dispositivos móveis. *PrimeFaces Mobile (PFM)* é um Kit de interface do usuário para criar aplicativos JSF otimizados para

dispositivos móveis. PFM é construído em cima do jQuery Mobile, um toque otimizado de *HTML5 UI* fornecendo suporte para várias plataformas.(PRIMEFACES, 2016)

Além da integração com *jQuery Mobile*, PFM apresenta uma *RenderKit Mobile* para componentes populares do *Primefaces*, *extensões framework Ajax*, eventos de comportamento *ajax* móveis, modelo de navegação integrado, o carregamento de páginas por demanda, *widgets* responsivos e muito mais.(PRIMEFACES, 2016)

4.5.4 Spring Security

”O Spring Security é um framework de controle de acesso, que nos permite definir de forma declarativa quem acessará o quê em nossos sistemas.” (WEISSMANN, 2012, p. 227)

O controle de acesso do Spring Security pode ser aplicado em dois níveis: nas requisições que chegam à nossa aplicação, caso estejamos lidando com um projeto web e na invocação de métodos dos *beans* gerenciados pelo contexto do Spring. Como veremos, não é acidentalmente que usei a palavra *aspecto* no parágrafo anterior: controlar o acesso é um interesse transversal que é resolvido pelo *framework* através do suporte a *AOP*. Mas antes da prática é necessário conhecermos os dois conceitos principais por trás do seu funcionamento.(WEISSMANN, 2012)

O *framework* é composto por uma série de componentes semi independentes que possibilitam ao programador usar apenas aqueles que serão úteis ao seu projeto.(WEISSMANN, 2012)

Segundo (WEISSMANN, 2012) a versão 3.1 é composta pelos módulos a seguir:

- ***spring-security-core***: o único módulo obrigatório por se tratar da essência do framework. Encontram-se aqui implementados os mecanismos de autorização e autenticação, além de todas as interfaces básicas que são reaproveitadas por todos os demais módulos e componentes terceirizados desenvolvidos para o framework;
- ***spring-security-config***: a partir da versão 2.0, o framework passou a oferecer um namespace próprio que reduziu dramaticamente a quantidade de configuração que o desenvolvedor precisava digitar. Dado o ganho de produtividade deve ser considerado obrigatório por todos os não masoquistas;
- ***spring-security-web***: toda a infraestrutura necessária para o desenvolvimento de projetos web. Neste módulo é que se encontra presente o código responsável por lidar com a interceptação de requisições a projetos web e portanto de uso obrigatório em projetos deste tipo;

- ***spring-security-taglibs***: contém uma biblioteca de tags que nos permite definir quais áreas de nossas páginas serão acessíveis aos usuários do sistema com base em suas permissões. Assim como o módulo *spring-security-config*, apesar de ser opcional acaba sendo usado sempre por todos aqueles que projetem suas páginas com alguma tecnologia baseada em JSP;
- ***spring-security-remoting***: provê 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 on CAS;
- ***spring-security-openid***: suporte a OpenID.

4.5.5 PostgreSQL

O PostgreSQL é um sistema poderoso banco de dados open source objeto-relacional. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura comprovada que ganhou uma forte reputação de confiabilidade, integridade de dados e correção. É executado em todos os principais sistemas operacionais, incluindo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e Windows. (POSTGRES, 2015).

É totalmente compatível com ACID, tem suporte completo para chaves estrangeiras, joins, views, triggers e procedimentos armazenados (em várias línguas). Ele inclui mais SQL: 2008 tipos de dados, incluindo *INTEGER*, *NUMERIC*, *BOOLEAN*, *CHAR*, *VARCHAR*, *DATE*, *INTERVALO* e *TIMESTAMP*. Também suporta o armazenamento de grandes objetos binários, incluindo imagens, sons ou vídeo. Possui interfaces de programação nativas para *C / C++*, *Java*, *.Net*, *Perl*, *Python*, *Ruby*, *Tcl*, *ODBC*, entre outros. (POSTGRES, 2015).

Um banco de dados de classe empresarial, o PostgreSQL possui características sofisticadas tais como Multi-Versão Controle de Concorrência (MVCC), ponto no tempo de recuperação, espaços de tabela, a replicação assíncrona, transações aninhadas (savepoints), on-line / backups quentes, um sofisticado planejador de consultas / otimizador, e escrever frente log para tolerância a falhas. (POSTGRES, 2015).

4.5.6 JPA

”A JPA nada mais é do que um conjunto de regras, normas e interfaces para definir um comportamento também conhecido como especificação.” (OLIVEIRA, 2013)

Segundo (FARIA, 2015) com JPA, não é necessário nada de especial para tornar os objetos persistentes. Bastando adicionar algumas anotações nas classes que representam

as entidades do sistema e começar a persistir ou consultar objetos. Para isso deve ser lembrado que a classe referida deve implementar a interface *java.io.serializable*.

Complementando, "JPA é uma especificação, e não um produto. Para trabalhar com JPA, precisamos de uma implementação"(FARIA, 2015). Isso significa que para utilizar os recursos da JPA é necessário que haja uma tecnologia que implemente suas especificações. Para este projeto implementação utilizada foi o *Hibernate*.

4.5.7 Hibernate

Segundo (FARIA, 2015) Hibernate é uma implementação da especificação JPA, ou seja, é um produto que implementa as funções requeridas pala JPA em um projeto Java.

O Hibernate Core é a base para o funcionamento da persistência, com APIs nativas e metadados de mapeamentos em arquivos XML, uma linguagem de consultas chamada HQL (parecido com SQL), um conjunto de interfaces para consultas usando critérios (Criteria API) (FARIA, 2015, p. 16)

Segundo a especificação JPA, as instâncias das entidades são administradas pelos *Entity Managers*. As duas principais responsabilidades dos *Entity Managers* são gerenciar o estado dos objetos e sincronizar os dados da aplicação e do banco de dados. (K19, 2015b)

Existem estado que uma Entidade pode passar dentro do contexto do *Entity Manager*. Uma instância de uma entidade pode passar pelos seguintes estados:(K19, 2015b)

- **Novo (*New*):** Um objeto nesse estado não possui uma identidade (chave) e não está associado a um *Entity Manager*. O conteúdo desse objeto não é enviado para o banco de dados. Toda instância de uma entidade que acabou de ser criada como comando *new* encontra-se no estado *new* do JPA.(K19, 2015b)
- **Administrado (*Managed*):** Um objeto no estado *managed* possui uma identidade e está associado a um *Entity Manager*. A cada sincronização, os dados de um objeto no estado *managed* são atualizados no banco de dados. Desvinculado (*Detached*): Umobjeto no estado *detached* possui uma identidade, mas não está associado a um *Entity Manager*. Dessa forma, o conteúdo desse objeto não é sincronizado com o banco de dados.(K19, 2015b)
- **Removido (*Removed*):** Um objeto no estado *removed* possui uma identidade e está associado a um *Entity Manager*. O conteúdo desse objeto será removido do banco de dados quando houver uma sincronização.(K19, 2015b)

Uma sincronização consiste em propagar para o banco de dados as modificações, remoções e inserções de entidades realizadas em memória através de um *Entity Manager*.(K19, 2015b)

Quando houver uma sincronização, as modificações realizadas no estado dos objetos *managed* são propagadas para o banco de dados, assim como os registros referentes aos objetos em estado *removed* são apagados do banco de dados. De acordo com a especificação, uma sincronização só pode ocorrer se uma transação estiver ativa.(K19, 2015b)

4.5.8 Twitter Bootstrap

Trata-se de um framework front-end que tem por objetivo aumentar a performance de desenvolvimento de interfaces e padronizar projetos web com a biblioteca javascript JQuery e CSS, estruturado em componentes que podem ser integrados com muitos tipos plataformas de desenvolvimento. Uma de suas características principais é proporcionar interfaces web responsivas, ou seja, que se adaptam a diferentes tamanhos de displays, como tablets, celulares e monitores desktop.

O Twitter Bootstrap trás em seu rol de componentes uma lista variada de itens que podem ser utilizadas em diversos tipos de projetos, componentes como menus, menus de navegação, botões *dropdown*, *breadcrumbs*, paginadores, thumbnails, alertas e popups, progress bars, media objects, entre outros. (GETBOOTSTRAP, 2016)

4.6 Ferramentas

4.6.1 Apache Tomcat

”Apache Tomcat é um software Opensource definido como um Container de Servlets ou container web, que implementa as tecnologias Java Servlet, Java Server Pages(JSP)Java, Java Expression Language e WebSocket technologies” (APACHE, 2016). O Apache Tomcat v.8.0 foi escolhido para ser o servidor de base na fase de desenvolvimento da aplicação.

4.6.2 Astah

Astah é um software utilizado na construção de modelos e diagramas UML. Com ele foram criados os Diagrama de Classes e Diagrama de Caso de Uso do sistema para visualização e compreensão das especificações a fim de orientar a fase de desenvolvimento. Na sua versão Professional podemos encontrar funções de exportação que auxiliam a gerar o código básico das classes utilizadas no projeto Java, diminuindo assim o tempo de codificação da aplicação.

O Astah Community é uma versão disponível para a comunidade de forma gratuita que conta com muitas ferramentas e funções para criação de diversos tipos de diagramas, como:

- Classes
- Casos de Uso
- Atividades
- Comunicação
- Estado
- Componentes
- Estrutura de Composição
- Desenvolvimento

A ferramenta Astah Community conta com plugins como PHP Export e Script que podem ser incorporados a sua instalação para utilização mais avançada de seus recursos. A ferramenta também conta com função de sugestão de alinhamento, personalização de atalhos, exportação de projetos em imagem jpg ou png, e impressão de diagramas. (ASTAH, 2016)

4.6.3 Ambiente de Desenvolvimento Netbeans

O NetBeans IDE é um ambiente de desenvolvimento, ou seja, uma ferramenta para programadores, que permite escrever, compilar, depurar e instalar programas. O IDE é completamente escrito em Java, mas pode suportar qualquer linguagem de programação. Existe também um grande número de módulos para estender as funcionalidades do IDE NetBeans. O NetBeans IDE é um produto livre, sem restrições à sua forma de utilização. (NETBEANS-IDE, 2016)

O Editor do NetBeans possui recursos avançados de edição de código como recua linhas, associa palavras e colchetes e realça códigos-fonte sintática e semanticamente. Ele também fornece modelos de código, dicas de codificação e ferramentas de refatoração. (NETBEANS-EDIT, 2016)

4.6.4 iText

iText 7 Core é o alicerce da iText 7. Ele permite que você monte, expanda, extraia, divida e interaja com qualquer arquivo PDF.([ITEXT, 2016](#))

É compatível com a maioria dos padrões PDF ISO, bem como normas para assinaturas digitais (*PAdES*) Ele permite que você utilize o seu tempo de forma mais produtiva, automatizando tarefas de documentação e arquivamento de rotina.([ITEXT, 2016](#))

Os documentos podem ser lidos por qualquer máquina, a qualquer momento, não importa os formatos de entrada originais, e vai ficar legível por um tempo muito longo. O núcleo do iText 7 serve ainda como a plataforma que pode ser expandida com add-ons para tarefas específicas PDF de valor agregado.([ITEXT, 2016](#))

4.6.5 JRebel

JRebel é um *plugin* para ambientes de desenvolvimento java com o objetivo de evitar o *redesploy* de uma aplicação para ver o impacto das alterações de código, economizando tempo e melhorando o desempenho de produção de um software java, ou seja, o desenvolvedor faz as alterações no código e o *JRebel* aplica essas alterações instantaneamente com a aplicação ainda em execução.([JREBEL, 2016](#))

5 Requisitos do Sistema

5.1 Requisitos Funcionais

”Requisitos funcionais. São as declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações” (PRESSMAN, 2011, p. 80). O *template* utilizado para descrever os requisitos deste sistema é baseado no modelo apresentado em (GUEDES, 2005).

Tabela 1 – Requisito Funcional - Autenticar Usuário

RF 01 - Autenticar Usuário			
Descrição: O sistema deve autenticar o usuário no sistema mediante a informação de login e senha.			
Ator: Sistema			
Prioridade: obrigatório			
Entradas e pré-condições: nenhum			
Requisitos Não Funcionais			
Nome	Restrição	Categoria	Prioridade
NF 1.1 Tipos de Usuários	Essa função deverá autenticar os seguintes tipos de usuário: Administrador, Secretário, Líder de Célula e Supervisor de Setor	Segurança	Obrigatório

Tabela 2 – Requisito Funcional - Manter Membros

RF 02 - Manter Membros	
Descrição: Realizar o cadastro com todas as informações existentes do membro, tais como nome, endereço, telefone, e-mail, dia e mês de aniversário e estado civil. Realizar consulta , alteração, exclusão e inativação de membros.	
Ator: Secretário, Administrador e Lder	
Prioridade: obrigatório	
Entradas e pré-condições: RF 01	
Regras de Negócio	
Nome	Descrição
RN 2.1 Identificação do Membro	Os membros serão identificados por um número sequencial (id) gerado no momento do cadastro.
RN 2.1 Exclusão	A exclusão do membro será possível apenas se não houver nenhum registro de atividades do mesmo no sistema.

Tabela 3 – Requisito Funcional - Manter Células

RF 03 - Manter Células

Descrição: O sistema deverá permitir que sejam cadastradas e alteradas Células com os seguintes atributos: nome da célula, nome do líder, endereço, dia de reunião, perfil e status (Ativada ou desativada).

Ator: Secretário e Administrador

Prioridade: obrigatório

Entradas e pré-condições: RF 01 e RF 02

Requisitos Não Funcionais

Nome	Restrição
RF 3.1 Líder de Célula	Ao cadastrar ou alterar uma Célula deve ser selecionado um membro previamente cadastrado no sistema para ser o seu líder.
RF 3.2 Perfil	Toda célula deve ser cadastrada com um perfil que representa o seu público alvo. Os perfis podem ser: Mista, Homens, Mulheres, Crianças, Pré-Adolescentes, Adolescentes, Jovens, Casados, Supervisores

Tabela 4 – Requisito Funcional - Manter Setores

RF 04 - Manter Setores

Descrição: Cadastrar e alterar informações referentes aos Setores com os seguintes atributos: nome do supervisor, nomes das células que o compõem e status (Ativado ou desativado).

Ator: Secretário e Administrador

Prioridade: obrigatório

Entradas e pré-condições: RF 01, RF 02 e RF 03

Requisitos Vinculados

Nome	Descrição
RF 4.1 Estrutura de Setor	Cada setor é composto por um grupo de células que devem ser selecionadas no momento do cadastro ou alteração do setor
RF 4.2 Supervisor de Setor	Ao cadastrar ou alterar uma Setor deve ser selecionado um líder de célula previamente cadastrado no sistema para ser o seu Supervisor.

Tabela 5 – Requisito Funcional - Manter Reunião de Célula

RF 05 - Reunião de Célula

Descrição: O sistema deverá disponibilizar para cada Líder de célula um formulário referente a reunião realizada na referida semana. Tal formulário deverá conter os seguintes campos: nome da célula, nome do líder, data da reunião, número de convidados, número de grupos de evangelismo, número de discípulos, nomes dos membros presentes na reunião, valor das ofertas e observação do líder.

