



**INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO E  
COMUNICAÇÃO**

**SISTEMA DE GESTÃO DE PLANO DE TRABALHO PARA A  
DIREÇÃO DO INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO E  
COMUNICAÇÃO (INSTIC)**

**TRABALHO DE FIM DE CURSO DE LICENCIATURA EM  
ENGENHARIA DE INFORMÁTICA.**

**AUTORES:**

**KAPINGE NETO DOS SANTOS DE ALMEIDA**

**E**

**IVÂNIA PATRICIA CAMBACA QUIOSA**

**Luanda, 2021**



**INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO E  
COMUNICAÇÃO**

**SISTEMA DE GESTÃO DE PLANO DE TRABALHO PARA A  
DIREÇÃO DO INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO E  
COMUNICAÇÃO (INSTIC)**

**TRABALHO DE FIM DE CURSO DE LICENCIATURA EM  
ENGENHARIA DE INFORMÁTICA.**

**AUTORES:**

**KAPINGE NETO DOS SANTOS DE ALMEIDA**

**E**

**IVÁLIA PATRICIA CAMBACA QUIOSA**

**ORIENTADORES:**

**MSc, Yanelis Benítez Fernández.**

**Eng. Yubismel Perdomo Velázquez**

**Luanda, 2021**

## **Dedicatória**

Tendo em conta que ao longo do nosso percurso como estudantes nunca estivemos sozinhos. Estamos e sempre estivemos na companhia de pessoas que querem o nosso melhor e que nos ajudam a alcançar os nossos objetivos. Por isso, dedicamos este trabalho a todas as pessoas especiais em nossas vidas em especial aos nossos pais, professores, colegas, amigos e a todos que direta ou indiretamente contribuíram de alguma forma para a realização do mesmo.

Nossa dedicatória estende-se também á todos estudantes que pelo atrevimento pretendem envolver-se em projetos que vão para além de atividades que estão relacionadas com a sua formação e ambiente de trabalho ou outro tipo de conhecimento prévio.

## **Agradecimentos**

Começamos por agradecer a Deus pela força que tem nos dado pois seria impossível a realização da nossa monografia sem Ele. Os nossos sinceros agradecimentos, vão para todos aqueles que contribuíram, de alguma forma para realização deste projeto, aos nossos pais que financeiramente contribuíram para a elaboração deste projeto, a todos os docentes que durante os cinco anos de aprendizado não pouparam esforço para nos transmitir os seus conhecimentos e que além de serem professores foram amigos e pais para nós.

Também agradecemos às pessoas com quem convivemos ao longo dos 5 anos de curso aqui na instituição, e que nos incentivaram e encorajaram a continuar e que certamente tiveram impacto positivo na nossa formação.

## Resumo

O Instituto de Tecnologia de Informação e Comunicação (INSTIC) é um instituto de ensino superior localizado no município do Rangel bairro KM8 CTT voltado nas áreas das tecnologias, para o auxílio do Plano de Trabalho para Direção, foi desenvolvido um sistema para ajudar no processo de Plano de Trabalho. O Sistema de Plano de Trabalho para a Direção do INSTIC (SGPT) está marcado pelo avanço das Tecnologias de Informação e Comunicação, que tende a melhorar nos processos da Direção do INSTIC, de modo geral é um sistema que tem a capacidade de organizar todas as Tarefas, Relatórios, Reuniões e Actas a serem realizadas pelos Trabalhadores da Direção do INSTIC facilitando o processo de Trabalho.

A presente pesquisa tem o objectivo de propôr a modelagem e o desenvolvimento de uma aplicação web para a Direção do INSTIC. No processo de desenvolvimento foi empregada o paradigma de Programação Orientada a Objectos (POO), gerando documentos segundo a notação Unified Modeling Language (UML) e os artefactos definidos da metodologia OpenUp. Como resultado da análise do problema definido-se uma proposta para a arquitectura do sistema dividida em três camadas: apresentação, aplicação e dados. A camada de apresentação é responsável pela interface com o usuário, requisita serviços ao servidor da aplicação e apresenta os resultados. Na camada de negócios são definidas as regras de negócio. Na camada de dados se encontra a base de dados da aplicação. Conclui-se que a linguagem PHP aliada ao Framework Laravél permite um desenvolvimento ágil com software de boa qualidade.

Palavras-chave: Direção, INSTIC, PHP, Sistema, Plano de Trabalho.

## **Abstract**

The Institute of Information and Communication Technology (INSTIC) is a higher education institute located in the municipality of Rangel, KM8 CTT, in the areas of technologies, to help with the Work Plan for Management, a system was developed to help in the process of Work plan. The Work Plan System for the Board of INSTIC (SGPT) is marked by the advancement of Information and Communication Technologies, which tends to improve the processes of the Board of INSTIC, in general it is a system that has the capacity to organize all the Tasks, Reports, Meetings and Minutes to be carried out by the Workers of the Direction of the INSTIC facilitating the Work process.

This research aims to propose the modeling and development of a web application for the Board of INSTIC. In the development process, the Object Oriented Programming (OOP) paradigm was used, generating documents according to the Unified Modeling Language (UML) notation and the defined artifacts of the OpenUp methodology. As a result of the problem analysis, a proposal for the system architecture was defined, divided into three layers: presentation, application and data. The presentation layer is responsible for the user interface, requests services from the application server and presents the results. In the business layer, business rules are defined. In the data layer is the application database. It is concluded that the PHP language combined with the Laravél Framework allows an agile development with good quality software.

Keywords: Direction, INSTIC, PHP, System, Work Plan.

## Índice Geral.

|  |    |
|--|----|
| 1. Capítulo 1: Fundamentação Teórica.....  | 5  |
| 1.1. Introdução do capítulo.....   | 5  |
| 1.2. Principais conceitos associados.....  | 5  |
| 1.2.1. Sistema.....  | 5  |
| 1.2.2. Gestão .....  | 6  |
| 1.2.3. Sistema de gestão .....   | 6  |
| 1.2.4. Plano de trabalho.....  | 6  |
| 1.3. Estado da arte a do processo de plano de trabalho a nível internacional e nacional. | 7  |
| 1.3.1. Internacional .....   | 7  |
| 1.3.1.1. Google Agenda .....   | 7  |
| 1.3.1.2. Yahoo Agenda.....   | 7  |
| 1.3.2. Nacional.....   | 8  |
| 1.3.3. Justificativa .....   | 8  |
| 1.4. Metodologias de desenvolvimento de software.....                                    | 8  |
| 1.4.1. Metodologias pesadas.....   | 8  |
| 1.4.2. Metodologias ágeis.....   | 9  |
| 1.4.2.1. Extreme Programming (XP) .....  | 10 |
| 1.4.2.2. OpenUp .....  | 12 |
| 1.4.2.3. Scrum .....   | 15 |
| 1.5. Ferramentas e tecnologias nas quais se apoia para a solução.....                    | 16 |
| 1.5.1. Linguagem de programação web .....  | 16 |
| 1.5.1.1. PHP 7.4 .....   | 16 |
| 1.5.1.2. PHP 7.4 vs JSP (Java Server Pages) 8.0 .....                                    | 17 |

|          |  |    |
|----------|--|----|
| 1.5.2.   | Framework de desenvolvimento de software .....                     | 17 |
| 1.5.2.1. | BootsTrap 4.1.3.....   | 18 |
| 1.5.2.2. | Laravel 8.....   | 18 |
| 1.5.2.3. | Laravel 8 vs Symfony 5.4 .....                                     | 19 |
| 1.5.3.   | Editor de texto de código .....                                    | 19 |
| 1.5.3.1. | Visual Studio Code 1.47.1 .....                                    | 19 |
| 1.5.4.   | Ferramentas de modelagem UML .....                                 | 20 |
| 1.5.4.1. | Visual Paradigma 13.1 .....  | 20 |
| 1.5.5.   | Sistema gestor de base de dados MySQL .....                        | 21 |
| 1.6.     | Conclusões parciais do capítulo.....                               | 21 |
| 2.       | Capítulo 2: Características do sistema. ....                       | 23 |
| 2.1.     | Introdução .....   | 23 |
| 2.2.     | Objecto de automação .....   | 23 |
| 2.3.     | Modelagem do Negócio: .....  | 23 |
| 2.3.1.   | Descrição do negócio .....   | 24 |
| 2.3.2.   | Modelo conceptual ou domínio.....                                  | 24 |
| 2.3.3.   | Descrição dos trabalhadores e actores do negócio.....              | 25 |
| 2.3.3.1. | Actor do negócio .....   | 25 |
| 2.3.3.2. | Trabalhador de negócio.....  | 25 |
| 2.3.4.   | Diagrama de Casos de Uso do Negócio.....                           | 26 |
| 2.3.5.   | Descrições de Casos de Uso do Negócio. ....                        | 26 |
| 2.4.     | Modelagem do Sistema:.....   | 28 |
| 2.4.1.   | Requisitos funcionais.....   | 28 |
| 2.4.2.   | Requisitos não-funcionais.....                                     | 29 |
| 2.4.3.   | Diagrama de Casos de Uso do Sistema.....                           | 30 |
| 2.4.4.   | Descrição de Casos de Uso do Sistema. Protótipos de interface..... | 31 |



|  |    |
|--|----|
| 2.5. Conclusões parciais do capítulo.....                      | 35 |
| 3. Capítulo 3: Desenho, Implementação e Provas do Sistema..... | 37 |
| 3.1. Introdução do capítulo.....                               | 37 |
| 3.2. Diagrama de Classes .....                                 | 37 |
| 3.2.1. Diagrama de Classe com estereótipo .....                | 38 |
| 3.3. Estilo arquitetônico .....                                | 39 |
| 3.4. Padrão de desenho utilizado .....                         | 41 |
| 3.4.1. Padrão Grasp .....                                      | 41 |
| 3.4.2. O Padrão GoF .....                                      | 42 |
| 3.5. Análise e desenho do banco de dados .....                 | 42 |
| 3.5.1. Modelo físico de banco de dados .....                   | 42 |
| 3.5.2. Descrição das Entidades .....                           | 43 |
| 3.6. Modelo de despliegue(Implantação) .....                   | 44 |
| 3.6.1. Descrição do modelo de implantação.....                 | 45 |
| 3.7. Validação do Sistema.....                                 | 46 |
| 3.7.1. Provas unitárias.....                                   | 46 |
| 3.7.1.1. Caixa branca .....                                    | 47 |
| 3.7.2. Provas funcionais .....                                 | 48 |
| 3.8. Conclusões parciais do capítulo.....                      | 52 |
| 2. Conclusões .....  | 53 |
| 3. Recomendações. (Folha única) .....                          | 54 |
| 4. Referências bibliográficas.....                             | 55 |