Ator: Administrador, Secretário e Líder

Prioridade: obrigatório

Entradas e pré-condições: RF 01, RF 02 e RF 03

Requisitos Não Funcionais

Nome	Descrição
RF 5.1 Lista de Presença	A função deverá exibir uma lista com os nomes dos membros da Célula para o líder informar se o membro esteve Presente, Ausente ou se sua falta foi Justificada

Tabela 6 – Requisito Funcional - Manter Reunião de Treinamento de Lideres (Tadel)

RF 06 - Manter Tadel

Descrição: Cadastrar, alterar, remover e consultar Informações referentes a reunião de Treinamento de Lideres (Tadel), com as seguintes informações: Data da reunião e observação.

Ator: Administrador, Secretário

Prioridade: obrigatório

Entradas e pré-condições: RF 01

Requisitos Vinculados

Nome	Descrição
RF 6.1 Exclusão de Reunião	A função deverá permitir a remoção de Reuniões independentemente das presenças vinculadas a ela. Todas as presenças deve ser automaticamente excluídas ao excluir uma reunião

Tabela 7 – Requisito Funcional - Manter Culto de Celebração

RF 07 - Manter Culto de Celebração

Descrição: Cadastrar, alterar, remover e consultar Informações referentes ao Culto de Celebração, com as seguintes informações: Data do Culto e observação.

Ator: Administrador, Secretário

Prioridade: obrigatório

Entradas e pré-condições: RF 01

Requisitos Vinculados

Nome	Descrição
RF 7.1 Exclusão de Culto	Todas as presenças deve ser automaticamente excluídas ao excluir uma reunião.

Tabela 8 – Requisito Funcional - Consultar Relatório de Célula

RF 08 - Consultar Relatório de Célula

Descrição: Disponibilizar relatório referente as reuniões semanais da célula, Tadel e Culto de Celebração.

Ator: Administrador, Secretário, Líder e Supervisor de Setor

Prioridade: obrigatório

Entradas e pré-condições: RF 01 e RF 05

Requisitos Vinculados

Nome	Descrição
RF 8.1 Consulta ao Relatório	A função deverá solicitar que o usuário informe o Mês para que seja gerado o seu respectivo relatório.
RF 8.2 Organização Semanal	Exibir uma lista do mês referente organizado por semanas, informando em cada item(semmana) informações das reuniões de célula, Tadel e Culto de Celebração que ocorreram
RF 8.3 Presença em Reunião de Célula	Exibir a porcentagem de presença nas reuniões de célula
RF 8.4 Quantidades de presentes	A função deverá exibir a quantidade de presentes nas reuniões de célula, Tadel, e culto em cada semana do mês selecionado.
RF 8.5 Reunião de Célula	Exibir o número de convidados, Vínculos de Discipulado e valor da oferta arrecadada em cada semana do mês selecionado.

Tabela 9 – Requisito Funcional - Manter Discipulado

RF 09 - Manter Discipulado

Descrição: Cadastrar, alterar, remover e consultar Informações referentes aos vínculos e reuniões de discipulado. Os vínculos devem ter as seguintes informações: Nome do discipulador, nome do discípulo, dia da semana e hora da reunião. As reuniões devem ser cadastradas apenas com a data em que ocorrerem.

Ator: Líder

Prioridade: obrigatório

Entradas e pré-condições: RF 01 e RF 02

Regras de Negócio

Nome	Descrição
RN 9.1 Lista de Presença	Após a inserção de um vínculo de discipulado, o mesmo não pode mas ser excluído mas apenas inativado

5.2 Requisitos Não Funcionais

”Requisitos não funcionais. São restrições sobre os serviços ou funções oferecidas pelo sistema. Eles incluem restrições de *timing*, restrições sobre os processos de desenvolvimento e padrões” (SOMMERVILLE, 2007, p. 80)

Tabela 10 – Requisitos Não Funcionais

ID	Descrição	Categoria	Prioridade
RNF 01	O sistema deverá estar disponível na internet para ser acessado de qualquer lugar através da web	Disponibilidade	Obrigatório
RNF 02	O sistema deverá ter uma interface intuitiva, amigável e objetiva	Usabilidade	Obrigatório
RNF 03	A aplicação deverá ser construída com a linguagem de programação Java	Implementação	Obrigatório
RNF 04	O sistema deverá ser acessado por qualquer sistema operacional	Portabilidade	Desejável
RNF 05	O sistema deverá ser apresentado de forma adequada em dispositivos móveis	Mobilidade	Obrigatório
RNF 06	O usuário que ficar sem atividades no sistema durante cinco minutos deverá ser desconectado	Segurança	Desejável

5.3 Diagrama de Casos de Uso

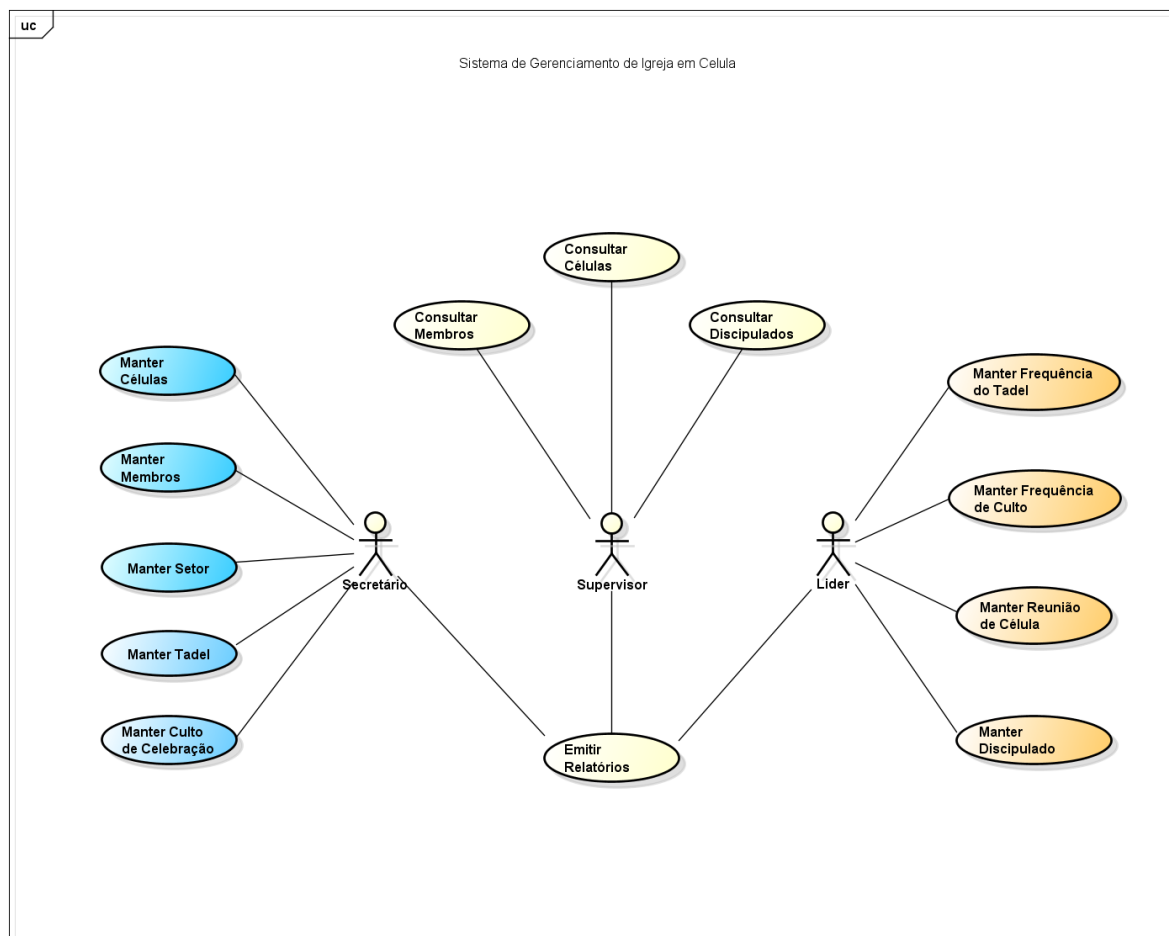


Figura 5 – Diagrama de Casos de Uso

5.4 Detalhamento dos Casos de Uso

”A documentação de um Caso de Uso costuma descrever por meio de uma linguagem bastante simples, a função em linhas gerais do Caso de Uso, destacando quais Atores interagem com o mesmo, quais etapas devem ser executadas pelo Ator e pelo sistema”. (GUEDES, 2005, p. 28)

O *template* utilizado para descrever os Casos de Uso deste sistema é baseado no modelo apresentado em (GUEDES, 2005).

Tabela 11 – Caso de Uso - Manter Membros

UC01 - MANTER MEMBROS	
Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta, exclusão e inativação de membros do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)	
Ator: Secretário	
Entradas e pré-condições: Usuário Autenticado (UC01)	
F01 - CADASTRAR MEMBRO	
Fluxo Principal	
Passo	Descrição
01	Usuário acessa o menu principal e seleciona inserção de novo membro
02	O sistema exibe um formulário com os campos de informações do membro para serem preenchidos como: nome, data de aniversário, sexo, estado civil, telefone, e-mail, cidade, bairro, cep, logradouro, número e complemento
03	Usuário preenche os campos e confirma o cadastro
04	O sistema valida as informações preenchidas
05	O sistema exibe mensagem de membro cadastrado com sucesso
06	Fim do fluxo.
Fluxos Alternativos	
FA01 - Não preencheu ou não selecionou campos Obrigatórios	
Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)
FA02 - Campo "Aniversário"preenchido incorretamente	
Passo	Descrição
01	O sistema exibe uma mensagem informando ao usuário que o campo está incorreto exibindo um exemplo da forma correta de preenchimento.
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR MEMBRO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o Caso de Uso UC09 (Consultar Membros)
02	O sistema exibe uma página de detalhes do membro que se deseja alterar.
03	Usuário seleciona a opção "alterar"
04	O sistema exibe um formulário com todos os campos presentes no fluxo (F01 - Cadastrar Membro)
05	O usuário preenche o campo que deseja alterar e confirma
06	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Usuário não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F02 passo 04)

FA02 - Campo "Aniversário" preenchido incorretamente

Passo	Descrição
01	O sistema exibe uma mensagem informando ao usuário que o campo está incorreto exibindo um exemplo da forma correta de preenchimento.
02	Retorna ao fluxo principal (F02 passo 04)

F03 - REMOVER MEMBRO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o Caso de Uso UC09 (Consultar Membros)
02	O sistema exibe uma página de detalhes do membro que se deseja remover, e um menu de opções.
03	Usuário seleciona a opção de remover membro
04	O sistema exibe um diálogo perguntando se o usuário deseja realmente remover o membro
05	O usuário confirma remoção
06	O sistema exibe uma mensagem informando ao usuário que o membro foi removido com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Membro possui registros de atividades no sistema**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível remover o membro por existirem registros que devem ser mantidos no histórico do sistema
02	Retorna ao fluxo principal (F03 passo 02)

F04 - INATIVAR MEMBRO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F02 - Alterar Membros)
02	O sistema exibe uma página de edição do membro.
03	Usuário marca/desmarca o item "Ativo" depois confirma
04	O sistema exibe um diálogo perguntando se deseja continuar, informando que o membro será inativado se continuar
05	O usuário confirma ação
06	O sistema exibe uma mensagem informando ao usuário que o membro foi inativado.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Membro possui discipulado ativo**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível desativar o membro por ter o registro de discipulado ativo no seu perfil
02	Retorna ao fluxo principal

FA02 - Membro está em uma célula

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível inativar o membro por ter o seu registro vinculado a uma célula
02	Retorna ao fluxo principal

FA03 - Membro é líder de célula

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível inativar o membro por ter o seu registro vinculado a uma célula como líder
02	Retorna ao fluxo principal

FA04 - Membro é supervisor de Setor

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível inativar o membro por ter o seu registro vinculado a um Setor como Supervisor
02	Retorna ao fluxo principal

Tabela 12 – Caso de Uso - Manter Células

UC02 - MANTER CÉLULAS

Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta, exclusão e inativação de células do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

F01 - CADASTRAR CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário acessa o menu principal e seleciona a opção de cadastrar nova célula
02	O sistema exibe um formulário com os campos de informações da célula para serem preenchidos como: nome, líder, perfil, setor, dia de reunião, cidade, cep, bairro, logradouro, número e complemento
03	Usuário preenche os campos e confirma o cadastro
04	O sistema valida as informações preenchidas
05	O sistema exibe mensagem de célula cadastrada com sucesso
06	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário executa o Caso de Uso UC10 (Consultar Células).
02	O sistema exibe uma página de detalhes da célula que se deseja alterar, e nela, um menu de ações.
03	Usuário seleciona a opção alterar
04	O sistema exibe um formulário com todos os campos presentes no fluxo (F01 - Cadastrar Célula)
05	O usuário preenche o campo que deseja alterar e confirma edição
06	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Usuário não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal

F03 - REMOVER CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário executa o Caso de Uso UC10 (Consultar Células)
02	O sistema exibe uma página de detalhes da célula que se deseja remover, e um menu de ações.
03	Usuário seleciona a opção de remover célula
04	O sistema exibe um diálogo perguntando se o usuário deseja realmente remover a célula
05	O usuário confirma remoção
06	O sistema exibe uma mensagem informando ao usuário que a célula foi removido com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - A célula possui registros de atividades no sistema, ex: reuniões com presença de membros**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível remover a célula por existirem registros que devem ser mantidos no histórico do sistema
02	Retorna ao fluxo principal

F04 - INATIVAR CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário executa o Caso de Uso UC10 (Consultar Células)
02	O sistema exibe uma página de detalhes da célula.
03	Usuário seleciona a opção desativar
04	O sistema exibe um diálogo perguntando se deseja continuar, informando que a célula será inativada se continuar
05	O usuário confirma desativação
06	O sistema exibe uma mensagem informando ao usuário que a célula foi desativada com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Célula possui membros**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível desativar a célula por ter o registro de membros vinculados a ela, pedindo que remova os membros antes de inativá-la
02	Retorna ao fluxo principal

FA02 - A célula pertence a um Setor

Passo	Descrição
01	O sistema exibe uma mensagem informando que não é possível inativar a célula por ter o seu registro vinculado a um Setor
02	Retorna ao fluxo principal

Tabela 13 – Caso de Uso - Manter REUNIÃO DE CÉLULA

UC03 - MANTER REUNIÃO DE CÉLULA

Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta e exclusão de reuniões de células do sistema. Tem início com o usuário acessando a página de detalhamento da célula

Ator: Líder de Célula

Entradas e pré-condições: Usuário Autenticado (UC01) e Células Cadastradas (UC01 - Fluxo F01)

F01 - CADASTRAR REUNIÃO DE CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona opção de nova reunião
02	O sistema exibe uma página contendo um formulário com os campos de cadastro de reunião como data, oferta, número de visitantes número de membros compromissados e uma lista de membros da célula para que o usuário marque a presença dos membros na referida reunião
03	O sistema exibe uma lista contendo as reuniões que a referida célula já realizou, contendo os itens cadastrados no fluxo f01
04	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenchê-lo para concluir o cadastro

FA02 - Campo "Data"preenchido incorretamente

Passo	Descrição
01	O sistema exibe uma mensagem informando ao usuário que o campo está incorreto exibindo um exemplo da forma correta de preenchimento.
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR REUNIÃO DE CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Reuniões de Célula)
02	O sistema exibe a lista de reuniões que a célula realizou.
03	Usuário seleciona a opção "editar" no registro que deseja alterar
04	O sistema exibe um formulário com todos os campos presentes no fluxo (F01 - Cadastrar Reunião de Célula)
05	O usuário preenche o campo que deseja alterar e confirma alteração
06	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Usuário não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal no passo 05

FA02 - Campo "Data" preenchido incorretamente

Passo	Descrição
01	O sistema exibe uma mensagem informando ao usuário que o campo está incorreto exibindo um exemplo da forma correta de preenchimento.
02	Retorna ao fluxo principal no passo 05

F03 - CONSULTAR REUNIÕES DE CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona a opção "reuniões"
02	O sistema exibe uma lista contendo as reuniões que a referida célula já realizou.
03	Fim do fluxo.