## Índice de Figuras

|   |    |
|---|----|
| Figura 2-1: Modelo de domínio. Fonte: Elaboração própria .....                      | 24 |
| Figura 2-2: diagrama de caso de uso do negócio. Fonte: Elaboração própria .....     | 26 |
| Figura 2-3: Diagrama de Casos de Uso do Sistema. Fonte: Elaboração própria .....    | 31 |
| Figura 3-1: Diagrama de classe. Fonte: Elaboração própria .....                     | 38 |
| Figura 3-2: Diagrama de classe com estereótipo. Fonte: Elaboração própria .....     | 39 |
| Figura 3-3: Diagrama Modelo Vista Controlador (MVC). Fonte: (Vilarinho, 2018).....  | 40 |
| Figura 3-4: Modelo físico de banco de dados. Fonte: Elaboração própria .....        | 43 |
| Figura 3-5: modelo de implantação. Fonte: Elaboração própria.....                   | 44 |
| Figura 3-6: Código do teste de caixa branca. Fonte: Elaboração própria .....        | 47 |
| Figura 3-7: Gráfico de fluxo, teste de caixa branca. Fonte: Elaboração própria..... | 48 |
| Figura 3-8: prova funcional de cadastrar usuário. Fonte: Elaboração própria .....   | 50 |
| Figura 3-9: prova funcional de cadastrar usuário. Fonte: Elaboração própria .....   | 50 |
| Figura 3-10: prova funcional de cadastrar usuário. Fonte: Elaboração própria .....  | 51 |
| Figura 3-11: Resultados das provas. Fonte: Elaboração própria .....                 | 52 |

## Índice de Tabelas

|   |    |
|---|----|
| Tabela 1: Comparação entre metodologias (Almeida, 2017) .....                           | 9  |
| Tabela 2: Actores de negócio.....   | 25 |
| Tabela 3: Descrição dos trabalhadores e atores do negócio .....                         | 25 |
| Tabela 4: Descrição do caso de uso para o negócio “Enviar notificação de reunião” ..... | 27 |
| Tabela 5: Descrição do caso de uso para o negócio “Elaborar Acta” .....                 | 27 |
| Tabela 6: Requisitos funcionais.....  | 28 |
| Tabela 7: Requisitos não funcionais.....  | 30 |
| Tabela 8: Descrição dos actores do sistema.Fonte: Elaboração própria .....              | 31 |
| Tabela 9: Descrição do caso de uso “Criar reunião” .....                                | 31 |
| Tabela 10: Descrição do caso de uso “Cadastrar Usuário” .....                           | 33 |
| Tabela 11: Descrição do caso de uso “Anexar acta após reunião” .....                    | 34 |
| Tabela 12: descrição das entidades.....   | 43 |
| Tabela 13: descrição do modelo de implantação .....                                     | 45 |
| Tabela 14: descrição dos conectores usados no modelo de implantação .....               | 45 |
| Tabela 15: Provas de Caixa Preta.....   | 49 |

## Lista de Abreviaturas e Símbolos

|              |  |
|--------------|--|
| INSTIC ..... | Instituto de Tecnologia de Informação e Comunicação                                    |
| RAD.....     | Rapid Application Development  |
| RUP.....     | Rational Unified Process   |
| XP.....      | Extreme Programming  |
| UML.....     | Unified Modeling Language  |
| PHP.....     | Hypertext Preprocessor   |
| API.....     | Application Program Interface (Interface do programa de Aplicação)                     |
| ASP.....     | Active Server Pages  |
| JSP.....     | Java Server Pages  |
| MVC.....     | Modelo Vista Controlador   |
| SGBD.....    | Sistema de Gerenciamento de Banco de Dados   |
| VS Code..... | Visual Studio Code   |
| RF.....      | Requisitos Funcionas   |
| RNF.....     | Requisitos não Funcionas   |
| PDO.....     | PHP Data Object  |
| HTTPS...     | Hyper Text Transfer Protocol Secure ( Protocolo de Transferência de Hipertexto Seguro) |
| GOF.....     | Gang of four   |

## **Introdução**

Com a evolução tecnológica no mundo e principalmente no nosso país Angola, torna-se necessário as instituições acompanharem passo a passo esta evolução, isto é, fazendo o uso das novas tecnologias para obterem o máximo de segurança, eficiência e eficácia nas suas tarefas diárias.

Geralmente as instituições guardam as suas informações em papéis, usando métodos não convencionais ou não tecnológicos o que faz com que as essas mesmas informações sejam facilmente perdidas, danificadas, que tenha falta de segurança e grandes volumes de papéis, etc.

O Instituto de Tecnologia de Informação e Comunicação (INSTIC) é uma instituição pública de ensino superior, vocacionada para a promoção do ensino e investigação científica, para criação, transmissão e difusão da cultura, ciência e tecnologia em prol da sociedade Angolana, particularmente da comunidade em que está inserida.

### **MISSÃO**

O INSTIC tem como missão o desenvolvimento de actividades de ensino, investigação científica e prestação de serviços à comunidade, através da promoção, difusão, criação, transmissão da ciência e cultura, bem como a promoção e realização da investigação científica.

### **OBJECTIVOS**

O INSTIC tem como objectivo primordial dar todo o suporte necessário à criação de uma Instituição do Ensino Superior que forme quadros técnicos no domínio das tecnologias da informação, com excelência.

## **Situação problemática**

Fazer a gestão de plano de trabalho é de extrema importância para as organizações. Devido a frequência com que as reuniões acontecem, possuir estratégias de agendamento de reunião eficaz de modo a tornar cada vez mais productiva as reuniões, é muito importante para que as organizações obtêm bons resultados.

O Instituto de Tecnologias de Informação e comunicação (INSTIC) é uma instituição pública de ensino superior, situada no distrito urbano do Rangel em Luanda.

Todo o processo de gestão de Plano de Trabalho do INSTIC é feito manualmente, utilizando papéis, provocando problemas como: Dificuldade ao enviar convocatória; Dificuldade para disponibilizar aos participantes os materiais a serem usados antes da reunião acontecer; Lentidão no aprovação de acta; Dificuldade em Localizar uma determinada acta.

Tendo em conta o que se levantou anteriormente se identifica como **problema de investigação**: Como melhorar o processo de gestão de Plano de Trabalho para Direção do INSTIC?

Segundo ao problema identificado, se define como **objeto de estudo**: Sistema de Gestão de Plano de Trabalho para Direção do INSTIC, centrado no **campo de acção**: Processo de Gestão de Plano de Trabalho para Direção do INSTIC.

Para dar resposta ao problema identificado, se define como o **objetivo geral** deste trabalho: Desenvolver um sistema informático que permite facilitar a gestão de Plano de Trabalho para a Direção do INSTIC.

Para dar cumprimento ao objectivo geral, se consideram os seguintes **objectivos específicos**:

1. Elaborar o marco teórico da investigação mediante um estudo do estado da arte sobre a gestão de Plano de trabalho.
2. Selecionar a metodologia, ferramentas e tecnologias a usar no desenvolvimento do sistema informático para a gestão de Planio de Trabalho.
3. Definir os requisitos funcionais e não funcionais do sistema informático para a gestão de Plano de Trabalho.
4. Desenhar o sistema informático para a gestão de Plano de Trabalho para a Direção do INSTIC.
5. Implementar o sistema informático para a gestão de Plano de Trabalho para a Direção do INSTIC.
6. Validar o sistema informático mediante provas de softwares.

Para dar solução ao problema levantado se propõe as seguintes **tarefas de investigação**:

1. Estudo dos sistemas similares no âmbito nacional e internacional que realizem a gestão de Plano de Trabalho.
2. Seleção e caracterização da metodologia, e das ferramentas e tecnologias a utilizar no desenvolvimento do sistema informático.
3. Definição de requisitos funcionais e não funcionais que dão solução ao problema de investigação.
4. Desenhar os modelos do sistema informático de acordo com a metodologia de desenvolvimento de software selecionada.
5. Implementação do sistema informático para a gestão de Plano de Trabalho para a Direção do INSTIC.
6. Realização das provas de software para validar o comportamento do sistema informático.

## **Métodos de investigação**

Para o desenvolvimento da investigação foram aplicados os seguintes métodos de investigação:

### **Métodos Teóricos:**

- **Modelação:** Se utilizou para representar os processos definidos para o sistema mediante a construção de modelos e diagramas ao longo do desenvolvimento da investigação, simplificando a realidade e facilitando a compreensão do mesmo.
- **Analítico-Sintético:** A aplicação deste método permitiu a busca, investigação e análise dos documentos necessários e sites na internet para entender o processo que se leva a cabo para a gestão de Plano de Trabalho para a direção do INSTIC.

### **Métodos Empíricos:**

- **Entrevista:** realizaram-se aos funcionários da Direção do INSTIC, que se vêm imersos no processo de gestão de Plano de Trabalho da Direção do INSTIC.
- **Questionário:** fez-se aos funcionários da Direção do INSTIC, uma série de perguntas sobre como é o processo de Plano de Trabalho a fim de se encontrar as dificuldades durante todo o processo e consequentemente encontrar uma solução.

## Resumo

A presente investigação é composta pela Introdução, três capítulos, conclusões, recomendações, referências bibliográficas e anexos por três capítulo e a Conclusão, a seguir são descritos os principais aspectos abordados em cada um dos capítulos.

**Capítulo 1: Fundamentação teórica:** Neste capítulo especificam-se os principais conceitos relacionados com o tema, realiza-se o estudo do estado da arte a nível nacional e internacional sobre o sistema de Plano de Trabalho. Também se definem as ferramentas, metodológicas e tecnológicas a utilizar durante o desenvolvimento do sistema.

**Capítulo 2: Características do Sistema:** Desenho do Sistema de Plano de Trabalho. Neste capítulo se expõe uma descrição geral da solução proposta e seu funcionamento, capturam-se os requisitos funcionais e não funcionais e outros artefatos do desenho como os casos de uso e sua descrição.