Fluxos Alternativos**FA01 - Nenhuma reunião cadastrada**

Passo	Descrição
01	O sistema exibe uma mensagem ao usuário informando que não há reuniões registradas para a referida célula

F04 - REMOVER REUNIÃO DE CÉLULA**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar reunião de Célula)
02	O sistema exibe a lista de reuniões que a célula realizou.
03	Usuário seleciona a opção "Excluir"
04	O sistema exibe um diálogo perguntando se o usuário deseja realmente remover a reunião
05	O usuário confirma exclusão
06	O sistema exibe uma mensagem informando ao usuário que a reunião foi removida com sucesso.
07	Fim do fluxo.

Fluxos Alternativos

* Sem fluxos alternativos

Tabela 14 – Caso de Uso - Manter DISCIPULADO

UC04 - MANTER DISCIPULADO

Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta, exclusão e inativação de discipulado do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

F01 - CADASTRAR DISCIPULADO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Membros)"
02	O sistema exibe uma página de detalhes do membro que se deseja alterar, e um menu de ações.
03	Usuário seleciona a opção "Discipulado"
04	O sistema exibe uma página de detalhamento de discipulado, apresentando o referido membro como "Discípulo"
05	o usuário seleciona a opção "Editar Discipulado"
06	o Sistema exibe um formulário com o campo de seleção do discipulador, o campo de seleção do dia da semana em que as reuniões acontecerão e um campo para registrar a hora em que serão realizadas as reuniões.
07	O usuário preenche corretamente os campos e confirma.
08	O sistema informa que as informações foram salvas com sucesso.
09	O sistema atualiza as informações da tela de detalhamento e exibe as informações do novo discipulador
10	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR DISCIPULADO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Discipulado)
02	O sistema exibe uma página de detalhes do discipulado que se deseja alterar.
03	Usuário seleciona a opção "Editar Discipulado"
04	O sistema exibe um formulário com todos os campos presentes no fluxo (F01 - Cadastrar Discipulado)
05	O usuário preenche o campo que deseja alterar e confirma cadastro
06	O sistema informando ao usuário que as informações foram salvas com sucesso
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Usuário não preencheu ou não selecionou campos Obrigatórios**

Passo	Descrição
01	O sistema exibe uma mensagem para cada campo obrigatório informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F02 passo 04)

F04 - INATIVAR DISCIPULADO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Discipulado)
02	O sistema exibe uma página de detalhamento do discipulado.
03	Usuário seleciona a opção "Desativar Discipulado"
04	O sistema exibe um diálogo perguntando se deseja continuar, informando que o discipulado será desativado se continuar
05	O usuário confirma desativação
06	O sistema exibe uma mensagem informando ao usuário que o discipulado foi inativado.
07	Fim do fluxo.

Tabela 15 – Caso de Uso - MANTER TADEL

UC05 - MANTER TADEL

Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta e exclusão da reunião de Treinamento Avançado de Lideres "Tadel". Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

F01 - CADASTRAR TADEL**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona no menu principal o item "Tadel"
02	O sistema exibe uma lista com todas as reuniões inseridas.
03	Usuário seleciona "Nova Reunião"
04	O sistema exibe uma página contendo um campo para informar a data em que a reunião ocorreu e um campo de observação
05	O usuário preenche corretamente os campos e confirma.
06	O sistema exibe uma mensagem informando que as informações foram salvas com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu o campo "Data"**

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá preenchê-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

FA02 - Preencheu incorretamente o campo "Data"

Passo	Descrição
01	O sistema exibe uma mensagem informando que o campo foi preenchido incorretamente e exibirá um exemplo de como preenchê-lo de forma válida.
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR TADEL

Fluxo Principal

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Tadel)
02	o Usuário seleciona a opção "Editar" no registro da reunião desejada.
03	Usuário preenche os campos que deseja editar e confirma edição
04	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
05	Fim do fluxo.

Fluxos Alternativos

FA01 - Não preencheu o campo "Data"

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá preenchê-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

FA02 - Preencheu incorretamente o campo "Data"

Passo	Descrição
01	O sistema exibe uma mensagem informando que o campo foi preenchido incorretamente e exibirá um exemplo de como preenchê-lo de forma válida.
02	Retorna ao fluxo principal (F01 passo 04)

F03 - CONSULTAR TADEL**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona no menu principal a opção "Tadel").
02	O sistema exibe uma lista com todas as reuniões cadastradas.
05	Fim do fluxo.

Fluxos Alternativos**FA01 - Nenhuma reunião cadastrada**

Passo	Descrição
01	O sistema exibe uma mensagem informando que ainda não foi registrado nenhuma reunião.
02	Fim do fluxo.

F04 - REMOVER TADEL**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Tadel)
02	O sistema exibe a lista de reuniões realizadas.
03	Usuário seleciona a opção "Excluir" no registro da reunião desejada
04	O sistema exibe um diálogo perguntando se o usuário deseja realmente remover a reunião
05	O usuário confirma a remoção
06	O sistema exibe uma mensagem informando ao usuário que a reunião foi removida com sucesso.
07	Fim do fluxo.

Fluxos Alternativos

*** Sem fluxos alternativos**

Tabela 16 – Caso de Uso - MANTER CULTO DE CELEBRAÇÃO

UC06 - MANTER CULTO DE CELEBRAÇÃO

Descrição: Este caso de uso tem como finalidade realizar o cadastro, alteração, consulta e exclusão dos Cultos de Celebração. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

F01 - CADASTRAR CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona no menu principal o item "Culto de Celebração"
02	O sistema exibe uma lista com todas os cultos inseridos.
03	Usuário seleciona a opção "Novo Culto"
04	O sistema exibe uma página contendo um campo para informar a data em que o culto ocorreu e um campo de observação
05	O usuário preenche corretamente os campos e confirma.
06	O sistema exibe uma mensagem informando que as informações foram salvas com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu o campo "Data"**

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

FA01 - Não preencheu incorretamente o campo "Data"

Passo	Descrição
01	O sistema exibe uma mensagem informando que o campo foi preenchido incorretamente e exibirá um exemplo de como preenche-lo de forma válida.
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Culto de Celebração)
02	o Usuário seleciona a opção "Editar" no registro do culto desejada.
03	Usuário preenche os campos que deseja editar e confirma
04	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
05	Fim do fluxo.

Fluxos Alternativos**FA01 - Não preencheu o campo "Data"**

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá preenche-lo para concluir o cadastro
02	Retorna ao fluxo principal (F01 passo 04)

FA02 - Preencheu incorretamente o campo "Data"

Passo	Descrição
01	O sistema exibe uma mensagem informando que o campo foi preenchido incorretamente e exibirá um exemplo de como preenche-lo de forma válida.
02	Retorna ao fluxo principal (F01 passo 04)

F03 - CONSULTAR CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	Usuário seleciona no menu principal a opção "Culto de Celebração".
02	O sistema exibe uma lista com todos os cultos cadastrados.
05	Fim do fluxo.

Fluxos Alternativos**FA01 - Nenhum culto cadastrado**

Passo	Descrição
01	O sistema exibe uma mensagem informando que ainda não foi registrado nenhum culto.
02	Fim do fluxo.

F04 - REMOVER CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	Usuário executa o fluxo (F03 - Consultar Culto de Celebração)
02	O sistema exibe a lista de cultos realizados.
03	Usuário seleciona a opção "Excluir" no registro do culto desejado
04	O sistema exibe um diálogo perguntando se o usuário deseja realmente remover o culto)
05	O usuário confirma remoção
06	O sistema exibe uma mensagem informando ao usuário que o culto foi removido com sucesso.
07	Fim do fluxo.

Fluxos Alternativos

*** Sem fluxos alternativos**

Tabela 17 – Caso de Uso - MANter FREQUêNCIA DO CULTO DE CELEBRAÇÃO

UC07 - MANter FREQUêNCIA DO CULTO DE CELEBRAÇÃO

Descrição: Este caso de uso tem como finalidade realizar a inserção, atualização, consulta e exclusão dos registros de frequência dos membros no Culto de Celebração. Tem início com o usuário acessando a página de detalhamento da célula

Ator: Líder de Célula

Entradas e pré-condições: Usuário Autenticado (UC01), culto cadastrado (UC06), Célula cadastrada (UC02)

F01 - INSERIR FREQUENCIA EM CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	O líder seleciona o item "Culto de Celebração"
02	O sistema exibe uma lista com todos os cultos cadastrados.
03	O líder seleciona o culto que deseja inserir frequência
04	O sistema exibe uma página contendo uma lista de frequência dos membros da célula
05	O líder seleciona o tipo de frequência para cada membro: Presente, Justificado, Faltou. e confirma
06	O sistema exibe uma mensagem informando que as informações foram salvas com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Não selecionou o tipo de presença para algum membro**

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá selecionar todos os itens para concluir
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR FREQUÊNCIA DE CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	O líder executa o fluxo (F03 - Consultar Frequência de Culto de Celebração)
02	o líder seleciona "Marcar Presença" no registro do culto desejado.
03	O líder modifica o tipo de frequência que deseja alterar e confirma
04	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
05	Fim do fluxo.

*** Sem fluxos alternativos**

F03 - CONSULTAR FREQUÊNCIA DE CULTO DE CELEBRAÇÃO**Fluxo Principal**

Passo	Descrição
01	O líder seleciona a opção "Culto de Celebração").
02	O sistema exibe uma lista com todos OS cultos cadastrados.
03	O líder seleciona o culto que deseja consultar.
04	O sistema exibe a lista de frequência da célula para aquele culto.
05	Fim do fluxo.

*** Sem fluxos alternativos**

Tabela 18 – Caso de Uso - MANTER FREQUÊNCIA DO TADEL

UC08 - MANTER FREQUÊNCIA DO TADEL

Descrição: Este caso de uso tem como finalidade realizar a inserção, atualização, consulta e exclusão dos registros de frequência dos membros no Tadel. Tem início com o usuário acessando a página de detalhamento da célula)

Ator: Líder de Célula

Entradas e pré-condições: Usuário Autenticado (UC01), Tadel cadastrado (UC05), Célula cadastrada (UC02)

F01 - INSERIR FREQUENCIA EM TADEL**Fluxo Principal**

Passo	Descrição
01	O líder seleciona o item "Tadel"
02	O sistema exibe uma lista com todas as reuniões cadastradas.
03	O líder seleciona a reunião que deseja inserir frequência
04	O sistema exibe uma página contendo uma lista de frequência dos membros da célula
05	O líder seleciona o tipo de frequência para cada membro: Presente, Justificado, Faltou. e confirma
06	O sistema exibe uma mensagem informando que as informações foram salvas com sucesso.
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Não selecionou o tipo de presença para algum membro**

Passo	Descrição
01	O sistema exibe uma mensagem informando que o usuário deverá selecionar todos os itens para concluir
02	Retorna ao fluxo principal (F01 passo 04)

F02 - ALTERAR FREQUÊNCIA DE TADEL**Fluxo Principal**

Passo	Descrição
01	O líder executa o fluxo (F03 - Consultar Frequência de Tadel)
02	o líder seleciona "Marcar Presença" no registro da reunião desejada.
03	O líder modifica o tipo de frequência que deseja alterar e confirma
04	O sistema exibe uma mensagem informando ao usuário que as informações foram salvas com sucesso
05	Fim do fluxo.

*** Sem fluxos alternativos**

F03 - CONSULTAR FREQUÊNCIA DE TADEL**Fluxo Principal**

Passo	Descrição
01	O líder seleciona a opção "Tadel").
02	O sistema exibe uma lista com todas as reuniões cadastrados.
03	O líder seleciona a reunião que deseja consultar.
04	O sistema exibe a lista de frequência da célula para aquela reunião.
05	Fim do fluxo.

*** Sem fluxos alternativos**

Tabela 19 – Caso de Uso - Consultar Membros

UC09 - CONSULTAR MEMBROS

Descrição: Este caso de uso tem como finalidade realizar o consulta de membros do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Prioridade: obrigatório

Entradas e pré-condições: Usuário Autenticado (UC01)

FLUXO PRINCIPAL

Passo	Descrição
01	Usuário acessa o menu principal e seleciona a a opção de listar membros.
02	O sistema exibe uma lista com os membros cadastrados no sistema e campos de busca, com opções de busca por nome e/ou célula.
03	Usuário insere os dados requeridos para a busca.
04	O sistema exibe uma lista de membros cujas suas informações correspondam com as informações inseridas.
05	o usuário seleciona o membro desejado.
06	o sistema redireciona o usuário para a página de detalhes do membro
07	Fim do fluxo.

Fluxos Alternativos**FA01 - Usuário preencheu informações que não correspondam com nomes de membros ou células**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não há membros que correspondam a essa busca
02	Retorna ao fluxo principal

Tabela 20 – Caso de Uso - Consultar Células

UC10 - CONSULTAR CÉLULAS

Descrição: Este caso de uso tem como finalidade realizar consulta de células do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

FLUXO PRINCIPAL

Passo	Descrição
01	Usuário acessa o menu principal e seleciona a opção de listar células.
02	O sistema exibe uma lista com as células cadastrados no sistema.
03	o usuário seleciona a célula desejada.
04	o sistema redireciona o usuário para a página de detalhes da célula, onde exibirá todas as informações cadastradas no Caso de Uso UC01 Fluxo F01 (cadastrar célula).
05	Fim do fluxo.

Fluxos Alternativos**FA01 - Não há células cadastradas**

Passo	Descrição
01	O sistema exibe uma mensagem informando que não há células cadastradas no sistema
02	Retorna ao fluxo principal

Tabela 21 – Caso de Uso - Consultar Discipulado

UC12 - CONSULTAR DISCIPULADO

Descrição: Este caso de uso tem como finalidade realizar consulta de discipulado do sistema. Tem início com o usuário acessando a página inicial do sistema (Dashboard)

Ator: Secretário

Entradas e pré-condições: Usuário Autenticado (UC01)

FLUXO PRINCIPAL

Passo	Descrição
01	Usuário executa o caso de uso UC09 (Consultar membro).
02	O sistema exibe a página de detalhamento do membro.
03	Usuário seleciona a opção "Discipulado"
04	O sistema exibe a página de detalhamento do discipulado contendo as informações cadastradas no Caso de Uso UC05 fluxo f01 (Cadastrar Discipulado).
05	Fim do fluxo.