**Capítulo 3: Desenho, implementação e provas do sistema:** Neste capítulo se abordam aspetos relacionados com a implementação do sistema em base à arquitetura de desenvolvimento de software. Documentam-se as provas de software ao sistema, realizando-se as provas unitárias e de aceitação para verificar que responda a um correto funcionamento, garantir a confidencialidade e detectar falhas no sistema.



# 1. Capítulo 1: Fundamentação Teórica

## 1.1. Introdução do capítulo

Neste capítulo é feita a apresentação sobre a descrição geral do objeto de estudo, domínio actual do problema, principais conceitos relacionados ao tema, realiza-se também um estudo mais abrangente sobre as ferramentas, tecnologias, sistemas existentes a nível internacional e nacional, bem como a metodologia de desenvolvimento de software a ser utilizada para dar resposta ao problema existente.

## 1.2. Principais conceitos associados.

### 1.2.1. *Sistema*

(OLIVEIRA, 2002) Conceitua sistema como “um conjunto de partes interagentes e interdependentes que, conjuntamente, formam um todo unitário com determinado objetivo e efetuam determinada função”.

Reforçando essa ideia, (BATISTA, 2004) entende que sistema corresponde a “disposição das partes de um todo que, de maneira coordenada, formam a estrutura organizada, com a finalidade de executar uma ou mais atividades ou, ainda, um conjunto de eventos que repetem ciclicamente na realização de tarefas predefinidas”.

Segundo (REZENDE, et al., 2000), em geral os sistemas procuram atuar como:

- Ferramentas para exercer o funcionamento das empresas e de sua intrincada abrangência e complexidade;
- Instrumentos que possibilitam uma avaliação analítica e, quando necessária, sintética das empresas;
- Facilitadores dos processos internos e externos com suas respectivas intensidades e relações;
- Meios para suportar a qualidade, produtividade e inovação tecnológica organizacional;
- Geradores de modelos de informações para auxiliar os processos decisórios empresariais;
- Produtores de informações oportunas e geradores de conhecimento;

Valores agregados e complementares à modernidade, perenidade, lucratividade e competitividade empresarial.

### **1.2.2. Gestão**

De acordo com (RODRIGUEZ, 2010), gestão é a forma que os relacionamentos entre as pessoas acontecem, na busca de um objectivo comum.

(BARBARÁ, 2008), define gestão como um conjunto de atividades coordenadas para dirigir e controlar um grupo de pessoas e instalações com responsabilidade, autoridade e relações definidas.

Como gestão se subentende que é o que leva a organizar, dispor, dirigir e dar uma ordem para que se consiga um determinado objetivo. Utiliza-se a gestão para orientar a resolver um problema específico, a concretizar um projeto.

### **1.2.3. Sistema de gestão**

Segundo (CALDEIRA, 2011), O objectivo principal de um sistema de informação é o fornecer aos seus utilizadores informação em tempo útil e num formato adequado e compreensível.

Para a (Fundação Nacional da Qualidade, 2018), “Sistema de Gestão é um conjunto de práticas padronizadas, logicamente inter-relacionadas com a finalidade de gerir uma organização e produzir resultados. O Sistema de Gestão da organização abrange todos os seus subsistemas de gestão, composto por práticas. O sistema de gestão costuma ser um emaranhado de práticas de gestão que interagem entre si, produzindo resultados financeiros ou não”.

### **1.2.4. Plano de trabalho**

Afirma (Editorial Conceitos, 2016), que “entende-se por plano de trabalho o conjunto de ações que serão realizadas em um negócio ou departamento qualquer com o fim de atingir certos objetivos”.

Plano de trabalho é um instrumento que será avaliado após a efetivação do cadastro do proponente e deverá conter, no mínimo:

- Justificativa para a celebração do convênio;
- Descrição completa do objeto a ser executado. Devem ser descritos os objetivos a curto e médio prazos e os produtos esperados;

- Descrição das metas a serem atingidas, definindo as etapas ou fases da execução;

Cronograma de execução do objeto, cronograma de desembolso e plano de aplicação dos recursos a serem desembolsados pelo concedente. (Universidade Federal de Uberlândia, 2020)

### 1.3. Estado da arte a do processo de plano de trabalho a nível internacional e nacional.

#### 1.3.1. *Internacional*

A nível internacional foram encontrados dois softwares que possuem similaridades com o software em estudo, estes softwares são: **Google Agenda** e **Yahoo Agenda**.

##### 1.3.1.1. *Google Agenda*

É um serviço do Google de agenda virtual que pode ser acessada pelo navegador ou pelo aplicativo para Android e IOS. Essa ferramenta tem como objetivo aumentar a produtividade das pessoas em suas vidas pessoais e profissionais, além de facilitar para gestores e directores.

Ou seja, a agenda do Google ajuda a diminuir os custos com gestão ao permitir a organização do dia a dia com muito mais agilidade e eficácia. Entre seus muitos diferenciais a facilidade de uso e sua gratuidade são as que chamam mais atenção. Além disso, não existe a necessidade de instalação para o acessar por meio do computador, basta fazer o login em sua conta Google pelo seu navegador.

##### 1.3.1.2. *Yahoo Agenda*

O Yahoo Calendar é o serviço de calendário de usuários do Yahoo. Trata-se de um serviço online e gratuito de agenda eletrônica virtual que permite simplificar na organização de todos os afazeres, permite cadastrar compromissos para que o usuário não se esqueça de nenhum evento importante, facilitando a vida na internet.

Com versão web e compatibilidade com a maioria dos clientes de correio eletrônico no computador e em dispositivos móveis, ele permite criar e importar calendários em categorias pré-definidas como Feriados e Esportes. Além disso, oferece uma lista de tarefa

integrada que permite listar compromissos como Urgente, Importante e Normal, entre outras funções.

### **1.3.2. Nacional**

A nível nacional não foi encontrado softwar que possui similaridade com o software em estudo.

### **1.3.3. Justificativa**

Os dois sistemas não se enquadram as necessidades da direção do instituto de tecnologias de informação e comunicação pois os dois não permitem anexar uma acta após uma reunião a fim de todos os participantes poderem aprovar a mesma.

## **1.4. Metodologias de desenvolvimento de software**

Metodologia de desenvolvimento entende-se o conjunto das atividades, responsabilidades, artefactos (documentos, diagramas, código-fonte, etc.), orientações e boas práticas usadas para planejar, construir e implantar software (MACHADO, 2015).

De acordo com (Monitora, 2020), O sucesso de qualquer projeto voltado à elaboração de software depende diretamente da escolha da metodologia mais adequada.

(Monitora, 2020), afirma que “As metodologias de desenvolvimento de software consistem, basicamente, no conjunto de abordagens que podem ser utilizadas para a criação de sistemas de processamento de dados”.

Segundo (SOMMERVILLE, 2011), um modelo de processo de software, é uma representação abstrata, mais simples, de um processo de software. Os modelos de processo, incluem as atividades que fazem parte do processo.

Uma metodologia define estados, etapas e fases de desenvolvimento. Também define tarefas, e atividades. As metodologias podem ser pesadas ou ágeis.

### **1.4.1. Metodologias pesadas**

As metodologias tradicionais, são também chamadas de pesadas ou orientadas a documentação. Surgindo com o intuito, de acabar com os insucessos no desenvolvimento de software. O objetivo principal das metodologias tradicionais, é permitir a previsão dos

requisitos do sistema, que tem como propósito, tornar os projetos completamente planejados, facilitando a organização dos mesmos. (PEREIRA, 2018)

Segundo (PRESSMAN, 2016), a metodologia tradicional tem como principal característica a total documentação do software antes de sua implementação e não são acessíveis a mudanças ao longo da produção. De acordo com (OLIVEIRA, 2018), esta característica descrita, é a principal limitação dos modelos tradicionais.

Tipos de metodologias pesadas: Modelo cascata, Modelo incremental, RAD (Rapid Application Development), e o RUP (Rational Unified Process)

#### **1.4.2. Metodologias ágeis**

A ideia de desenvolvimento rápido e ágil de software, foi apoiada no ano de 1980, no entanto, o conceito realmente decolou no final da década de 1990, com o desenvolvimento da noção de abordagens ágeis. (SOMMERVILLE, 2011)

Corroborando essa ideia, (ZENARO, 2012), afirma também que as metodologias ágeis, surgiram na década de 90, mas declara que “foi como uma revolução dos métodos tradicionais de desenvolvimento de software”.

segundo (SOMMERVILLE, 2011), o objetivo dessas metodologias ágeis é, “reduzir a burocracia do processo, evitando qualquer trabalho de valor incerto de longo prazo e também, evitar qualquer documentação que provavelmente nunca será usada”.

Na Tabela 1, Segundo o (ALMEIDA, 2017) compara a metodologia tradicional e metodologia ágil:

Tabela 1: Comparação entre metodologias (Almeida, 2017)

| <b>Metodologia tradicionais</b>  | <b>Metodologias ágeis</b>   |
|--|---|
| O trabalho é desenvolvido em grandes grupos de pessoas, às vezes distribuídas. | O trabalho se concentra em grupos de poucas pessoas.  |
| O cliente não faz parte da equipe de desenvolvimento, ele está presente        | O cliente faz parte da equipe e do processo de desenvolvimento com reuniões frequentes e até informais. |

|  |   |
|--|---|
| apenas através de reuniões planejadas.                                       |   |
| Alguma resistência a mudanças  | Alterações podem aparecer no processo de desenvolvimento e são tomadas, se necessário                       |
| Os projectos apresentam uma grande quantidade de documentação e detalhamento | A documentação é gerada quando se torna necessária e, em geral, não é extensa ou rigorosa em sua estrutura. |

Dada a comparação acima e levando em conta a todos os elementos levantados em ambas metodologias, a metodologia ágil é considerada e definida para a solução proposta. Essa decisão se deve ao fato de uma pequena equipe de desenvolvimento, o cliente é uma parte fundamental da equipe e está disposto a manter uma comunicação fluida e constante durante o processo de desenvolvimento. Além disso, as partes interessadas estarão cientes de todas as mudanças que podem ser geradas à medida que a execução das tarefas progride, tomando as decisões necessárias para o bem do Sistema.

Exemplo de metodologias ágeis: xp, scrum e *openUp*.

#### **1.4.2.1. Extreme Programming (XP)**

Segundo o (DEV MEDIA, 2018), faz uma explanação sobre a Metodologia XP: A Extreme Programming (XP) é uma Metodologia Ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente. Entre as principais diferenças da XP em relação às Metodologias Clássicas estão o feedback constante, a abordagem incremental e o encorajamento da comunicação entre as pessoas.

A maioria das regras da XP causa surpresa no primeiro contato e muitas não fazem sentido se aplicadas isoladamente. É a força de seu conjunto que sustenta o sucesso da XP, trazendo uma verdadeira revolução no desenvolvimento de software. O principal objetivo da XP é dar agilidade ao desenvolvimento do projeto e busca garantir a satisfação do cliente. As práticas, regras, e os valores da XP garantem um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro princípios básicos:

- **Princípio da Comunicação** - Busca manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação.
- **Princípio da Simplicidade** - Entende-se como simplicidade, a busca do objetivo de implementar o software com o menor número possível de classes e métodos. Outra ideia importante deste princípio é procurar implementar apenas requisitos atuais, evitando assim adicionar funcionalidades que podem ser importantes apenas no futuro. A aposta da XP é que é melhor fazer algo simples hoje do que implementar algo complicado hoje que talvez não venha a ser usado.
- **Princípio do Feedback** - A prática do feedback constante significa que o desenvolvedor terá informações constantes do código e do cliente. A informação do código é dada pelos testes constantes, que indicam os erros tanto individuais quanto do software integrado.
- **Princípio da Coragem** - Sabe-se que não são todas as pessoas que possuem facilidade de comunicação e têm bom relacionamento interpessoal, este princípio também dá suporte à simplicidade, pois assim que a oportunidade de simplificar o software é percebida, a equipe pode experimentar e buscar novas soluções, além disso, é preciso coragem para obter e cobrar constantemente um feedback do cliente.

#### **Principais práticas da Extreme Programming (XP):**

- **Planejamento** - Define o que é ou não necessário ser feito no projeto.
- **Entregas Frequentes** - Baseiam-se no desenvolvimento de um software simples, e conforme os requisitos aparecem, há a atualização da versão do software.
- **Metáfora** - São as descrições de um software sem a utilização de termos técnicos com o objetivo de guiar o desenvolvimento do software com a maior transparência possível para o cliente.

- **Projecto simples** - Deve ser o mais simples possível e satisfazer os requisitos atuais, sem a preocupação de requisitos futuros.
- **Testes** - A Extreme Programming (XP) prioriza a validação do projecto durante todo o processo de desenvolvimento.
- **Programação em pares** - A implementação do código é feita em dupla, ou seja, dois desenvolvedores trabalham em um único computador.
- **Refatoração** - Focaliza a lapidação do projecto do software e está presente em todas as etapas do desenvolvimento.
- **Propriedade coletiva** - O código do projecto pertence a todos os membros da equipe.
- **Integração contínua** - É a prática de interagir e construir o sistema de software várias vezes por dia, mantendo os programadores em sintonia, além de possibilitar processos rápidos.
- **40 horas de trabalho semanal** - A XP assume que não se deve fazer horas extras constantemente.
- **Cliente presente** - É fundamental a participação do cliente durante todo o desenvolvimento do projeto.
- **Código padrão** - Baseia-se na padronização da arquitetura do código, para que este possa ser compartilhado entre todos os programadores e até mesmo entre outros softwares.

#### **1.4.2.2. OpenUp**

É baseado no Processo Unificado. Ele aplica uma abordagem iterativa e incremental dentro de um ciclo de vida estruturado e abraça uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento de software. (TIESPECIALISTAS, 2015)

Assim como todos os demais métodos ágeis, o OpenUp também apresenta melhores práticas que visam criar o ambiente ideal para o desenvolvimento de projectos. Dentre elas destacam-se:



- Engloba uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento do software;
- É um processo independente de ferramenta e de pouca cerimónia que pode ser usado para atender uma ampla variedade de tipos de projectos;
- Não provê orientação em alguns tópicos que o projecto possa tratar, como equipes grandes, situações contratuais, aplicações de segurança ou missão crítica, orientação de tecnologia específica, etc.;
- Foco antecipado na arquitectura, para minimizar os riscos e organizar o desenvolvimento;
- Envolver, para continuamente obter feedback e melhorar;

O seu ciclo de vida consiste de quatro fases: Início, Elaboração, Construção e Transição. (TIESPECIALISTAS, 2015).

- **Concepção (Inception)** - onde os *stakeholders* e os membros da equipe colaboram para determinar o escopo e os objetivos do projeto, e determinar se o projeto deve ou não continuar.
- **Elaboração (Elaboration)** - quando riscos arquiteturais significantes são tratados.
- **Construção (Construction)** - foca no detalhamento dos requisitos, no desenho, na implementação e nos testes da maior parte do software.
- **Transição (Transition)** - focada na transição do software para o ambiente do cliente e na obtenção da concordância dos *stakeholders* de que o desenvolvimento do produto está completo.

O ciclo de vida de projeto fornece aos *stakeholders* e à equipe de projeto, visibilidade e pontos de decisão durante o projeto. Isto permite uma efetiva supervisão para tomar decisões de “prosseguir ou parar” em momentos apropriados. Um plano de projeto define o ciclo de vida, e o resultado final é uma aplicação passível de ser utilizada.

### Princípios do OpenUP

O OpenUP está baseado em **quatro princípios** fundamentais mutuamente suportados:

- **Equilibrar as prioridades concorrentes para maximizar o benefício aos *Stakeholders*:** promover práticas que permitam aos participantes do projeto e aos

stakeholders desenvolver uma solução que maximize os benefícios para o stakeholder, e que seja compatível com as restrições impostas ao projeto.

- **Colaborar para alinhar os interesses e compartilhar o entendimento:** promover práticas que estimulem um ambiente de equipe saudável, permitam a colaboração e desenvolvam uma compreensão compartilhada do projeto.
- **Focar na arquitetura, o mais cedo possível, para reduzir o risco e organizar o desenvolvimento:** promover práticas que permitam à equipe focar na arquitetura para reduzir o risco e organizar o desenvolvimento.
- **Evoluir para continuamente obter feedback e promover melhorias:** promover práticas que permitam à equipe obter feedback dos stakeholders, o mais cedo possível e de forma contínua, e demonstrar valor incremental para eles.

### **Papéis do OpenUP**

As pessoas nos papéis executam as tarefas que usam e produzem os artefatos, ninguém constrói um bom software sozinho, mas uma equipe trabalhando junto pode fazer coisas extraordinárias.

**Arquiteto:** responsável por definir a arquitetura de software, incluindo a tomada das principais decisões técnicas que orientam todo o desenho e a implementação do projeto.

**Gerente de Projeto:** conduz o planejamento do projeto, coordena as interações com os stakeholders e mantém a equipe de projeto focada em alcançar os objetivos do projeto.

**Analista:** representa os interesses do cliente e do usuário final recolhendo informações dos stakeholders para entender o problema a ser resolvido, capturando os requisitos e definindo suas prioridades.

**Testador:** responsável pelas principais atividades do esforço de teste. Estas atividades incluem identificar, definir, implementar e conduzir os testes necessários, bem como registrar e analisar os resultados dos testes.

**Qualquer papel:** Qualquer um em uma equipe pode atuar neste papel executando diversas tarefas.

**Desenvolvedor:** responsável por desenvolver uma parte do sistema, incluindo a construção de seu desenho de forma que ele atenda a arquitetura e possivelmente a prototipagem da interface de usuário, e então implementar, executar o teste de unidade e integrar os componentes que são parte da solução.

**Stakeholder:** representa grupos de interessados cujas necessidades devem ser satisfeitas pelo projeto. É um papel que pode ser executado por qualquer um que seja (ou potencialmente possa ser) afetado pelo resultado do projeto.

#### **1.4.2.3. Scrum**

O Scrum é o método ágil mais usado nos dias de hoje, principalmente porque pode ser integrado a outros métodos ágeis com facilidade, aplicando-se não só ao desenvolvimento de softwares como a qualquer ambiente de trabalho; ela mantém o foco na gestão do projecto. (BUILDER, 2017)

Diferentemente dos demais métodos ágeis, o Scrum tem papéis e práticas muito bem definidas e essenciais para o sucesso do projecto. Dentre elas destacam-se:

- Tem como base o planeamento iterativo e incremental;
- Reitera desde o início do projecto a lista de funcionalidades a serem desenvolvidas;
- Indivíduos e interação mais do que processos e ferramentas;
- Software em funcionamento mais do que documentação;
- Colaboração com o cliente mais do que contratos e negociações;
- Respostas a mudanças mais do que planeamento. (BUILDER, 2017).

#### **Metodologia Utilizada**

Entre as metodologias de desenvolvimento de software descritas anteriormente decidiu-se utilizar OpenUp, por ser adaptável às necessidades da Direção do INSTIC, o que facilita que não se adicione um trabalho excessivo pela quantidade de papéis ou documentação que se gera com o uso das metodologias pesadas. Também é uma metodologia de desenvolvimento de software desenhada para pequenas equipas organizadas. OpenUp permite detectar erros através de um ciclo iterativo e como metodologia ágil é idónea para realizar o sistema de gestão em questão, já que se destaca por ser um processo de desenvolvimento de software simplificado, baseada nas melhores práticas de RUP, as quais já provaram sua efetividade.

OpenUp permite detetar erros temporários através de um ciclo iterativo e como metodologia ágil é idónea para realizar o sistema de gestão em questão, já que se destaca por ser um processo de desenvolvimento de software simplificado. Apesar de ser uma metodologia ágil tem princípios das metodologias pesadas como é que brinda uma

documentação detalhada da informação, e muito apesar de que se requer a obtenção do produto de forma imediata.

### 1.5. Ferramentas e tecnologias nas quais se apoia para a solução.

A integração ou interligação de ferramentas de desenvolvimento de tecnologias, surgiu como uma alternativa que permite desembrulhar aplicações. Eles também permitem resolver problemas actuais de uma organização, facilitando, assim, a interação do usuário com os sectores envolvidos no problema .

Para o desenvolvimento da investigação e implementação da solução, foram usadas ferramentas de modelagem UML, editores de códigos, frameworks de desenvolvimento e sistema de gerenciamento de banco de dados. Dentre os quais a posterior se fará a escolha das ferramentas começando pelas ferramentas de modelagem.