Fluxos Alternativos**FA01 - Segunda forma de consultar**

Passo	Descrição
01	O usuário seleciona no menu principal o item "Discipulados"
02	O sistema exibe uma lista com todos os discipulados ativos no sistema, com três campos de busca no topo da página, dando a possibilidade do usuário buscar discipulados pelo nome do discípulo, nome do discipulador ou dia da semana em que acontecem as reuniões.
03	O usuário pode usar os campos de busca e selecionar um discipulado
04	O sistema exibe a página de detalhamento do discipulado contendo as informações cadastradas no Caso de Uso UC05 fluxo f01 (Cadastrar Discipulado).
05	Fim do fluxo.

Tabela 22 – Caso de Uso - Emitir Relatório

UC13 - EMITIR RELATÓRIO

Descrição: Este caso de uso tem como finalidade emitir relatório de célula. Tem início com o usuário acessando a página de detalhamento da célula

Ator: Secretário, Líder e Supervisor de Setor

Entradas e pré-condições: Usuário Autenticado (UC01)

FLUXO PRINCIPAL

Passo	Descrição
01	Usuário seleciona a opção "Relatório".
02	O sistema exibe a página de emissão de relatório de célula
03	Usuário seleciona o mês referente ao relatório desejado
04	O sistema trás as informações pertinentes ao mês selecionado, como quantidade de membros presentes das reuniões de <i>célula</i> , <i>Tadel</i> e <i>culto</i> , número de convidados, discipulados e valor da oferta da célula
05	Usuário seleciona a opção "Salvar em PDF".
06	O sistema exibe um diálogo de salvamento de arquivo.
07	O usuário confirma o download do arquivo PDF
08	Fim do fluxo.

FA02 - Campo de seleção do mês preenchido incorretamente

Passo	Descrição
01	O sistema exibe uma mensagem informando ao usuário que o campo está incorreto exibindo um exemplo da forma correta de preenchimento.
02	Retorna ao fluxo principal

5.5 Diagrama de Classes

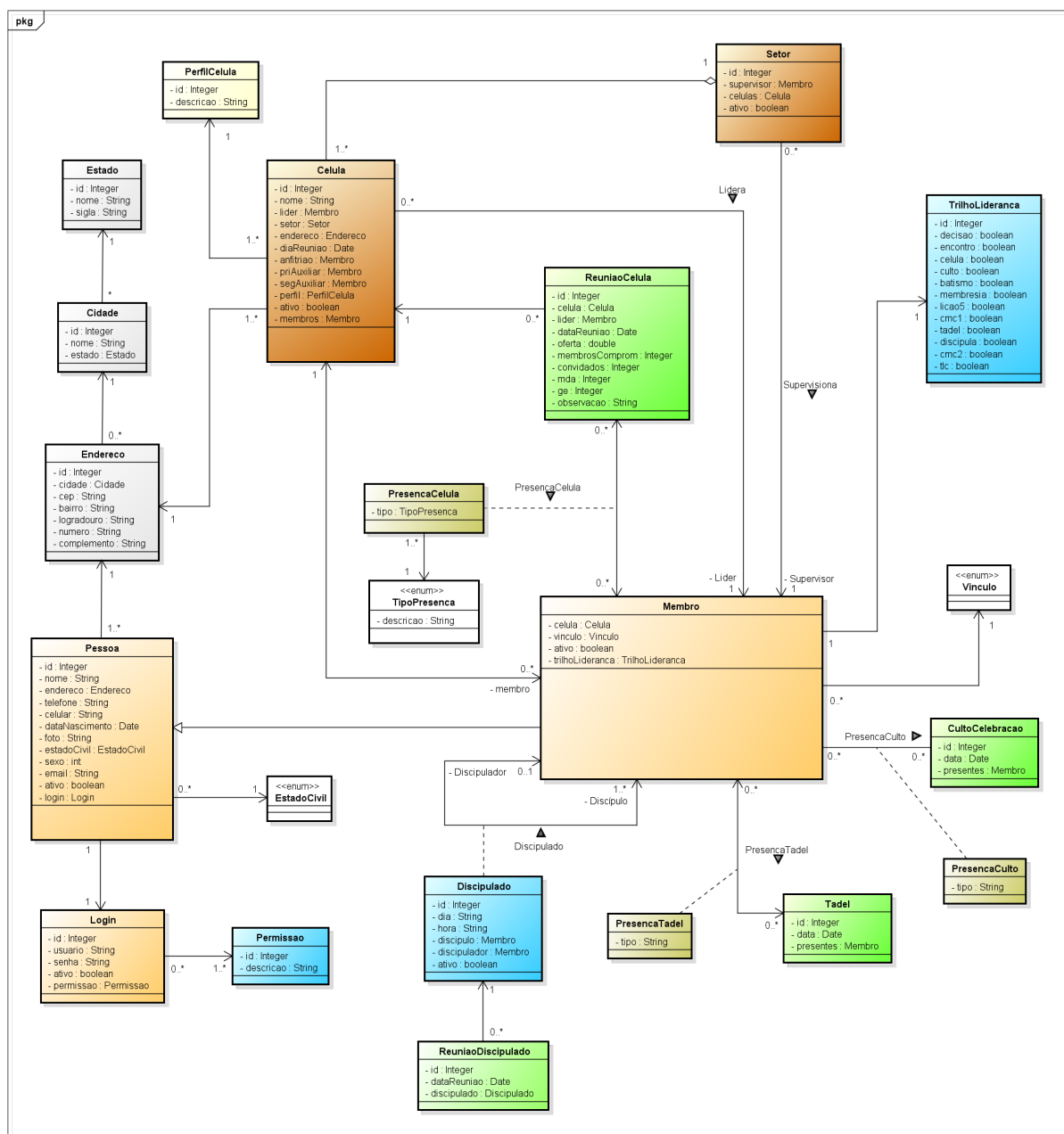


Figura 6 – Diagrama de Classes

5.5.1 Descrição do Diagrama de Classes

Neste Diagrama de Classes temos a classe *Célula* que representa um grupo de membros e a classe *Setor* que representa um grupo de Células. A classe *Membro* é uma generalização da classe *Pessoa* que contém as informações básicas de cadastro de uma pessoa no sistema. A classe *Pessoa* pode ter um *Login* com permissões que determinam que tipo de ator essa pessoa representa.

A classe *Discipulado* mantém as informações de um auto-relacionamento da classe *Membro* onde um membro é o discipulador e o outro é o discípulo, a classe também mantém informações de dia e hora das reuniões de discipulado. As classes *Tadel* e *CultoCelebracao* são reuniões semanais onde são marcadas as presenças dos membros.

A classe *ReuniaoCelula* é a principal classe do sistema, nela estão representadas as informações das reuniões de célula e dela são extraídas as principais informações do relatório de Célula. A classe *TrilhoLideranca* representa os requisitos para um membro se tornar apto a ser líder de célula.

6 Desenvolvimento

6.1 Implementando Padrão MVC

Como foi dito no capítulo de Aporte teórico, o MVC é um padrão de arquitetura de projetos que divide o projeto em *Model* (Modelo), *View* (Visão) e *Controller* (Controlador).

6.1.1 Model

O *Model* desse projeto está representado pelas classes de domínio. Um exemplo de uma classe de domínio é a classe *Celula* a seguir apresentada:

```

1 @Entity
2 public class Celula implements Serializable {
3     @Id
4     @GeneratedValue
5     private Integer id;
6     @Column(length = 255, nullable = true, unique = false)
7     private String nome;
8     @ManyToOne(optional = false, cascade = CascadeType.MERGE)
9     private Membro lider;
10    @OneToOne(optional = true)
11    private Membro anfitriao;
12    @OneToOne(optional = true)
13    private Membro priAuxiliar;
14    @ManyToOne(cascade = CascadeType.ALL, optional = true)
15    private Endereco endereco = new Endereco();
16    @Enumerated(EnumType.STRING)
17    @Column(nullable = false)
18    private DiaSemana diaReuniao;
19    @ManyToOne(optional = true)
20    private PerfilCelula perfil;
21    @ManyToOne
22    @JoinColumn(name = "setor_id", nullable = true)
23    private Setor setor;
24    @OneToMany(mappedBy = "celula", fetch = FetchType.LAZY)
25    private List<Membro> membro = new ArrayList();
26    @Column(nullable = false)
27    private boolean ativo = true;
28 }

```

6.1.2 View

O *View* está representado pelas páginas XHTML do JSF, que renderizam o conteúdo da aplicação no navegador do usuário.



Figura 7 – Exemplo de View no sistema - Dashboard

A *View* é a camada que interage com o usuário e envia suas requisições para o *Controller* trabalhar as regras de negócio e atualizar novamente a *View*. Adiante um exemplo de código de uma *View*: relatorioCelula.xhtml

```

1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
   http://www.w3.org/TR/html4/loose.dtd">
3 <ui:composition template="/WEB-INF/templates/atlant/main.xhtml"
4   xmlns="http://www.w3.org/1999/xhtml"
5   xmlns:f="http://java.sun.com/jsf/core"
6   xmlns:ui="http://java.sun.com/jsf/facelets"
7   xmlns:p="http://primefaces.org/ui"
8   xmlns:o="http://omnifaces.org/ui"
9   xmlns:c="http://java.sun.com/jsp/jstl/core"
10  xmlns:h="http://xmlns.jcp.org/jsf/html">
11
12
13 <ui:define name="conteudo">
14 <f:metadata>
15 <o:viewParam name="celula" value="#{relatorioCelulaBean.celula}"
   converter="celulaConverter"/>
16 <f:viewParam name="" />

```

```

17 </f:metadata>
18
19 <!-- START CONTENT FRAME -->
20
21 <div class="content-frame">
22 <!-- START CONTENT FRAME TOP -->
23 <div class="content-frame-top">
24 <div class="page-title">
25 <div class="pull-left">
26 <h2><span class="fa fa-users"/>
27 <c:if test="#{relatorioCelulaBean.celula.id != null}">
28     <h:outputText value="#{relatorioCelulaBean.celula.nome}"
29         styleClass="margin-left-5"/>
30 </c:if>
31 <c:if test="#{relatorioCelulaBean.celula.id eq null}">
32     <h:outputText value="Nenhuma celula selecionada" styleClass="
33         margin-left-5"/>
34 </c:if>
35 </h2>
36 </div>
37 </div>
38 <!-- START CONTENT FRAME BODY -->
39 <div class="content-frame-body content-frame-body-left">
40 <h:panelGroup id="semanass">
41 <ui:repeat id="listaSemanas" value="#{relatorioCelulaBean.semanas
42     }" var="semana" varStatus="passo">
43 <fieldset class="caixa">
44 <legend class="legendSemana">Semana #{passo.index+1}</legend>
45
46 <div class="reuniao">
47     <fieldset>
48         <legend>Culto
49         <h:outputText value="#{semana.culto.dataCulto}" style
50             ="color: #95b75d;">
51         <f:convertDateTime timeZone="America/Sao_Paulo"
52             pattern="dd/MM/yyyy"/>
53         </h:outputText>
54     </legend>
55     <div class="reuniao-in">
56         <div class="culto">
57             <label class="descriptor">Membros Presentes:</

```

```

        label>
54         <span><h:outputText value="#{semana.presencaCulto
            }"/></span>
55     </div>
56 </div>
57 </fieldset>
58 </div>
59 </fieldset>
60 </ui:repeat>
61 </h:panelGroup>
62 </div>
63 </div>
64 </ui:define>
65 <!-- THIS PAGE PLUGINS -->
66 <ui:define name="scriptPage">
67 <h:outputScript library="atlant" name="/js/plugins/ichack/ichack.
    min.js"/>
68 <h:outputScript library="atlant" name="/js/plugins/knob/jquery.
    knob.min.js"/>
69 <h:outputScript library="atlant" name="/js/plugins/owl/owl.
    carousel.min.js"/>
70 </ui:define>
71 </ui:composition>

```

6.1.3 Controller

Quando se trabalha com o padrão MVC em um projeto JSF a *View* é composta pelos arquivos XHTML e os *Managed Beans*, mas por questão de tempo, este projeto está usando os *Managed Beans* como *Controller*, mas sabendo que para o modelo MVC completo deve-se usar outro pacote de classes que trabalhe com as regras de negócio. A camada *Controller* está representado pelos *Managed Beans* que mantêm as regras de negócio e controlam as *Views* e requisitam o *Model*. Um exemplo de Controller é o Managed Bean "LoginBean":

```

1 @SessionScoped
2 @ManagedBean(name = "loginBean")
3 public class LoginBean implements Serializable {
4     private static final long serialVersionUID = 1L;
5     private FacesContext facesContext;
6     private HttpServletRequest request;
7     private HttpServletResponse response;
8     private String usuario;

```

```
9
10     public final void atualizaContexto() throws IOException{
11         facesContext = FacesContext.getCurrentInstance();
12         request = FacesUtil.getContextRequest();
13         response = FacesUtil.getContextResponse();
14     }
15
16     public void preRender() throws IOException {
17         atualizaContexto();
18         if ("true".equals(request.getParameter("invalid"))) {
19             FacesContext.getCurrentInstance().addMessage("
20                 msgLogin", new FacesMessage(FacesMessage.
21                     SEVERITY_ERROR, "Usuario ou senha invalido!", "
22                     Usuario ou senha invalido!"));
23         }
24     }
25
26     public void login() throws ServletException, IOException {
27         atualizaContexto();
28         RequestDispatcher dispatcher = request.
29             getRequestDispatcher("/j_spring_security_check");
30         dispatcher.forward(request, response);
31         facesContext.responseComplete();
32     }
33     ... getters e setters...
34 }
```

Este *controller* usa o método *atualizaContexto()* para sempre capturar o *request* atualizado do contexto do JSF. Se houver mensagem de erro, é exibida pelo método *preRender()*. O método *login()* despacha o *request* capturado pelo método *atualizaContexto()* para o filtro de acesso do *Spring Security* disponível no contexto da aplicação no link *j_spring_security_check*.