#### **1.5.1. Linguagem de programação web**

Linguagem De Programação Web são linguagens de programação específicas para o desenvolvimento de sítio eletrónico, portais e aplicações web em geral (SCRIPTCASE, 2021) .

##### **1.5.1.1. PHP 7.4**

PHP (acrônimo recursivo para PHP: Hypertext Preprocessor) é uma linguagem de script de uso geral de código aberto amplamente utilizada que é especialmente adequada para desenvolvimento web e pode ser incorporada em HTML (The PHP Group, 2019).

O PHP é uma linguagem de script do lado do servidor, especialmente adequada para a criação de páginas dinâmicas da web. Esta linguagem de programação oferece aos desenvolvedores da web uma grande variedade de instrumentos. O PHP, que se tornou a base para muitas aplicações web, permite fácil inserção em código HTML e conexão com bancos de dados MySQL e PgSQL (NTCHOSTING, 2019).

Vantagens do PHP, segundo o (1STWEBDESIGNER, 2019):

- Software livre liberado sob a licença PHP.
- Fácil de aprender (curva de aprendizado).
- Grande comunidade de usuários e desenvolvedores.
- Fornece suporte extensivo ao banco de dados.
- Oferece grande número de extensões disponíveis e códigos-fonte.

- Permite a execução de código em ambientes restritos.
- Oferece gerenciamento de sessão nativa e API de extensão.
- Uma ótima alternativa para concorrentes como o ASP da Microsoft (Active Server Pages).
- Pode ser implantado na maioria dos servidores da web.
- Funciona em quase todos os sistemas operacionais e plataformas.

Além da linguagem de programação PHP 7.1 existem outras linguagens de programação que dariam para programar o software, como os casos das linguagens ASP e JSP. A baixo vai a comparação entre uma das duas com a linguagem PHP e o motivo da escolha da escolha da mesma.

#### **1.5.1.2. PHP 7.4 vs JSP (Java Server Pages) 8.0**

Java Server Page é uma tecnologia usada para criação de paginas web geradas dinamicamente baseadas em XHTML e HTML, controla o conteúdo ou a aparência de paginas web através do uso de servlets, é ideal para projectos de maior porte, e mais robusto e flexível. . Esta tecnologia permite ao desenvolvedor produzir aplicações que acessem o banco de dados, manipulem arquivos no formato texto, capturem informações a partir de formulários e captam informações sobre o visitante e sobre o servidor, assim como essa existem muitas outras linguagens que serviriam para programar o sistema de informação proposto. Assim sendo para a programação do software proposto escolheu - se a linguagem PHP 7.4, por ser mais sensível a plataforma Web; a maior parte dos aplicativos Web existentes são feitas nessa linguagem, possui licença gratuita, fácil de aprendizagem e é mais popular, fornece maior sensibilidade de interagir com diferentes bancos de dados e, é ideal para implementação de pequenos e médios projectos.

#### **1.5.2. Framework de desenvolvimento de software**

Um framework é um facilitador no desenvolvimento de diversas aplicações e, sem dúvida, sua utilização poupa tempo e custos para quem o utiliza, pois de forma mais básica, é um conjunto de bibliotecas utilizadas para criar uma base onde as aplicações são construídas, um otimizador de recursos. Tem como principal objetivo resolver problemas recorrentes com

uma abordagem mais genérica. Ele permite ao desenvolvedor focar nos “problemas” da aplicação, não na arquitetura e configurações (ANDRADE, 2019).

#### **1.5.2.1. *BootsTrap 4.1.3***

O Bootstrap é uma framework de front-end livre e de código aberto para projetos de sites, sistemas de gestão e aplicativos móveis. Ele contém modelos de design baseados em HTML e CSS para tipografias, formas, botões, navegação e outros componentes de interface, bem como extensões de JavaScript opcionais. Ao contrário de muitos frameworks de código aberto disponíveis, ele é exclusivo para o desenvolvimento de front-end.

Trata-se apenas de um dos melhores e mais utilizados frameworks de front-end em HTML, CSS e JS. Sua popularidade é grande entre os desenvolvedores por oferecer uma série de componentes e recursos prontos, acelerando significativamente o desenvolvimento de softwares.

Cada componente e plugin está completamente documentado com exemplos e blocos de código para facilitar o uso e personalização do sistema. Essas vantagens ajudam a economizar dias e horas de trabalho com a codificação de uma UI (User Interface). O mais legal é que ele permite que desenvolva a estrutura primeiro para aplicar as fontes, cores e estilos depois.

Embora ele ofereça uma estrutura que seja fácil para começar um projeto, não se deixe enganar por sua simplicidade à primeira vista. Antes de iniciar o projeto, o ideal é que se dedique um tempo para estudar a sua estrutura, reduzindo as peças-chave de sua infraestrutura e componentes, para evitar problemas durante e após o desenvolvimento.

Vale ressaltar ainda que o Bootstrap oferece uma grande quantidade de recursos, particularmente úteis aos novos usuários, como artigos, tutoriais, plugins e extensões de terceiros, modelos prontos, construtores de temas e assim por diante.

#### **1.5.2.2. *Laravel 8***

Laravel é um framework PHP utilizado para o desenvolvimento web. Ele utiliza o padrão MVC (Model View Controller), que permite dividir o desenvolvimento em 3 camadas conceituais: Modelo, Visão e Controlador. O Laravel tem como principal característica ajudar a desenvolver aplicações seguras e performativas de forma rápida, com o código limpo e simples (ADRIEL, 2015). O framework em questão permite a criação de templates,

chamada de Blade. O Blade traz uma gama de ferramentas para auxiliar na criação de interfaces gráficas ricas e funcionais, com o conceito de evitar a duplicação de código (ADRIEL, 2015).

Além do Laravel existem outros frameworks que facilitam o desenvolvimento de software, como o Symfony. A baixo vai a comparação entre as duas e o motivo da escolha da mesma.

#### **1.5.2.3. *Laravel 8 vs Symfony 5.4***

Symfony é um frameworks livre projectado para o desenvolvimento de aplicações web, com a linguagem php e o modelo vista controlador (MVC), oferece um conjunto de ferramentas e classes destinadas a tornar a tarefa mais fácil e rápido. (MELGOZA, 2016)

Fica provado que tanto Laravel como Symfony serviriam para facilitar a criação do software proposto de forma eficiente, mas para este projecto preferiu-se usar a tecnologia Laravel para o desenvolvimento do mesmo.

#### **1.5.3. *Editor de texto de código***

Editores de Texto de Código são ferramentas de edição de código que auxiliam o desenvolvedor ou o programador com muitas funções necessárias que são: detetar erros de código, auto completo de código, numeração de linhas, iluminação de variáveis de programação, possibilidade de abrir vários documentos em abas e outros.

##### **1.5.3.1. *Visual Studio Code 1.47.1***

Em 2015 foi lançado pela Microsoft um editor de código destinado ao desenvolvimento de aplicações web chamado de Visual Studio Code, ou simplesmente VSCode. Anunciada durante o Build, evento voltado a desenvolvedores que ocorre nos Estados Unidos anualmente, trata-se de uma ferramenta leve e multiplataforma que está disponível tanto para Windows, quanto para Mac OS e Linux e atende a uma gama enorme de projetos, não apenas ASP.NET, como também Node.js. Adicionalmente, o editor possui suporte à sintaxe de diversas linguagens como Python, Ruby, C++. (DEVMEDIA, 2016)

Assim como nossos outros editores, o VSCode tem um bom ecossistema de plugins (extensões). O gerenciamento de extensões é embutido e já existem vários milhares disponíveis. Em sua essência, o Visual Studio Code possui um editor de código fonte super rápido, perfeito para o uso no dia-a-dia. Com suporte para centenas de linguagens, o VS Code ajuda a ser instantaneamente produtivo com realce de sintaxe, correspondência de

colchetes, recuo automático, seleção de caixa e muito mais. Atalhos de teclado intuitivos, fácil personalização e mapeamentos de atalhos de teclado contribuídos pela comunidade permitem que se navegue pelo código com facilidade.

#### **1.5.4. Ferramentas de modelagem UML**

UML (Unified Modeling Language) é uma linguagem de notação (um jeito de escrever, ilustrar, comunicar) para uso em projetos de sistemas.

Esta linguagem é expressa através de diagramas. Cada diagrama é composto por elementos (formas gráficas usadas para os desenhos) que possuem relação entre si. (VENTURA, 2019)

##### **1.5.4.1. Visual Paradigma 13.1**

Visual Paradigma é um modelador UML (Unified Modeling Language) para as equipes de desenvolvimento de software, criação de diagramas UML, produzir especificação do software através de HTML, PDF e relatórios em Word. Visual Paradigma suporta várias linguagens de modelagem mais populares, incluindo UML, ERD e SysML, ela suporta modelagem simultânea através da colaboração da equipe com vários tipos de sistema de controle de versão concorrente. Ela faz praticamente tudo que as outras ferramentas UML caro fazem, mas em uma fracção minúscula do custo. (L3SOFTWARE, 2017)

Dentre as suas características destacam-se: Está em conformidade com as políticas de software livre;

- É multiplataforma;
- Promove um pacote de ajuda para o desenvolvimento de software, desde o planeamento através de análise e projecto de geração de código para ambientes de desenvolvimento integrados, como Netbeans, Eclipse, Oracle JDeveloper, e JBuilder;
- Possui uma interface muito intuitiva e fácil de aprender, isto é, para os desenvolvedores;
- Permite a geração automática de diagramas a partir de descrições de casos de uso;
- Permite a descrição dos casos de uso que dão uma variedade de modelos padrão que permitem a personalização;



- Combina a funcionalidade de todos os problemas em uma plataforma de modelagem visual de comprimento;
- Possui ferramentas de auxílio ao mapeamento do modelo de classes para o modelo de dados relacional;
- Permite gerar tabela do banco de dados a partir dos diagramas de classe ou do modelo físico de dados. (DEVMEDIA, 2013)

#### **1.5.5. Sistema gestor de base de dados MySQL**

MySQL é um sistema de gerenciamento de banco de dados (SGBD), gratuito e livre. Criado inicialmente em 1995, foi sofrendo evoluções com o tempo, e atualmente é a plataforma mais utilizada no mundo. Trata-se de um serviço estável, seguro e confiável. Uma ferramenta bastante poderosa. O MySQL foi criado por David Axmark e Michael Widenius, dois engenheiros da Suécia (Bruno Feitosa, 2019).

As Vantagens do Mysql, segundo (DEVMEDIA, 2018):

- O MySQL é gratuito.
- Possui código fonte aberto.
- Facilidade de programação e aprendizado.
- Pode ser totalmente modificado.
- Possui funções importantes que auxiliam durante o processo de desenvolvimento.
- Pode ser utilizado em qualquer tipo de aplicação desde as mais simples as mais robustas.
- O mercado disponibiliza diversos profissionais com expertise no uso do banco de dados.
- Barateia significativamente o valor final do projeto.
- É multiplataforma.
- Permite que sejam implementadas regras de segurança no servidor.

### **1.6. Conclusões parciais do capítulo**

O estudo dos principais conceitos que constituem a base teórica do tema permitiu ter um melhor conhecimento e entendimento dos conceitos relacionados com objeto de estudo.

O estudo sobre as metodologias, tecnologias e ferramentas de desenvolvimento de software, permitiu a seleção da metodologia, tecnologias e ferramentas para o desenvolvimento do sistema. Desta maneira ficou definido o uso do OpenUp como metodologia de desenvolvimento de software, Visual Paradigma para UML na versão 13.1. Sistema gestor de banco de dados MySQL, como linguagem de programação PHP e Laravel 8 como frameworks para backend e o Bootstrap para o frontend.

## 2. Capítulo 2: Características do sistema.

### 2.1. Introdução

Neste capítulo faz-se a análise mais detalhada dos processos de negócio e do sistema proposto tendo em conta os requisitos funcionais e não funcionais, desta forma faz-se o modelo conceitual e a descrição do domínio, capturam-se os requisitos funcionais e não funcionais do sistema, elaboração de diagrama de casos de uso do sistema proposto com as suas descrições, isto em base a especificação de requisitos do sistema já capturado.

### 2.2. Objecto de automação

O sistema consiste em criar plano de trabalho para a direção do Instituto de Tecnologias de Informação e comunicação. Existem três níveis de acesso: Administrador e Funcionário.

O nível administrador é o mais alto, tem como função gerir os usuários do sistema, cadastrando-os e definir seus níveis de acesso ao sistema. O nível Funcionário tem a função solicitar e criar reunião, criar e delegar tarefa, e criar sua propria agenda de trabalho.

### 2.3. Modelagem do Negócio:

Segundo (Silva, Alberto Manuel Rodrigues da e Videira, Carlos Alberto Escaleira, 2001), a Modelagem “ é a arte e ciência de criar modelos de uma determinada realidade”. Desta forma pede-se se entender a modelagem de negócio como a criação de modelos do negócio, resultante das análises e reflexões sobre a natureza do negócio e a forma como ele é executado, ou seja, sobre as características do negócio e as rotinas organizacionais (Rodrigues, 2015).

O resultado da modelagem de negócio são os modelos de negócio, esses modelos reflectem a representação de um conjunto de actividades, processos, quem os realiza e os serviços que a organização oferece a seus clientes.

### 2.3.1. Descrição do negócio

Na Direcção do Instituto de Tecnologias de Informação e Comunicação (INSTIC) existem dois tipos de reunião: reunião ordinária e reunião extraordinária. As ordinárias são aquelas que ocorrem em período regular (em cada 15 dias, mensal ou anual). As extraordinárias ocorrem de forma irregular, normalmente quando se precisa analisar um documento que chegou à direcção. Logo após terminar a reunião é feita uma acta que é aprovada pelos participantes da mesma.

### 2.3.2. Modelo conceptual ou domínio

O modelo de domínio é uma representação visual de classes conceituais ou objectos do mundo real em um domínio de interesse. Usando a notação UML, o modelo de domínio é ilustrado com um conjunto de diagramas de classes nos quais não são definidas operações. Ele deve mostrar:

- Objetos do domínio ou classes conceituais;
- Associações entre classes conceituais;
- Atributos de classes conceituais.

A figura 2-1 mostra o modelo de domínio do sistema.

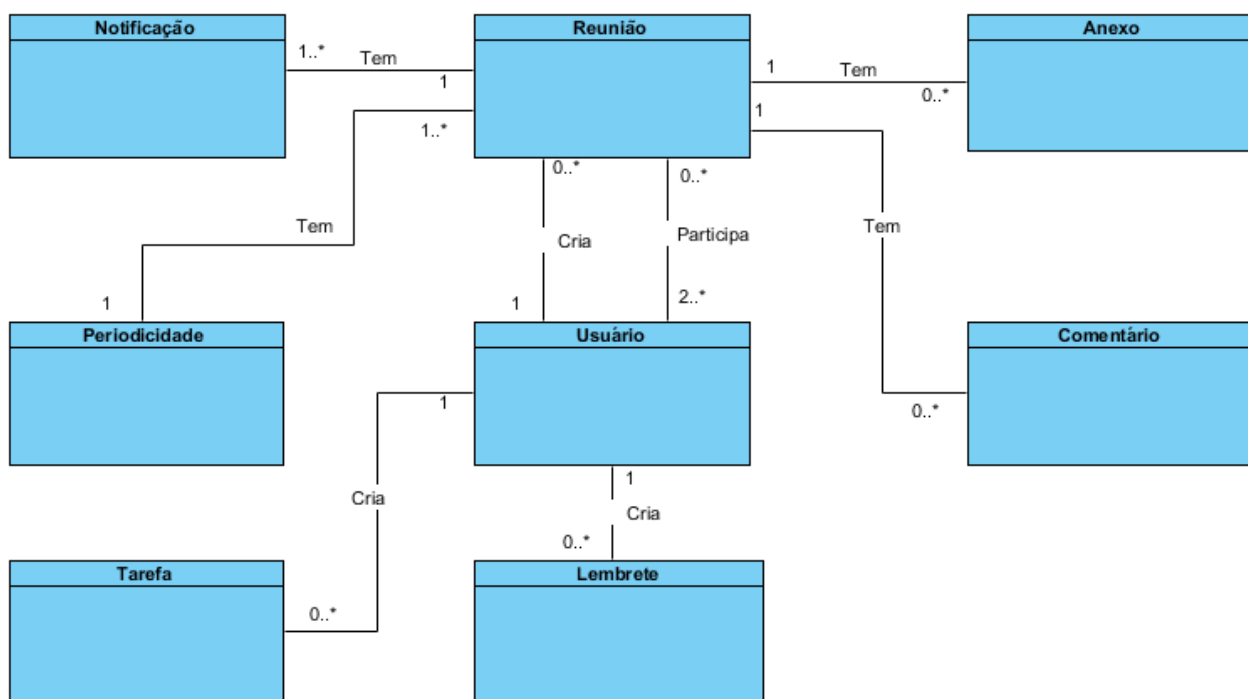


Figura 2-1: Modelo de domínio. Fonte: Elaboração própria

### **2.3.3. Descrição dos trabalhadores e actores do negócio**

Uma vez identificados os processos do negócio é possível determinar os atores e trabalhadores que participam em ditos processos.

#### **2.3.3.1. Actor do negócio**

Actor é algo ou alguém que interage com o sistema, mas sobre o qual se tem o controlo. Ele está fora da influência do sistema. Os atores têm um papel extremo e são os quem iniciam (e quem responde) aos casos de uso.

Tipicamente, um ator representa um papel que um ser humano, um outro processo, um outro sistema, ou até um dispositivo de hardware, desempenha ao interagir com o sistema.

Cada actor corresponde a um papel específico: uma mesma pessoa que desempenha diferentes papéis nas interações com o sistema é representada por diferentes actores;

Na Tabela 2, mostra as descrições dos actores de negócio, ou, seja as entidades que fazem parte do negócio.

Tabela 2: Actores de negócio

| Actor       | Descrição do Actor   |
|-------------|--|
| Funcionario | Individuo que pode solicitar uma reunião ou documento com o coordenador do seu departamento. |

#### **2.3.3.2. Trabalhador de negócio**

Trabalhador de negócio é uma classe que representa uma abstração de uma pessoa que actua no sistema. Os trabalhadores de negócios interagem entre si e manipulam entidades de negócios enquanto participam de realizações de casos de uso de negócios. A tabela 3 mostra a descrição dos trabalhadores e atores do negócio.

Tabela 3: Descrição dos trabalhadores e atores do negócio

| Trabalhador | Nome do trabalhador | Descrição trabalhador |
|-------------|---------------------|-----------------------|
|-------------|---------------------|-----------------------|

|               |                  |   |
|---------------|------------------|---|
| Trabalhador 1 | Director Geral   | Individuo que pode convocar uma reunião com todos os funcionarios   |
| Trabalhador 2 | Director Adjunto | Individuo que pode solicitar uma reunião com o Coordenador geral e todos os outros funcionarios do seu departamento |
| Trabalhador 3 | Funcionarios     | Individuo que pode solicitar uma reunião com o coordenador do seu departamento                                      |