6.2 Organização do Código Fonte

O projeto foi desenvolvido na IDE *Netbeans* e está organizado da seguinte forma:

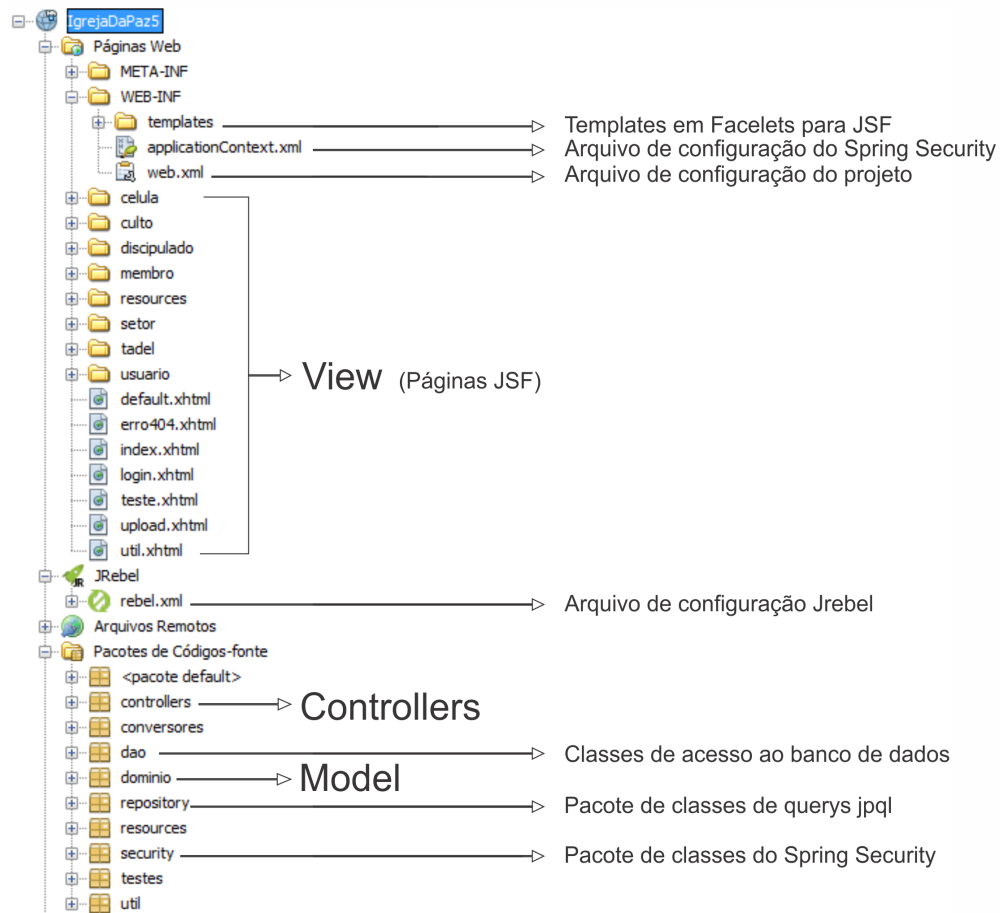


Figura 8 – Organização do Código

A figura 8 mostra a forma como foi organizado o código fonte dentro do ambiente de desenvolvimento *Netbeans*. Os diretórios do projeto contemplam as páginas JSF, também encontra-se o pacote de *controllers* que excepcionalmente são os *Managed Beans* que intermediam a *Viwe* e o *Model* que está representado pelos pacotes de domínio e de DAO (*Data Access Object*) que são as classes que lidam diretamente com a persistência dos dados da aplicação.

6.3 Autenticação e Autorização com Spring Security

Para utilizar o *Spring Security* no projeto, foi preciso lidar com três conceitos de seu funcionamento, ilustrados na figura 9:



Figura 9 – Processo de Autenticação - *Spring Security* (WEISSMANN, 2012, p.235)

6.3.1 Filtro de Acesso

Responsável por lidar com todas as chamadas externas que nossa aplicação receberá. Na aplicação o filtro de acesso foi configurado da seguinte forma: web.xml

```
1 <filter>
2   <filter-name>springSecurityFilterChain</filter-name>
3   <filter-class>
4     org.springframework.web.filter.DelegatingFilterProxy
5   </filter-class>
6 </filter>
7 <filter-mapping>
8   <filter-name>springSecurityFilterChain</filter-name>
9   <url-pattern>/*</url-pattern>
10 </filter-mapping>
```

Este filtro é o `DelegatingFilterProxy` e seu funcionamento consiste em encaminhar todas as requisições para um componente fundamental do Spring Security que é o Filter Chain (filtro encadeador), o real responsável pela configuração de acesso às URLs do sistema, e é executado como um bean gerenciado pelo contexto de aplicação do Spring. (WEISSMANN, 2012)

6.3.2 Gerenciador de Autenticação

Responsável por receber as credenciais de um usuário e verificar sua autenticidade, o que é feito testando-as contra um ou mais provedores de autenticação a que tem acesso. O gerenciador de autenticação foi configurado no arquivo de configuração padrão do *Spring Security applicationContext.xml* acrescentado o seguinte código:

```
1 <beans:bean id="appUserDetailsService" class="security.  
    AppUserDetailsService" />  
2  
3 <authentication-manager>  
4     <authentication-provider user-service-ref="  
        appUserDetailsService"/>  
5 </authentication-manager>
```

A tag *authentication-manager* está chamando a referencia para um serviço de detalhamento de usuários que é o serviço responsável por trazer do banco de dados o usuário requerido pelo filtro de acesso, então o *appUserDetailsService* por sua vez, informa que esse serviço está disponível em uma classe chamada *security.AppUserDetailsService* que foi implementada aparte para obter recursos que o *Spring Security* não disponibiliza por padrão. essa classe é denominada como *Proveror de Autenticação*.

6.3.3 Provedor de Autenticação

O Provedor de Autenticação é uma fonte de dados feita especificamente para armazenar informações de usuários capaz de executar uma única consulta que recebe como parâmetro as credenciais retornando informações do usuário caso existam (incluindo suas permissões de acesso). (WEISSMANN, 2012) No sistema o provedor de autenticação foi implementado da seguinte forma:

```
1 public class AppUserDetailsService implements UserDetailsService
2 {
3     @Override
4     public UserDetails loadUserByUsername(String usuario) throws
5         UsernameNotFoundException {
6         EntityManager em = JpaUtil.getEntityManager();
7         DaoFactory dao = new DaoFactory(em);
8         LoginRepository loginRep = new LoginRepository(em, dao);
9         Pessoa pessoa = loginRep.getPessoaByUsuario(usuario);
10        UsuarioSistema user = null;
11        if (pessoa != null) {
12            user = new UsuarioSistema(pessoa, getPermissoes(
13                pessoa));
14        }
15        em.close();
16        return user;
17    }
18    private Collection<? extends GrantedAuthority> getPermissoes(
19        Pessoa pessoa) {
20        List<SimpleGrantedAuthority> authorities = new ArrayList
21            <>();
22        pessoa.getLogin().getPermissoes().stream().forEach((permissao
23            ) -> {
24            String PREFIXO = "ROLE_";
25            authorities.add(new SimpleGrantedAuthority(PREFIXO +
26                permissao.getDescricao().toUpperCase()));
27        });
28        return authorities;
29    }
30 }
```

O método *loadUserByUsername* recebe o nome do usuário informado no formulário de login e retorna um *UserDetails* que é uma interface de usuário específica do *Spring Security*, então ela instancia um *UsuarioSistema* que é uma classe que estende da classe *User* do *Spring Security* que por sua vez implementa *UserDetails* e também armazena um objeto do tipo *Pessoa* do domínio da aplicação, dessa forma é possível ter um objeto no contexto do JSF que contenha as informações de usuário, permissões e da pessoa a quem se referem essas informações. A seguir o código da classe *UsuarioSistema*:

```
1 public class UsuarioSistema extends User {
2     private static final long serialVersionUID = 1L;
3     private final Pessoa pessoa;
4     public UsuarioSistema(Pessoa pessoa, Collection<? extends
5         GrantedAuthority> authorities) {
6         super(pessoa.getLogin().getUsuario(), pessoa.getLogin().
7             getSenha(), authorities);
8         this.pessoa = pessoa;
9     }
10    public Pessoa getPessoa() {
11        return pessoa;
12    }
13 }
```

A classe *UsuarioSistema* armazena um objeto do tipo *Pessoa* e as lista de permissões que o usuário possui. A partir dessas informações é possível gerenciar autorização de páginas e até mesmo componentes dentro do contexto do JSF.

No contexto do controle de acesso a URLs o Chain Filter acessa apenas o gerenciador de autenticação, não precisando portanto se preocupar com a origem dos dados de autenticação do usuário tal como pode ser visto no esquema a seguir:(WEISSMANN, 2012)

6.4 Relatório de Célula

O Relatório de Célula é um item importante do sistema, pois é através dele que os líderes e supervisores da igreja obtêm informações rápidas a respeito das reuniões e terem conhecimento do estado de cada célula analisando a frequência dos membros.

Para este projeto, o relatório de célula foi desenvolvido com o auxílio de uma biblioteca java chamada *Itext*. O relatório pode ser visualizado em uma página comum do sistema, bastando que o usuário selecione o mês e o ano que deseja e clicar em um botão para emití-lo e visualizar na própria página. Após sua visualização, o usuário tem a opção de salvar uma versão do relatório em PDF, nesse momento entra o trabalho com a biblioteca *Itext*, que foi utilizada em um ManageBean para gerar um relatório em PDF que mantenha as mesmas características definidas pelos requisitos do cliente.

Relatório de Célula					
Célula: Juliana					
Lider: Juliana S. de Castro					
Supervisor de Setor: Rosivaldo F. de Souza					
Mês: Junho					
Semana 01					
Célula 02/06/2016		Tadel 01/06/2016		Culto	
Presentes	11	Presentes	5	Presentes	0
Convidados	0				
Discípulos	0				
Oferta	5.5				
Semana 02					
Célula 09/06/2016		Tadel 08/06/2016		Culto 05/06/2016	
Presentes	10	Presentes	3	Presentes	7
Convidados	0				
Discípulos	0				
Oferta	7.25				
Semana 03					
Célula 16/06/2016		Tadel 15/06/2016		Culto 12/06/2016	
Presentes	9	Presentes	0	Presentes	6
Convidados	0				
Discípulos	0				
Oferta	2.65				
Semana 04					
Célula 23/06/2016		Tadel 22/06/2016		Culto 19/06/2016	
Presentes	9	Presentes	4	Presentes	6
Convidados	0				
Discípulos	0				
Oferta	6.0				
Semana 05					
Célula 30/06/2016		Tadel 29/06/2016		Culto 26/06/2016	
Presentes	9	Presentes	5	Presentes	8
Convidados	2				
Discípulos	0				
Oferta	5.75				

Figura 10 – Relatório de Célula em PDF

A seguir temos uma imagem de um relatório de célula emitido no sistema, nele estão descritas todas as reuniões de célula, reuniões de Tadel e cultos que a referida célula participou durante o mês selecionado:

IGREJA DA PAZ		Rosivaldo Freitas de Souza Filho	
Juliana		06/2016	
Semana 1		Frequências:	
Célula 02/06/2016		80,0%	
Presentes: 11 de 12 membros		Presença nas Reuniões de Célula	
Convidados: 0		Ações:	
Discípulos: 0		Salvar PDF	
Oferta: 5,50			
Semana 2			
Célula 09/06/2016			
Presentes: 10 de 12 membros			
Convidados: 0			
Discípulos: 0			
Oferta: 7,25			
Semana 3			
Célula 16/06/2016			
Presentes: 9 de 12 membros			
Convidados: 0			
Discípulos: 0			
Oferta: 2,65			
Semana 4			

Figura 11 – Relatório de Célula

O relatório é organizado em sessões de semanas, mantendo informações das reuniões de célula, Tadel e Culto de Celebração, cada uma no seu dia. Para esse sistema, foi definido que a semana comece no domingo e termine no sábado, pois é desta forma que estão organizadas as atividades da igreja, inclusive a ficha de relatório atualmente usada está definida da mesma forma.

Com a forma que o sistema está organizado, não temos mais uma ficha ou tabela onde se encontram todas as informações juntas, mas o que se têm são seções do sistema especializadas em cada área da gestão organizacional da igreja, como módulos, onde se pode encontrar sessões apenas para manter informações de membros, outras apenas para células, outras para discipulado, de forma que se o usuário quiser encontrar alguma informação terá de percorrer um caminho lógico e intuitivo para tal.

6.5 Interface Web

6.5.1 Template com Facelets

Para este projeto foi utilizado Facelets que é uma linguagem de descrição de páginas emph(PDL – Page Description Language) criada especificamente para JSF, permitindo criar emphtemplates possibilitando o reuso de padrões de telas em todo o sistema, assim é possível dar manutenção em sistemas complexos e evitar ao máximo a repetição de código para a criação de telas. O trecho de código a-segui representa o arquivo principal emphtemplate utilizado no projeto, sendo que existem arquivos secundários que são incorporados a este através de emphtags especiais do emphFacelets:

<Main.xhtml>

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
   http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:h="http://java.sun.com/jsf/html"
5     xmlns:f="http://java.sun.com/jsf/core"
6     xmlns:ui="http://java.sun.com/jsf/facelets">
7
8     <h:head>
9
10         <f:facet name="first">
11             <meta http-equiv="Content-Type" content="text/html;
12                 charset=utf-8" />
13             <meta http-equiv="X-UA-Compatible" content="IE=edge"
14                 />
15             <meta name="viewport" content="width=device-width,
16                 initial-scale=1" />
17             <title>Igreja da Paz</title>
18         </f:facet>
19
20         <f:facet name="middle">
21             <h:outputStylesheet id="theme" library="atlant" name=
22                 "/css/theme-default.css" />
23             <h:outputStylesheet name="/css/meu_estilo.css" />
24         </f:facet>
25
26     </h:head>
```

```
24     <h:body>
25         <!-- START PAGE CONTAINER -->
26         <div class="page-container">
27
28             <ui:insert name="sidebar" >
29                 <ui:include src="esquerda.xhtml" />
30             </ui:insert>
31
32             <ui:insert name="corpo" >
33                 <!-- PAGE CONTENT -->
34                 <div class="page-content">
35                     <ui:insert name="topo" >
36                         <ui:include src="topo.xhtml" />
37                     </ui:insert>
38                     <ui:insert name="conteudo" >
39
40                         </ui:insert>
41                     </div>
42                     <!-- END PAGE CONTENT -->
43                 </ui:insert>
44             </div>
45             <!-- END PAGE CONTAINER -->
46             <ui:insert name="base" >
47                 <ui:include src="footer.xhtml" />
48             </ui:insert>
49
50             <ui:insert name="scriptPage" ></ui:insert>
51     </h:body>
52 </html>
```

Os elementos principais desse emphtemplate estão definidos nas emphtags <ui:insert> que definem onde cada trecho de página será inserido no emphtemplate. Os arquivos secundários são inseridos no emphtemplate principal através da emphtag <ui:include> que recebe no parâmetro "src" o nome do arquivo externo a ser incluído. Os arquivos de emphtemplate secundários são constituídos basicamente de emphtags HTML com classes CSS presentes no "emphtemplate Bootstrap" que será apresentado na próxima sessão desse capítulo.

Uma grande vantagem de utilizar emphtemplates com emphFacelets é fazer com que cada página do sistema contenha basicamente apenas o conteúdo que interesse a ela, ou seja, quando o navegador acessa uma determinada página, o emphtemplate é renderizado trazendo para dentro de si o conteúdo dela, isso significa que não é necessário repetir em

cada página do sistema o código, por exemplo, do menu principal, ele é chamado toda vez que o navegador acessar uma página que utilize o emphtemplate. a seguir o código da página de listagens de Cultos:

<listarCultos.xhtml>

```

1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
   http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <ui:composition template="/WEB-INF/templates/atlant/main.xhtml"
4     xmlns:ui="http://java.sun.com/jsf/facelets"
5     ...outras taglibs...
6     xmlns:h="http://xmlns.jcp.org/jsf/html">
7
8     <ui:define name="conteudo">
9         <f:metadata>
10             <o:viewParam name="celula" value="#{cultoBean.celula}"
11                 " converter="celulaConverter"/>
12             <f:viewParam name="" />
13             <f:event listener="#{cultoBean.inicializarCultos}"
14                 type="preRenderView"/>
15         </f:metadata>
16         <div class="col-md-12">
17
18             <div class="panel panel-default" >
19                 <div class="panel-heading">
20                     <h3 class="panel-title"><span class="fa fa-
21                         plus-square"></span> Cultos</h3>
22                 </div>
23                 <div class="panel-body">
24                     <p:dataTable id="tab-reunioes" value="#{
25                         cultoBean.cultos}" var="culto" >
26                         <p:column headerText="Data da Reuniao"
27                             style="min-width: 10em !important;">
28                             <h:outputText value="#{culto.
29                                 dataCulto}" >
30                                 <f:convertDateTime timeZone="
31                                     America/Sao_Paulo" pattern="dd/
32                                     MM/yyyy"/>
33                             </h:outputText>
34                         </p:column>

```

```

28         <p:column headerText="Observacao">
29             <h:outputText value="#{culto.
30                 observacao}"/>
31         </p:column>
32         <p:column headerText="Acoes" style="width
33             : 12.8em;">
34             <h:form id="form">
35                 <div class="form-horizontal">
36                     <p:button value="Marcar
37                         Presen\c{c}a" outcome="/
38                         celula/presencaCulto">
39                         <f:param name="celula"
40                             value="#{cultoBean.
41                                 celula.id}"/>
42                         <f:param name="culto"
43                             value="#{culto.id}"/>
44                     </p:button>
45                 </div>
46             </h:form>
47         </p:column>
48     </p:dataTable>
49 </div>
50 </div>
51 </ui:define>
52 </ui:composition>

```

Para definir uma página de conteúdo é preciso apenas da tag `<ui:composition>` com o atributo "template" apontando para o arquivo do template principal `"/WEB-INF/template/atlant/main.xhtml"` e também uma tag `<ui:define name="conteudo">` onde o atributo "name" indica em qual dos "inserts" do template principal o conteúdo dessa página será inserido.

6.5.2 Layout

O layout utilizado neste projeto foi baseado em um *template Bootstrap* pago chamado *Atlant* que possui componentes de interface web que estão prontos para serem usados pelo desenvolvedor na construção das páginas, diminuindo assim o tempo de construção de um sistema.

Por se tratar de um projeto construído sobre o *framework JSF*, houve a necessidade

de utilizar bibliotecas de componentes adicionais como o *Primefaces* para agregarem funcionalidades mais completas ao JSF. A utilização de duas tecnologias de construção de interfaces web gerou uma demanda de integração entre as plataformas JSF /*Primefaces* e o *Bootstrap*. A solução encontrada para tal integração se deu da seguinte forma:

6.5.3 Integração JSF/Bootstrap

Ao adicionarmos a biblioteca de componentes *Primefaces* ao JSF temos a possibilidade de utilizarmos temas que podem ser comprados no próprio site do *Primefaces* em versões mais completas, baixadas gratuitamente em versões simplificadas ou até mesmo customizadas pelo desenvolvedor que os utiliza. No caso desse Projeto, foi utilizado o tema "Bootstrap" que encontra-se disponível gratuitamente no site do *Primefaces*.

o tema do *bootstrap* para *Primefaces* torna a aparência dos componentes bem mais próxima do *Bootstrap* e ele utiliza a biblioteca *Javascript JQuery* que também será necessária para o funcionamento do *template bootstrap*. A aplicação do tema do *bootstrap* no projeto foi feita com o acréscimo do seguinte trecho de código no arquivo `web.xml` encontrado no diretório `/WEB-INF` do projeto Java:

```
1 <context-param>
2     <param-name>primefaces.THEME</param-name>
3     <param-value>bootstrap</param-value>
4 </context-param>
```

Após atualizado o arquivo `web.xml` foi necessário mudar a forma de importar os arquivos *CSS* e *Javascript* no arquivo principal do *template* da seguinte forma:

Código original:

```
1 <!-- CSS INCLUDE -->
2 <link rel="stylesheet" type="text/css" id="theme" href="css/theme
   -default.css"/>
3 <!-- EOF CSS INCLUDE -->
```

Novo código:

```
1 <f:facet name="middle">
2     <h:outputStylesheet id="theme" library="atlant" name="/css/
   theme-default.css" />
3 </f:facet>
```

6.5.4 Interface Web-responsiva

Como foi exposto anteriormente, este projeto foi construído com base em um *template Bootstrap* que tem como uma das principais características a capacidade de exibir seus componentes de forma ajustada em diversos tamanhos de *Displays*. Aliando essa característica do *bootstrap* aos componentes do *Primefaces* foi possível desenvolver um sistema web Com *Java EE* e uma interface responsiva, intuitiva e agradável aos usuários.

O *Twitter Bootstrap* utiliza *Brakpoints* em suas folhas de estilo *CSS* para reorganizar a disposição dos elementos visuais de acordo com as dimensões do navegador que esteja acessando suas páginas. Os principais *Brakpoints* utilizados pelo *Twitter Bootstrap* em seu *CSS* são:

```
1 /* Telefones em landscape e abaixo */
2 @media (max-width: 480px) { ...}
3 /* Telefone em landscape a tablet em portrait */
4 @media (max-width: 767px) { ...}
5 /* tablet em portrait a landscape e desktop */
6 @media (min-width: 768px) and (max-width: 979px) { ...}
7 /* Desktop grande */
8 @media (min-width: 1200px) { ...}
```

Código extraído de (ZEMEL, 2012)



Figura 12 – Dashboard Responsivo

6.6 Testes

6.6.1 Testes de Rotinas

”Testes isolados de rotinas ou módulos ou ”funções” de programas. Obedecem aos padrões de qualidade estabelecidos pela *Engenharia de Software* e pela organização” (REZENDE, 2005, p. 272)

Um exemplo desse tipo de teste executado durante a fase de desenvolvimento do sistema foi o teste de *inserção de membro* para saber se o mapeamento da classe de domínio do membro estava corretamente configurado nos padrões JPA e de acordo com as regras estabelecidas nos requisitos do sistema. Este teste foi realizado em uma classe java comum dentro de um método *main*. A seguir está o código da classe *InserMembro.java*:

```
1 public class InserirMembro {
2     public static void main(String[] args) {
3         EntityManager em = JpaUtil.getEntityManager();
4         DaoFactory dao = new DaoFactory(em);
5         Membro membro = new Membro();
6         membro.setAtivo(true);
7         membro.setCelula(null);
8         membro.setCelular("(23) 0987-6758");
9         membro.setDataNascimento(new Date());
10        membro.setEmail("ana@teste.com.br");
11        membro.setFoto(null);
12        membro.setNome("Ana Julia");
13        membro.setSexo(Sexo.FEMININO);
14        membro.setEstadoCivil(EstadoCivil.SOUTEIRO);
15        membro.setTelefone("(23) 5432-9867");
16        dao.getMembroDao().addOrUpd(membro);
17        em.close();
18        System.exit(0);
19    }
20 }
```

Dependendo das respostas obtidas do compilador, correções eram feitas nas anotações JPA da classe Membro ou das classes que possuíam algum relacionamento com ela. Em alguns casos, as alterações feitas nas anotações JPA geravam a necessidade de alterar as tabelas do banco de dados, que eram feitas manualmente, pois o *Hibernate* não prevê alterações estruturais do banco após a criação das tabelas.

6.6.2 Testes de Verificação e Validação

A verificação questiona se a maneira que se desenvolve o software está correta. A validação questiona se o *software* que estamos construindo está correto, ou seja, se o software funciona da maneira esperada pelo usuário o cliente. A verificação geralmente está relacionada com o projeto de software. E a validação com os produtos do mesmo. (REZENDE, 2005)

Nesse teste foram executadas, juntamente com o cliente, rotinas de casos de uso específicas para avaliar se o comportamento do sistema estava de acordo com as especificações do projeto. Foram executadas rotinas dos casos de uso UC01, UC02, UC03, UC05, UC06, UC07 e UC08, necessários para se obter um relatório de célula completo no UC13(Emitir Relatório).

As informações obtidas da execução do caso de uso UC13 foram comparadas as informações contidas nos formulários utilizados atualmente como forma de relatório. A partir dessa comparação foi possível validar as funcionalidades do relatório emitido pelo sistema. Este processo foi repetido em dias diferentes para confirmar que a execução de tais tarefas podem ser repetidas em momentos diferentes, ou seja, em estados diferentes do sistema.

Ficha de relatório atual

IGREJA DA PAZ				Líder: Juliana				Mês: jun/16												
	Nome	Telefone	Aniversário	MDA		Semana 1			Semana 2			Semana 3			Semana 4			Semana 5		
	(completo)	(celular)	(data)	(discipulador)	Fez?	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC
1	Ana Luisa Castro	(65)8456-4962	27-jun	Juliana	SIM	F	P	P	F	P	P	J	P	P	F	P	P	F	P	P
2	Eduardo	(65)8423-4406	4-mai	Luís Henrique	SIM	P	P	P	F	P	P	J	P	P	P	P	P	P	P	P
3	Gabriel Castro	(65)8467-1482	7-mai	Carlos Coelho	SIM	P	P	P	P	P	P	J	P	F	F	P	P	P	P	P
4	Guilherme		4-mai	Gabriel	SIM	P	P	P	F	P	P	J	P	P	P	P	P	P	P	P
5	Julia Cardeira	(65)8420-0406			NÃO	F	P	F	F	J	F	F	F	F	F	P	F	F	F	F
6	Julio			Lucas	SIM	P	P	P	P	P	P	J	J	P	P	P	P	P	P	P
7	Lucas		20-mar	Guilherme	SIM	P	P	P	P	P	J	J	J	P	P	P	P	P	P	P
8	Luís Henrique Magalhães	(65)8424-3729	9-jun	Rodrigo	SIM	F	P	P	F	J	P	F	J	P	F	F	P	F	P	P
9	Victor Hugo Caldeira		30-out		NÃO	F	P	F	F	P	F	F	F	F	F	F	F	F	F	F
10	Salla				NÃO	F	F	F	F	P	F	F	F	F	F	F	P	F	F	F
11																				
12																				
13																				
14																				
15																				
NOMES (APLICANDO FÉ)																				
	Arthur	(65)8435-2431	26-dez																	
	Claudio																			
	Danilo																			
	Marco Antonio																			
Total (numérico):					7	5	9	7	3	8	6	0	4	6	4	7	8	5	7	

Relatório do sistema

Semana 2

Célula 09/06/2016

Tadel 08/06/2016

Culto 05/06/2016

Presentes: 8 de 10 membros

Membros Presentes: 3

Membros Presentes: 7

Convidados: 0

Discipulados: 0

Oferta: 4,20

Figura 13 – Teste de Validação

Apartir deste teste foi possível comprovar a existência de incoerências nos dados encontrados na ficha de relatório utilizada atualmente pela igreja. Um exemplo é a ficha apresentada no teste de validação:

Ficha de relatório atual

KIRIA DA PAZ				Líder: Juliana		Mês: jun/16															
	Nome	Telefone	Aniversário	MDA		Semana 1			Semana 2			Semana 3			Semana 4			Semana 5			
	(completo)	(celular)	(data)	(discipulador)	Fez?	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	
1	Ana Luisa	(65)8456-8888	27-jun	Juliana	SIM	F	P	P	S	F	P	P									
2	Eduardo	(65)8423-8888	4-mai	Luís Henrique	SIM	P	P	P	F	P	P										
3	Gabriel Castro	(65)8467-8888	7-mai	Carlos Coelho	SIM	P	P	P	P	P	P										
4	Guilherme		4-mai	Gabriel	SIM	P	P	P	F	P	P										
5	Julia Cardeira	(65)8420-8888			NÃO	F	P	F	F	J	F										
6	Julio			Lucas	SIM	P	P	P	P	P	P										
7	Lucas		20-mar	Guilherme	SIM	P	P	P	P	P	J										
8	Luís Henrique	(65)8424-8888	9-jun	Rodrigo	SIM	F	P	P	F	J	P										
9	Victor Hugo		30-out		NÃO	F	P	F	F	P	F										
10	Saila				NÃO	F	F	F	F	P	F										
11																					
12																					
13																					
14																					
15																					
NOMES (APLICANDO FÉ)																					
	Arthur	(65)8435-8888	26-dez																		
	Claudio																				
	Danilo																				
	Marco Antonio																				
Total (numérico):					7	5	9	7	3	8	6	0	4	6	4	7	8	5	7		

Relatório do sistema

Semana 1

Célula 02/06/2016

Tadel 01/06/2016

Culto

Presentes: 9 de 10 membros

Membros Presentes: 5

Membros Presentes: 0

Convidados: 0

Discipulados: 0

Oferta: 5,50

Figura 14 – Teste de Validação

Legenda: CC = Culto de Celebração; CE = Célula; TD = Tadel

Nota-se que na ficha de relatório em *excel* foram informadas as presenças dos membros no dia 5 da primeira semana, sendo que na primeira semana do mês de Junho de 2016 não houve domingo, pois a semana começou na quarta feira dia primeiro, consequentemente não haveria culto nesta semana. Nesse caso, as semanas posteriores do mês de Junho foram preenchidas incorretamente. A inconsistência de dados se encontra no momento em que outros líderes de célula preenchem relatórios do mesmo mês, uns da forma correta, outros da forma como apresentada acima, gerando confusão na hora de analisar e processar esses dados, dificultando o entendimento de tais fichas.

O relatório emitido pelo sistema já organiza as semanas em blocos para que sejam facilmente visualizadas e apresenta apenas as reuniões que realmente houveram e foram preenchidas com as presenças dos membros das células.

Os resultados deste teste comprova que o sistema realiza uma das funções que o seu projeto propôs que fizesse, que é a apresentação de relatórios dinâmicos, de fácil manuseio e compreensão.

7 Conclusão e Trabalhos Futuros

7.1 Conclusão

Este trabalho teve como objetivo principal a construção de uma aplicação web para o gerenciamento organizacional da *Igreja da Paz de Miracema TO* que resolvesse os problemas de inconsistência de dados e aumentasse a agilidade na coleta e processamento de dados de suas reuniões.

Ao longo do desenvolvimento deste trabalho foi possível obter os requisitos necessários para construir um modelo que orientou o desenvolvimento de uma aplicação com uma interface responsiva, que coleta e processa dados, gera relatórios dinâmicos e organiza as informações cadastrais de membros e células.

Com base nos testes e avaliações executados juntamente com os *stakeholders* sobre as funcionalidades apresentadas pelo sistema, conclui-se que o objetivo deste trabalho foi alcançado, trazendo como benefícios aos usuários o acesso rápido e conciso à informações pertinentes as células através do relatório de célula, a mobilidade de poderem acessar o sistema em seus dispositivos móveis e a organização das informações que os auxiliam em tomada de decisões.

7.2 Trabalhos Futuros

Novas funcionalidades foram sugeridas para as próximas versões deste sistema:

- Aplicativo Mobile: Desenvolvimento de um aplicativo para dispositivos móveis integrado a aplicação web.
- Trilho de liderança: Esta funcionalidade já aparece em partes no presente projeto, como uma lista de requisitos para um membro se tornar líder de célula. Futuramente esta funcionalidade estará implementada em sua totalidade.
- Cadastro de Áreas, Distritos, e Regiões: Essas são estruturas organizacionais da igreja de níveis superiores aos de Célula e Setor.
- Cadastro de Grupos de Evangelismo: São grupos externos, diretamente vinculados às células.
- Tesouraria: Registros financeiros da igreja.

Referências

- ALEIXO, V. C. *“DEUS FAZ, O TEMPLO DOS ANJOS MOSTRA”*: perfil eclesial e adesão religiosa na Igreja do Evangelho Quadrangular em Belo Horizonte. Tese (Doutorado) — UNIVERSIDADE FEDERAL DE MINAS GERAIS, Brasil, Belo Horizonte, 2014. Citado 2 vezes nas páginas 14 e 16.
- APACHE. *Apache Tomcat*. 2016. Disponível em: <<http://tomcat.apache.org/index.html>>. Acesso em: 13/09/2016. Citado na página 31.
- ASTAH. *Comunidade Astah*. 2016. Disponível em: <<http://astah.net/editions/community>>. Acesso em: 13/09/2016. Citado na página 32.
- BAUTAZAR, E. L. F. F. M. R. G. M. M. V. de Paula;Glauber da R. *Uma Abordagem Prática sobre a Aplicação do Padrão MVC com o Framework Struts*. Tese (Doutorado) — Faculdade Metodista Grambery (FMG), Brasil, Juiz de Fora - MG, 2006. Citado 2 vezes nas páginas 7 e 22.
- BEZERRA, E. *Princípios de Análise e Projeto de Sistemas*. [S.l.]: Rio de Janeiro: Elsevier, 2002. Citado na página 22.
- BOOCH GRADY, R. J. e. J. I. *UML, Guia do Usuário*. [S.l.]: Rio de Janeiro: Elsevier, 2012. Citado na página 22.
- CAELUM. *APOSTILA JAVA PARA DESENVOLVIMENTO WEB*. 2016. Disponível em: <<https://www.caelum.com.br/apostila-java-web/o-que-e-java-ee/>>. Acesso em: 25/08/2016. Citado na página 26.
- CORDEIRO, G. *Aplicações Java para web com JSF e JPA*. [S.l.]: Casa do Código, 2012. Citado na página 21.
- FARIA, N. J. T. *JPA e Hibernate*. [S.l.]: AlgaWorks, 2015. Citado 2 vezes nas páginas 29 e 30.
- FARIA, T. *Java EE 7 com JSF, PrimeFaces e CDI*. [S.l.]: AlgaWorks Softwares, 2013. Citado 2 vezes nas páginas 26 e 27.