#### 2.3.4. Diagrama de Casos de Uso do Negócio

Abaixo na Figura 2-2, se mostra o diagrama de caso de uso do negócio

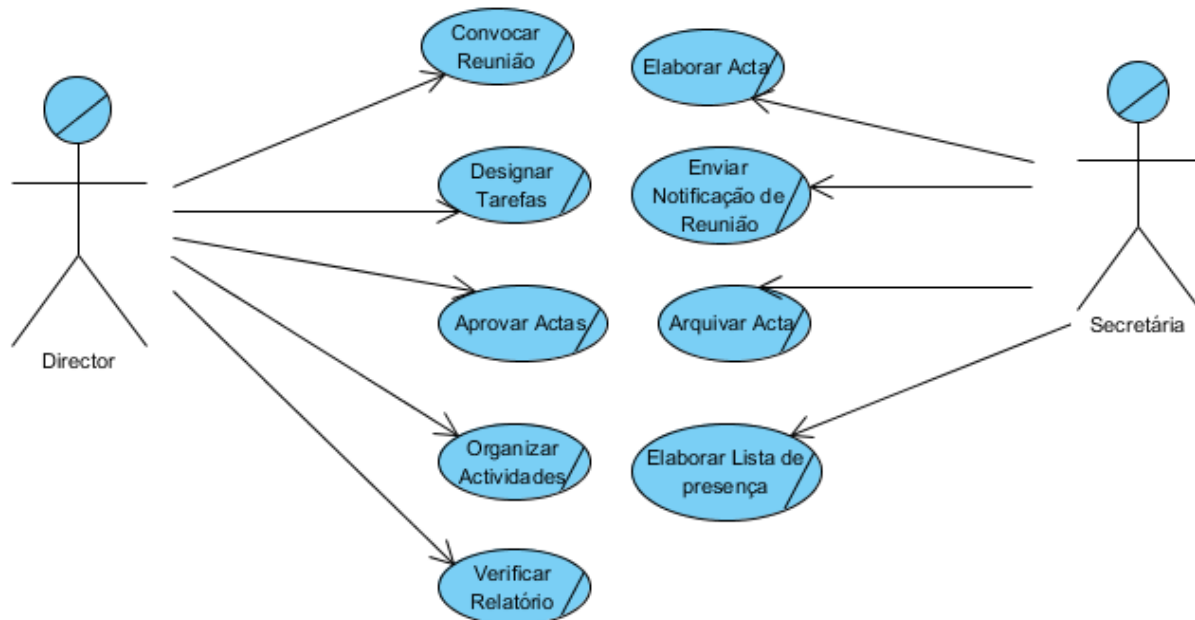


Figura 2-2: diagrama de caso de uso do negócio. Fonte: Elaboração própria

#### 2.3.5. Descrições de Casos de Uso do Negócio.

A Tabela 4 mostra a descrição do caso de uso para o negócio “Enviar notificação de reunião”. No caso de uso, o actor Director e o trabalhador Secretária interagem, para poder

notificar os participantes da reunião, o que permite que todos os participantes da reunião recebam uma notificação com as informações necessária para a realização da mesma.

Tabela 4: Descrição do caso de uso para o negócio “Enviar notificação de reunião”

|                                   |  |
|-----------------------------------|--|
| Caso de Uso:                      | Enviar notiicação de reunião   |
| Atores:                           | Director   |
| Trabalhadores:                    | Secretária   |
| Resumo:                           | O caso de uso começa quando o Director chega a secretária e convoca uma reunião. Esta notifica todos os participantes a se fazerem presentes na mesma. |
| Seção                             |  |
| Fluxo normal de eventos           |  |
| Ator                              | Trabalhador  |
| 1. Convocar reunião               | 2. Solicita Documentos a se tratar na mesma  |
| 3. Entrega documentos necessarios | 4. Recebe documentos necessários e verifica-os, copia-os e notifica todos os participantes da reunião.   |

A Tabela 5 mostra a descrição do caso de uso para o negócio “Elaborar Acta”. No caso de uso, o actor Director e o trabalhador Secretária interagem, para realizar uma reunião, o que permite no final na mesma realizar uma acta sobre o que se tratou na reunião.

Tabela 5: Descrição do caso de uso para o negócio “Elaborar Acta”

|                         |  |
|-------------------------|--|
| Caso de Uso:            | Elaborar Acta  |
| Actores:                | Director   |
| Trabalhadores:          | Secretária   |
| Resumo:                 | O caso de uso começa quando é marcada uma reunião, a secretária encarregada deve elaborar a acta após a reunião. |
| Seção                   |  |
| Fluxo normal de eventos |  |
| Actor                   | Trabalhador  |
| 1. Convocar reunião     | 2. Notificar todos os participantes da reunião   |
| 3. Inicia a reunião     | 4. Faz um relatorio do que se esta a tratar na reunião   |
| 5. Termina a Reunião    | 6. Elaborar a acta   |

## 2.4. Modelagem do Sistema:

A modelagem de sistemas é o processo de desenvolvimento de modelos abstractos de um sistema, de maneira que cada modelo apresenta uma visão ou perspectiva diferente do sistema. Ela ajuda o analista entender as funcionalidades do sistema e os modelos que são usados para a comunicação com os clientes.

Segundo (HALL, 2011), a modelagem de sistemas ajuda o analista a entender a funcionalidade do mesmo e os modelos são usados para comunicação com as empresas ou clientes. Os modelos geralmente são usados durante o processo de engenharia de requisitos para auxiliar a extração dos requisitos do sistema; durante o processo do projeto, são usados descrever o sistema para engenheiros que implementam e, após isso, são usados para documentar a estrutura e a operação do sistema.

Os requisitos de um sistema são as descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Esses requisitos refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido ou encontrar informações (SOMMERVILLE, 2011).

A seguir se mostram algumas tabelas com as descrições, entradas e saídas de dados dos requisitos funcionais do sistema.

### 2.4.1. Requisitos funcionais

Segundo (MACHADO, 2015) afirmou sobre os requisitos funcionais:

Os Requisitos funcionais são aqueles que descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, é aquele que descreve as funcionalidades, as quais se espera que o sistema forneça. Eles dependem do tipo de software que está sendo desenvolvido, do conhecimento passado pelos usuários sobre o negócio em si e do que fazer o software que se espera desenvolver.

Abaixo na Tabela 6, se mostra a descrição dos requisitos funcionais do sistema.

Tabela 6: Requisitos funcionais

| Categoria     | Requisitos funcionais                             | Complexidade | Prioridade |
|---------------|---|--------------|------------|
| Gerir usuário | RF1 - Cadastrar usuário<br>RF2 - Eliminar usuário |              |            |



|   |  |       |       |
|---|--|-------|-------|
|   | RF3 - Gerir nível de acesso  | Alta  | Alta  |
| Gerir actividade (reunião, tarefa e lembrete) | RF4 - Criar<br>RF5 - Editar<br>RF6 - Eliminar<br>RF7 - Listar<br>RF8 - Convocar reunião                                | Média | Alta  |
| Gerir acta de reunião                         | RF9 - Anexar acta após reunião   | Média | Média |
| Notificação                                   | RF10 - Notificar usuário marcado em uma reunião  | Alta  | Alta  |
| Relatório                                     | RF11 - Gerar relatório de actividade por usuário<br>RF12 - Gerar relatório de presença<br>RF13 - Gerar relatório total | Baixa | Baixa |

#### **2.4.2. Requisitos não-funcionais.**

Definem propriedades e restrições do sistema, como segurança, desempenho, espaço que aplicação ocupa no dispositivo de armazenamento, usabilidade, confiabilidade, disponibilidade, etc., são os requisitos relacionados ao uso e estabilidade da aplicação.

Segundo (MACHADO, 2015) sobre requisitos não funcionais:

- Os requisitos não funcionais não estão ligados diretamente com as funções fornecidas pelo sistema.
- Em geral se preocupam com padrões de qualidade como confiabilidade, desempenho, robustez, segurança, usabilidade, portabilidade, legibilidade, qualidade, manutenibilidade, entre outros.
- São muito importantes, pois definem se o sistema será eficiente para a tarefa que se propõe a fazer. Um sistema eficiente certamente não será usado.
- Para que o sistema execute foram identificados os seguintes requisitos não funcionais:

Abaixo na Tabela 7, se mostra a descrição dos requisitos não funcionais do sistema.

Tabela 7: Requisitos não funcionais

| Categoria      | Requisitos não funcionais   |
|----------------|---|
| Usabilidade    | <ul style="list-style-type: none"><li>• <b>RNF01</b>- Deve ter uma interface do usuário intuitiva e fácil de navegar.</li></ul>   |
| Confiabilidade | <ul style="list-style-type: none"><li>• <b>RNF02</b>- Os relatórios gerados pelo sistema deverão ser reais e precisos;</li><li>• <b>RNF03</b>- O sistema deve permitir apenas acesso e a modificação da informação por usuários autorizados.</li></ul>  |
| Fiabilidade    | <ul style="list-style-type: none"><li>• <b>RNF04</b>- O sistema deve informar ao usuário por meio mensagem indicando-lhe que ocorreu uma falha na operação que se realize.</li></ul>  |
| Software       | <ul style="list-style-type: none"><li>• <b>RNF05</b>- Cliente de Aplicação: Para aceder a aplicação requer usar um Navegador.</li><li>• <b>RNF06</b>- Servidor de Banco de Dados: para permitir a devida persistência dos dados o sistema deverá requerer um sistema gestor de banco de dados MySQL na versão 5.5.5-10.4.18 – MariaDB</li></ul>   |
| Segurança      | <ul style="list-style-type: none"><li>• <b>RNF7</b>- O sistema deve permitir ao usuário realizar suas actividades em função do nível de acesso que possui no sistema.</li><li>• <b>RNF8</b>- Se utilizará o protocolo HTTPS para a comunicação entre o cliente e o servidor no processo de envio de dados submetidos pelo usuário na autenticação e nas tarefas administrativas e de gestão de conteúdos.</li></ul> |

#### **2.4.3. Diagrama de Casos de Uso do Sistema.**

Abaixo na Figura 2-3, se mostra o diagrama de caso de uso do sistema

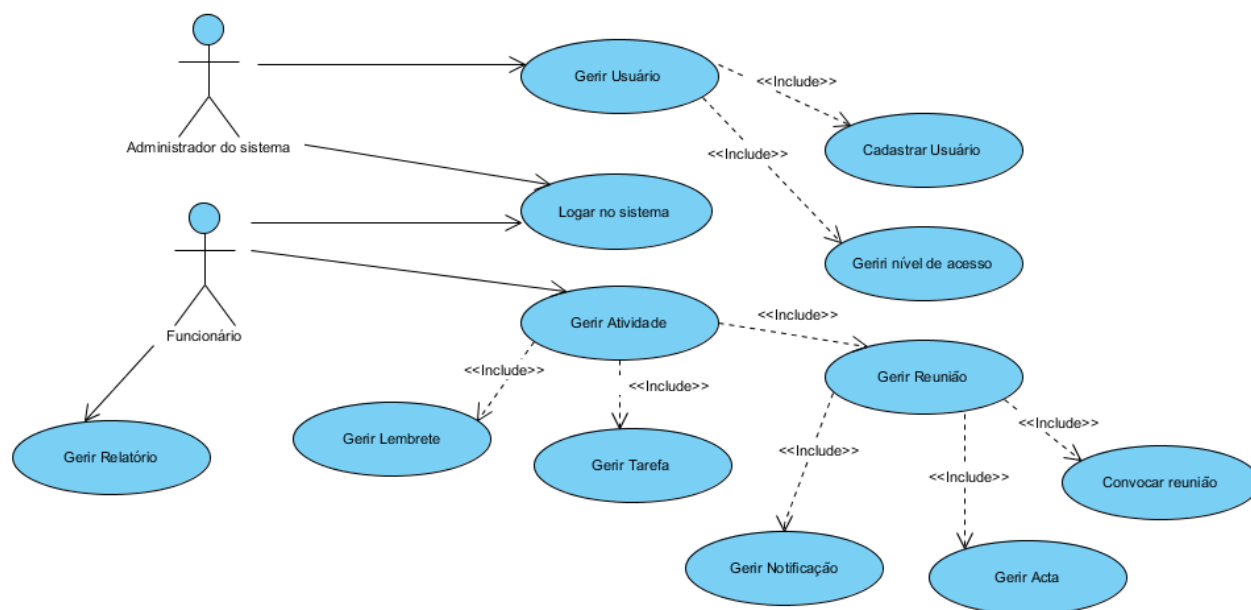


Figura 2-3: Diagrama de Casos de Uso do Sistema. Fonte: Elaboração própria

A Tabela 8. apresenta a descrição dos actores do sistema.