- GETBOOTSTRAP. *Edição de Códigos Rápida e Inteligente*. 2016. Disponível em: <<http://getbootstrap.com/2.3.2/components.html>>. Acesso em: 13/09/2016. Citado na página 31.
- GUEDES, G. T. *UML 2, Guia de Consulta Rápida*. [S.l.]: Novatec, 2005. Citado 2 vezes nas páginas 34 e 40.
- HEUSER, C. A. *Projeto de Banco de Dados 4 - Série Livros Didáticos*. [S.l.]: SAGRA LUZZATTO, 1998. (4º ed.). Citado 3 vezes nas páginas 22, 23 e 24.
- HUBER, P. A. *A Visão do MDA*. 1999. Disponível em: <<http://www.visaomda.com/a-visao-mda/>>. Acesso em: 30/07/2015. Citado 2 vezes nas páginas 14 e 17.

- ITEXT. *Núcleo iText 7*. 2016. Disponível em: <<http://itextpdf.com/itext7suite>>. Acesso em: 14/09/2016. Citado na página 33.
- JREBEL. *O que é myJRebel?* 2016. Disponível em: <<https://my.jrebel.com/e>>. Acesso em: 14/09/2016. Citado na página 33.
- K19. *Modelo Relacional e SQL*. 2015. Disponível em: <www.k19.com.br>. Acesso em: 14/09/2016. Citado na página 23.
- K19. *Persistência com JPA2 e Hibernate*. 2015. Disponível em: <www.k19.com.br>. Acesso em: 13/09/2016. Citado 2 vezes nas páginas 30 e 31.
- MENDES, D. R. *Programação Java com Ênfase em Orientação a Objetos*. [S.l.]: Novatec, 2009. Citado 2 vezes nas páginas 25 e 26.
- NETBEANS-EDIT. *Edição de Códigos Rápida e Inteligente*. 2016. Disponível em: <https://netbeans.org/features/index_pt_BR.html>. Acesso em: 13/09/2016. Citado na página 32.
- NETBEANS-IDE. *O que é o NetBeans?* 2016. Disponível em: <https://netbeans.org/index_pt_PT.html>. Acesso em: 13/09/2016. Citado na página 32.
- OLIVEIRA, D. de Pinho Rebouças de. *Estrutura organizacional: uma abordagem para resultados e competitividade*. [S.l.]: São Paulo: Atlas, 2006. Citado na página 16.
- OLIVEIRA, H. C. de. *JPA Eficaz*. [S.l.]: Casa do Código, 2013. Citado na página 29.
- POSTGRES. *About PostgreSQL*. 2015. Disponível em: <<http://www.postgresql.org/about/>>. Acesso em: 03/08/2015. Citado na página 29.
- PRESSMAN, R. S. *Engenharia de software*. [S.l.]: The McGraw-Hill Companies, 2011. Citado 2 vezes nas páginas 21 e 34.
- PRIMEFACES. *Primefaces Show Case*. 2016. Disponível em: <<http://www.primefaces.org/showcase/>>. Acesso em: 13/09/2016. Citado 2 vezes nas páginas 27 e 28.
- REZENDE, D. A. *Engenharia de Software e Sistemas de Informação*. [S.l.]: Brasport, 2005. (3º ed.). Citado 2 vezes nas páginas 81 e 82.
- SEIFFERT, J. A. d. S. C. P. Q. *Estruturação organizacional: Planejamento e Implantando uma Nova Estrutura*. [S.l.]: São Paulo: Atlas, 2007. Citado na página 16.
- SOMMERVILLE, I. *Engenharia de software*. [S.l.]: ADDISON WESLEY, 2007. (8º ed.). Citado na página 38.
- SOMMERVILLE, I. *Engenharia de software*. [S.l.]: Pearson Prentice Hall, 2011. (9º ed.). Citado na página 20.
- WEISSMANN, H. L. *Vire o Jogo com Spring Framework*. [S.l.]: Casa do Código, 2012. Citado 5 vezes nas páginas 7, 28, 69, 70 e 72.
- ZEMEL, T. *Web Design Responsivo - Páginas adaptáveis para todos os dispositivos*. [S.l.]: Casa do Código, 2012. (1º ed.). Citado 3 vezes nas páginas 24, 25 e 80.

Apêndices

APÊNDICE A – implantacao

A.1 Servidor

Com o objetivo de disponibilizar o sistema em nuvem, foi contratado um Servidor Virtual Privado (VPS) da "DigitalOcean" que disponibiliza maquinas virtuais, chamadas de *Droplets*, como serviços que podem ser contratados individualmente, escolhendo itens de *hardware* como quantidade de memória RAM, espaço em disco e taxa de transferência de rede.

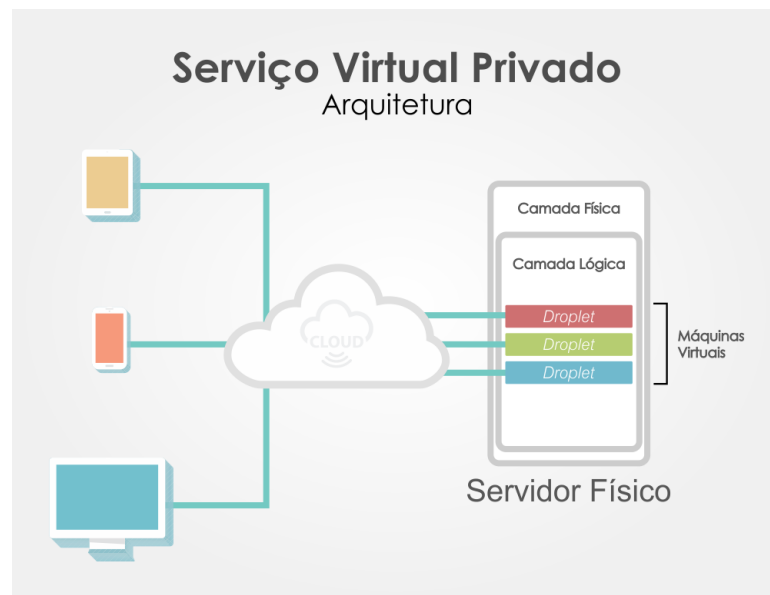


Figura 15 – Serviço Virtual Privado

No momento da criação de um *droplet*, selecionamos o sistema operacional que será utilizado, o local onde está o servidor real, escolher o conjunto de *hardware* que o *droplet* terá, e adicionar uma chave *Secure Shell* (SSH).

Apartir do contrato de um servidor virtual podemos acessar nosso *droplet* através de um cliente SSH e termos disponível um terminal para então começarmos a configurar os serviços que o sistema precisa para funcionar.

A.2 Serviços

No *droplet* que foi criado para comportar a aplicação, foram instalados e configurados os pacotes de *softwares* necessários para levantar os serviços essenciais para o seu

funcionamento, onde foi possível instanciar os serviços de Banco de Dados com Postgres, para armazenar os dados da aplicação, e de *Web Container* com Apache Tomcat, que armazena e executa os binários da aplicação, disponibilizando o sistema para ser acessado de qualquer lugar através da internet.

A.3 Aplicação em Produção

Como foi exposto no capítulo de Implantação, a aplicação deste projeto foi disponibilizada em um Servidor Virtual Privado, mais precisamente em uma máquina virtual chamada *droplet* que foi configurada com a distribuição Linux Ubuntu 16.04 LTS (Xenial Xerus), X64, com memória RAM de 1Gb e capacidade de Disco de 20Gb.

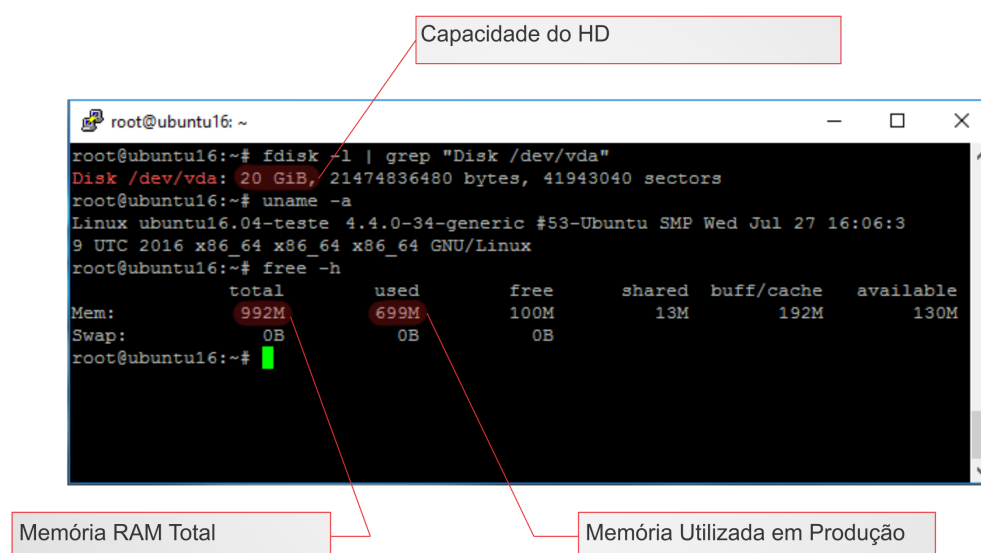


Figura 16 – Informações do Sistema em Produção

APÊNDICE B – Telas do Sistema



Figura 17 – Tela de Login



Figura 18 – Dashboard

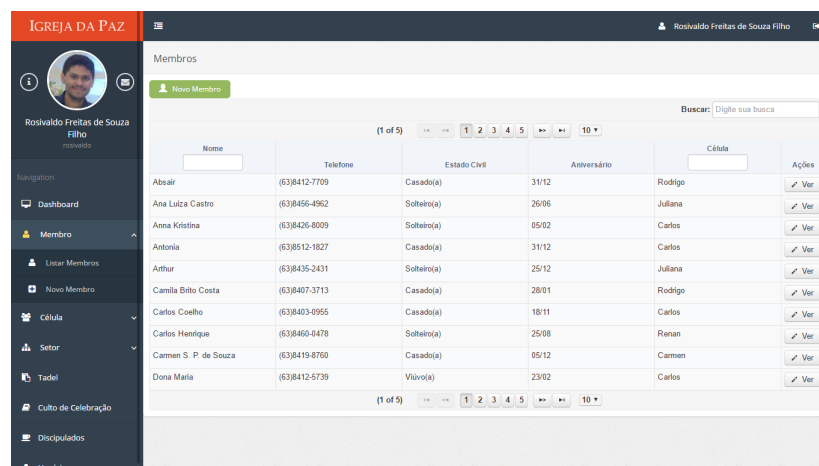


Figura 19 – Lista de Membros

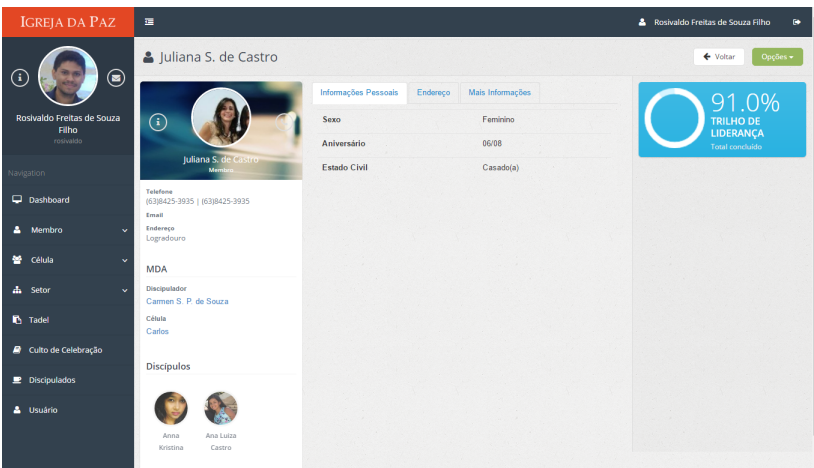


Figura 20 – Tela de Detalhamento de Membro

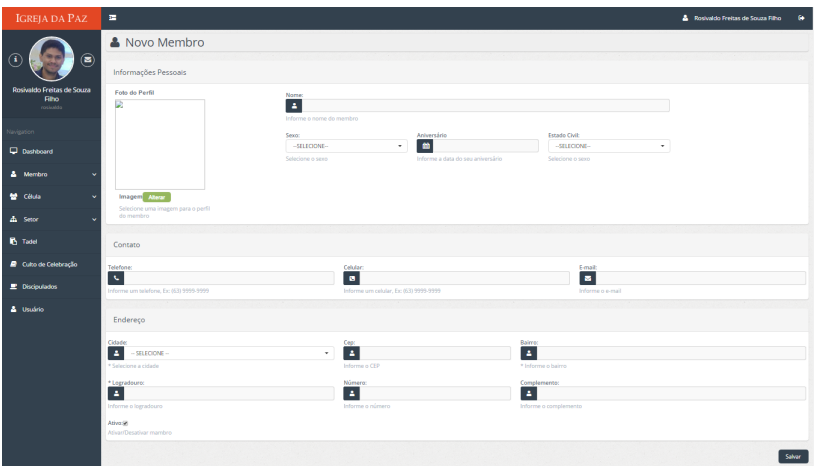


Figura 21 – Tela de Cadastro de Membro

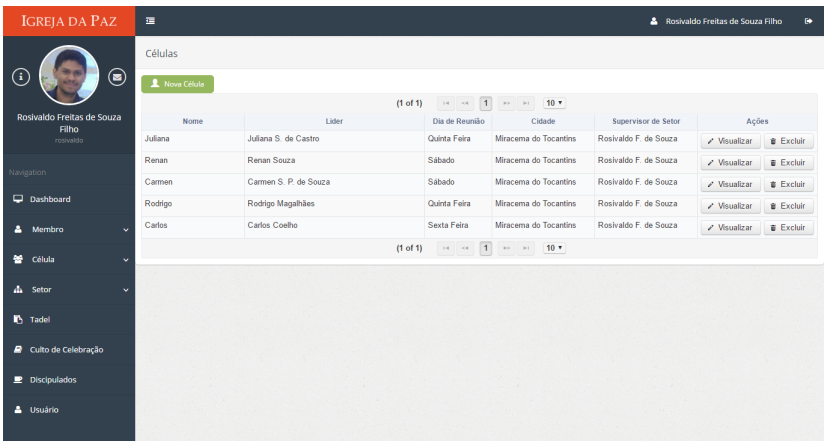


Figura 22 – Tela de Lista de Células

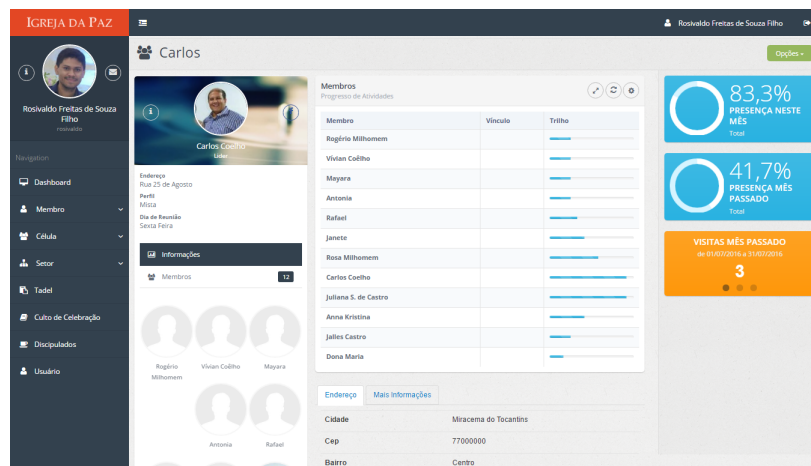


Figura 23 – Tela de Detalhamento de Célula

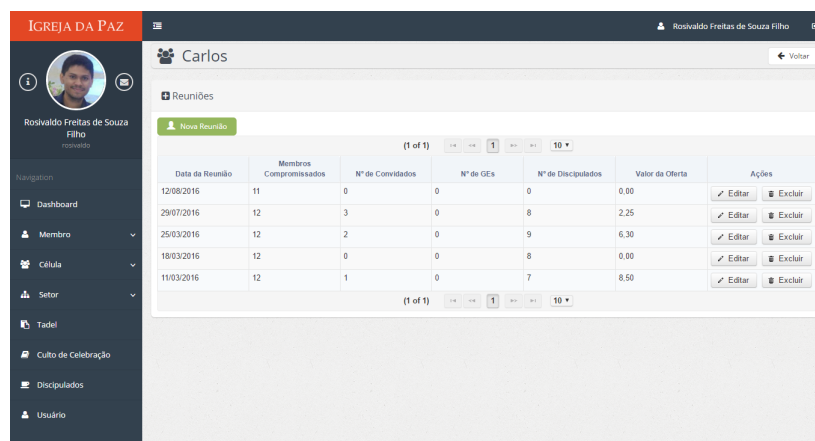


Figura 24 – Tela de Reuniões de Célula

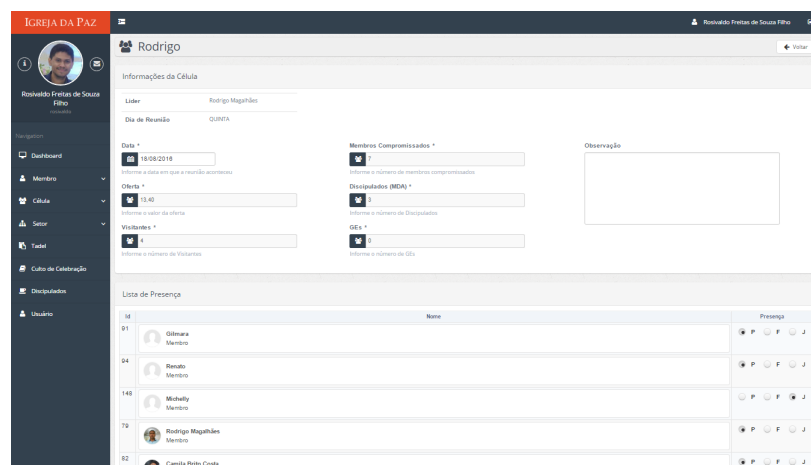


Figura 25 – Tela de Cadastro de Reunião de Célula

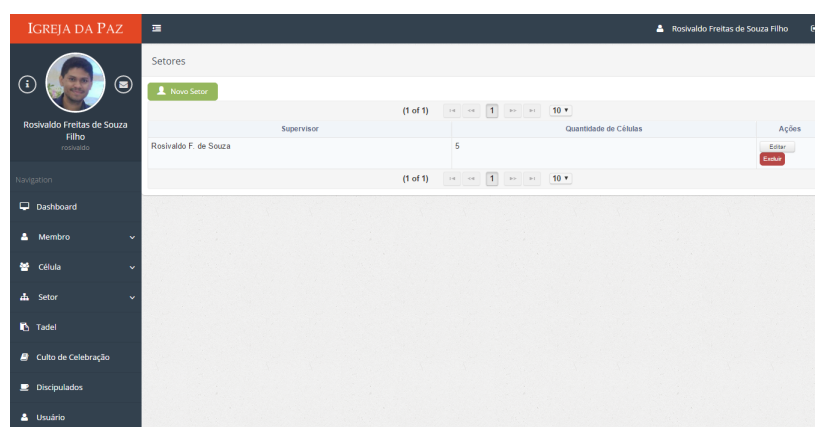


Figura 26 – Tela de Lista de Setores

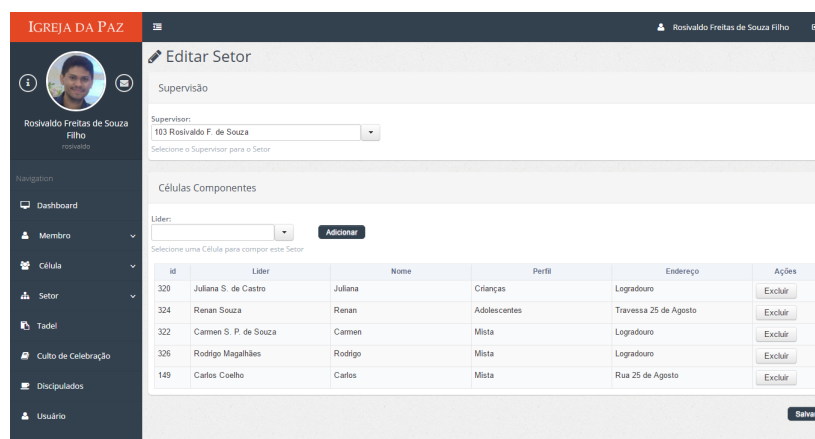


Figura 27 – Tela de Edição de Setores

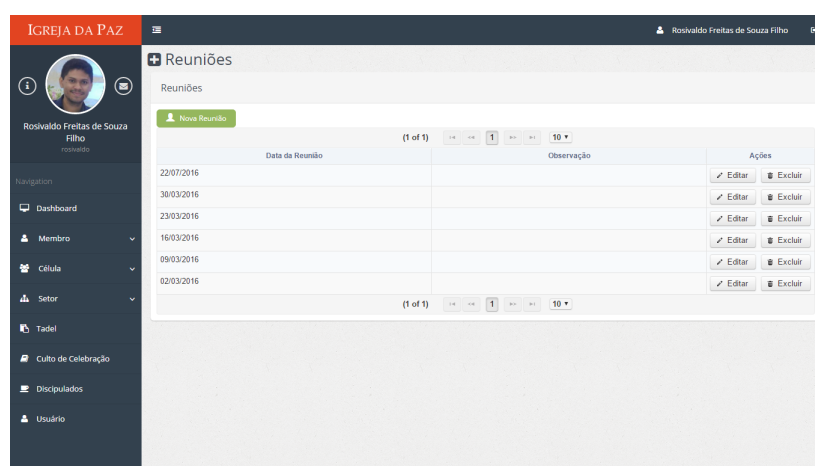


Figura 28 – Tela de Lista de Reuniões de Treinamento de Lideres (TADEL)

IGREJA DA PAZ

Rosivaldo Freitas de Souza Filho

22/07/2016

Informações da Reunião

Data * 22/07/2016

Observação

Informe a data em que a reunião aconteceu

Voltar Salvar

Figura 29 – Tela de Edição de TADEL

IGREJA DA PAZ

Rosivaldo Freitas de Souza Filho

Cultos

Registros

Novo Culto

Data da Reunião	Observação	Ações
27/03/2016		Editar Excluir
20/03/2016		Editar Excluir
13/03/2016		Editar Excluir
06/03/2016		Editar Excluir

Voltar

Figura 30 – Tela de Lista de Cultos

IGREJA DA PAZ

Rosivaldo Freitas de Souza Filho

27/03/2016

Informações da Reunião

Data * 27/03/2016

Observação

Informe a data em que a reunião aconteceu

Voltar Salvar

Figura 31 – Tela de Edição de Culto

Discipulo	Discipulador	Dia de Reunião	Ações
Absair	Rosivaldo F. de Souza	Terça Feia	Ver
Ana Luiza Castro	Juliana S. de Castro	Domingo	Ver
Anna Kristina	Juliana S. de Castro	Segunda Feia	Ver
Camila Brito Costa	Camren S. P. de Souza	Segunda Feia	Ver
Carlos Coelho	Rosivaldo F. de Souza	Segunda Feia	Ver
Carlos Henrique	Guilherme	Terça Feia	Ver
Jalles Castro	Rosivaldo F. de Souza	Quarta Feia	Ver
Juliana S. de Castro	Camren S. P. de Souza	Segunda Feia	Ver
Renan Souza	Rosivaldo F. de Souza	Quinta Feia	Ver
Rosa Milhomem	Camren S. P. de Souza	Quarta Feia	Ver

Figura 32 – Tela de Lista de discipulados

Figura 33 – Tela de Detalhamento de Discipulado

Figura 34 – Tela de Nova Reunião de Discipulado

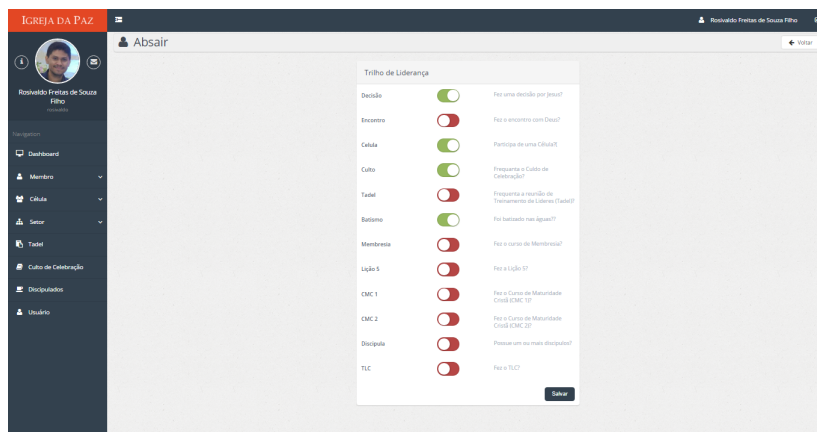


Figura 35 – Tela de Trilho de Liderança

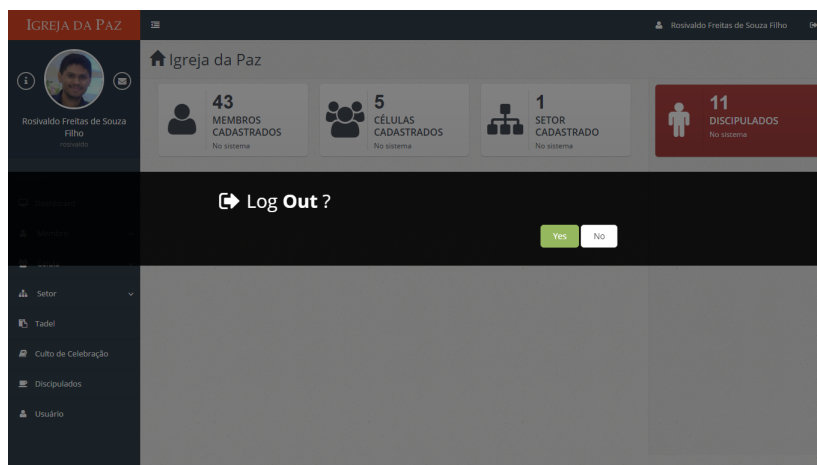


Figura 36 – Tela de Logout

Anexos

C ANEXO A - Formulário de Célula

Relatório Mensal de Célula

Líder: FÁBIO E CARLA Mês: FEVEREIRO

Nome (completo)	Telefone (celular)	Aniversário (data)	MDA (disciplinador)	Semana 1					Semana 2					Semana 3					Semana 4					Semana 5				
				TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC	TD	CE	CC				
1. Arsenio do Nasc.	846784	19.07	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
2. Carla de Andrade S.	8409 03	14.05	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
3. Cláudia Vilarinho	844 376	27.05	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
4. Dilce do Nascimento	8477 87	09.05	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
5. Fábio El Afouni L.	922853	31.01	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
6. Fernanda	81316 7	08/02	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
7. Humberto	841 880	26/02	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
8. Jaqueline	9211 53		S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
9. Leonardo	81313 0	03/01	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
10. Lucas L. de Castro	941445	23.01	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
11. Lucimar de Oliveira	84228 6		S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
12. Ludmila	9987 08	19/03	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
13. Marilene das S. D.	843 501	27.11	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
14. Marlini	9220 19	24/07	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
15. Pedro H. Barbosa	84547 0		S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
16. Peterson H. G.	848115	28.03	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
17. Sara Fernandes	84549 8	22.05	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
18. Tatiana Caelano	8403 80	27.08	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
19. Whelton C. do Nasc.	846 516	06.03	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
20. Theovany	9228 84	22/09	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
21. Anderson	923746	13/04	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					
22. Tulo	9280188	13/04	S	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P					

Observações: _____

Legenda: CE - Célula CC - Culto de Celebração TD - Tadel Preencher todos os espaços com: P (presença), F (faltou), J (justificado)

Semana 1: 10 Semana 2: 00 Semana 3: 02 Semana 4: 00 Semana 5: 00
 Semana 1: 18 Semana 2: 16 Semana 3: 14 Semana 4: 19 Semana 5: 18

Dizimista fiel? (X) Sim () Não

Figura 37 – Formulário de Célula Atual Preenchido à mão

Este é o formulário padrão que os líderes de célula preenchem para entregar aos supervisores de setor.