Tabela 8: Descrição dos actores do sistema.Fonte: Elaboração própria

| Actores                  | Descrição trabalhador   |
|--------------------------|---|
| Usuário<br>(Funcionário) | Individuo Responsável em gerir Actividades e gerar relatório  |
| Administrador            | Individuo responsável em gerir os usuarios e niveis de Acesso |

#### 2.4.4. Descrição de Casos de Uso do Sistema. Protótipos de interface.

A Tabela 9 mostra a descrição do caso de uso do sistema “Criar reunião”. No caso de uso, o ator Funcionário uma reunião.

Tabela 9: Descrição do caso de uso “Criar reunião”

|                     |               |
|---------------------|---------------|
| <b>Caso de Uso:</b> | Criar reunião |
| <b>Atores:</b>      | Funcionário   |

|   |   |  |
|---|---|--|
| Resumo:   | O caso de uso começa quando o funcionário pretende criar uma reunião, então ele acessa o sistema e escolhe a função que permite criar uma reunião, preenche os campos e salva os dados. |  |
| Pré-Condição  | Preencher os campos.  |  |
| Pós-Condição  | A reunião deve ser criada   |  |
| Referência RF   | RF4 - Criar reunião   |  |
| Prioridade  | Alta  |  |
| Fluxo normal de eventos   |   |  |
| Acção do ator   |   | Resposta do sistema  |
| 1. Na tela principal, clica em “Nova actividade”.                   |   | 2. Apresenta uma lista de actividades.   |
| 3. Clica em “Reunião”.  |   | 4. Apresenta a página com o formulário para criar uma reunião.                     |
| 5. Preenche os campos com os dados necessários e clica em “Salvar”. |   | 6. Guarda as informações e emite uma mensagem informando que os dados foram salvo. |
| Fluxo alterno de eventos  |   |  |
| Campo vazio   |   |  |
| 1. Clica em “Salvar”.   |   | 2. Emite uma mensagem informando que os campos vazios devem ser preenchidos.       |
| 3. Repete o passo 5 do fluxo principal                              |   | 4. Repete o passo 6 do fluxo principal.  |
| Protótipo de Interface  |   |  |

SGPT-INSTIC

Bem vindo/a, Kapinge de Almeida

Departamento de ...

Nova actividade

Painel de controle

Actividades

Estatística

Minha conta

Criar Reunião

Título

Adicionar título

Data

dd/mm/aaaa

Hora

--:--

Periodicidade

Não se repete

Local

Adicionar local

Descrição

Adicionar uma descrição

Adicionar Anexo

Escolher ficheiro

Nenhum ficheiro selecionado

Salvar

A figura apresentada na tabela 9 refere-se ao protótipo de interface do caso de uso “Criar Reunião”, pode-se observar que esta permite a realização das ações descritas no fluxo de eventos.

A Tabela 10 mostra a descrição do caso de uso do sistema “Cadastrar Usuário”. No caso de uso, o ator Administrador realizar o cadastramento de usuário, o que permite com que os dados sejam salvos e por sua vez criar uma conta de usuário.

Tabela 10: Descrição do caso de uso “Cadastrar Usuário”

|  |   |
|--|---|
| Caso de Uso:   | Cadastrar Usuário   |
| Atores:  | Administrador   |
| Resumo:  | O caso de uso começa quando o Administrador pretende cadastrar um novo usuário no sistema, então o Administrador acessa o sistema e no menu principal selecciona em “Usuário” e escolhe a função que permite adicionar novo usuario, preenche os campos e salva o novo usuário. |
| Pré-Condição   | Preencher os campos.  |
| Pós-Condição   | O novo usuário deve estar cadastrado no sistema   |
| Referência RF  | RF1- Cadastrar usuário  |
| Prioridade   | Alta  |
| Fluxo normal de eventos  |   |
| Ação do ator   | Resposta do sistema   |
| 1. No menu principal, clica em “Usuário”.                          | 2. Mostra uma lista de acções referente ao usuário.   |
| 3. Clica em “Cadastrar”.   | 4. Apresenta o formulário de cadastro de usuário.   |
| 5. Preenche os campos com os dados do usuário e clica em “Salvar”. | 6. Recolhe e guarda os dados do usuário e redireciona novamente à página de cadastro para um novo cadastramento.  |
| Fluxo alterno de eventos   |   |
| Campo vazio  |   |
| 1. Clica em “Salvar”.  | 2. Emite uma mensagem informando que os campos vazios devem ser preenchidos.  |
| 3. Repete o passo 5 do fluxo principal                             | 4. Repete o passo 6 do fluxo principal.   |
| Protótipo de Interface   |   |

A figura apresentada na tabela 11 refere-se ao protótipo de interface do caso de uso “Cadastrar Usuário”, pode-se observar que esta permite a realização das ações descritas no fluxo de eventos.

A Tabela 11 mostra a descrição do caso de uso do sistema “Anexar acta após reunião”. No caso de uso, o ator Director anexa a acta após a realização da reunião, o que permite a aprovação da acta por todos os participantes da reunião.

Tabela 11: Descrição do caso de uso “Anexar acta após reunião”

|                     |   |
|---------------------|---|
| <b>Caso de Uso:</b> | Anexar acta após reunião  |
| <b>Atores:</b>      | Funcionário   |
| <b>Resumo:</b>      | O caso de uso começa quando uma reunião é realizada. No final da reunião, o usuário que convocou a mesma deve anexar a acta para que os participantes analisem e aprovem. Para tal o Funcionário acessa o sistema e no menu principal clica em “Reuniões”, seleciona a reunião desejada e escolhe a função que permite anexar a acta. |
| <b>Pré-Condição</b> | Carregar a acta   |

|   |   |  |
|---|---|--|
| Pós-Condição  | A acta deve estar anexada na tarefa                                 |  |
| Referência RF   | RF6-Anexar acta após reunião  |  |
| Prioridade  | Alta  |  |
| Fluxo normal de eventos   |   |  |
| Acção do ator   | Resposta do sistema   |  |
| 1. No menu principal, clica em “Reuniões”.  | 2. Mostra a lista com todas as reuniões.                            |  |
| 3. Clica na reunião desejada  | 4. Apresenta a página com as informações da reunião.                |  |
| 5. Clica em “Anexar acta”   | 6. Apresenta o explorador de arquivo do sistema.                    |  |
| 7. Seleciona o arquivo desejado e clica em “Abrir”  | 8. Faz o carregamento do arquivo                                    |  |
| 9. Clica em “Salvar”  | 10. Guarda o arquivo no sistema.                                    |  |
| Fluxo alterno de eventos  |   |  |
| Campo vazio   |   |  |
| 5. Clica em “Salvar”.   | 6. Emite uma mensagem informando que deve antes carregar o arquivo. |  |
| 7. Clica em “OK”.   | 8. Repete o passo 5 a 10 do fluxo principal.                        |  |
| Protótipo de Interface  |   |  |
| <div><div>Descrição</div><div><div>Adicione uma descrição ao documento</div></div><div>Carregar documento</div><div><div>Escolher ficheiro</div><div>Nenhum ficheiro selecionado</div></div><div>Salvar</div></div> |   |  |

A figura apresentada na tabela 11 refere-se ao protótipo de interface do caso de uso “Anexar acta após reunião”, pode-se observar que esta permite a realização das ações descritas no fluxo de eventos.

## 2.5. Conclusões parciais do capítulo

Uma vez analisado e apresentado os elementos para conhecer as características do sistema, pode-se concluir que:

A modelagem de negócio, permitiu conhecer mais sobre o funcionamento do negócio da Direção do INSTIC, isto é, a partir do diagrama de descrição do caso de negócio foi possível verificar de que jeito ocorre o fluxo de negócio, permitiu-se ter um maior

entendimento com relação ao Plano de Trabalho, para assim definir quais funcionalidade devem ser implementadas para dar solução a situação problemática apresentada.

Na modelagem do sistema foi possível identificar e definir os requisitos para atender as necessidades da Direção do INSTIC, isto é, requisitos funcionais, bem como definir requisitos não funcionais que permitiram criar o ambiente necessário para a aplicação dos requisitos funcionais. Com a representação e descrição dos casos de uso do sistema foi possível detetar que o sistema está atendendo as reais necessidades do INSTIC. O protótipo de interface do utilizador e as telas de interface desenhadas permitiram facilitar o uso do software.



### 3. Capítulo 3: Desenho, Implementação e Provas do Sistema.

#### 3.1. Introdução do capítulo

Neste capítulo são abordados assuntos relacionados com o desenho, desenvolvimento e provas do sistema informático desta feita se realizou a modelação detalhada com aplicação de padrões de desenho e a construção da estrutura do sistema informático obtendo como artefacto diagrama de classes, arquitectura do sistema, modelo físico do banco de dados, diagrama de implantação, e finalmente documentam-se as provas realizada sobre o sistema informático, feita para verificar se o sistema corresponde ao que foi especificado e detectar falhas no sistema informático.

#### 3.2. Diagrama de Classes

O diagrama de classes é uma das técnicas mais utilizadas no desenvolvimento orientado por objectos. Este diagrama é uma descrição formal da estrutura de objectos num sistema. Para cada objecto descreve a sua identidade, os seus relacionamentos com os outros objectos, os seus atributos e as suas operações.

Abaixo na Figura 3-1 se mostra o diagrama de classe

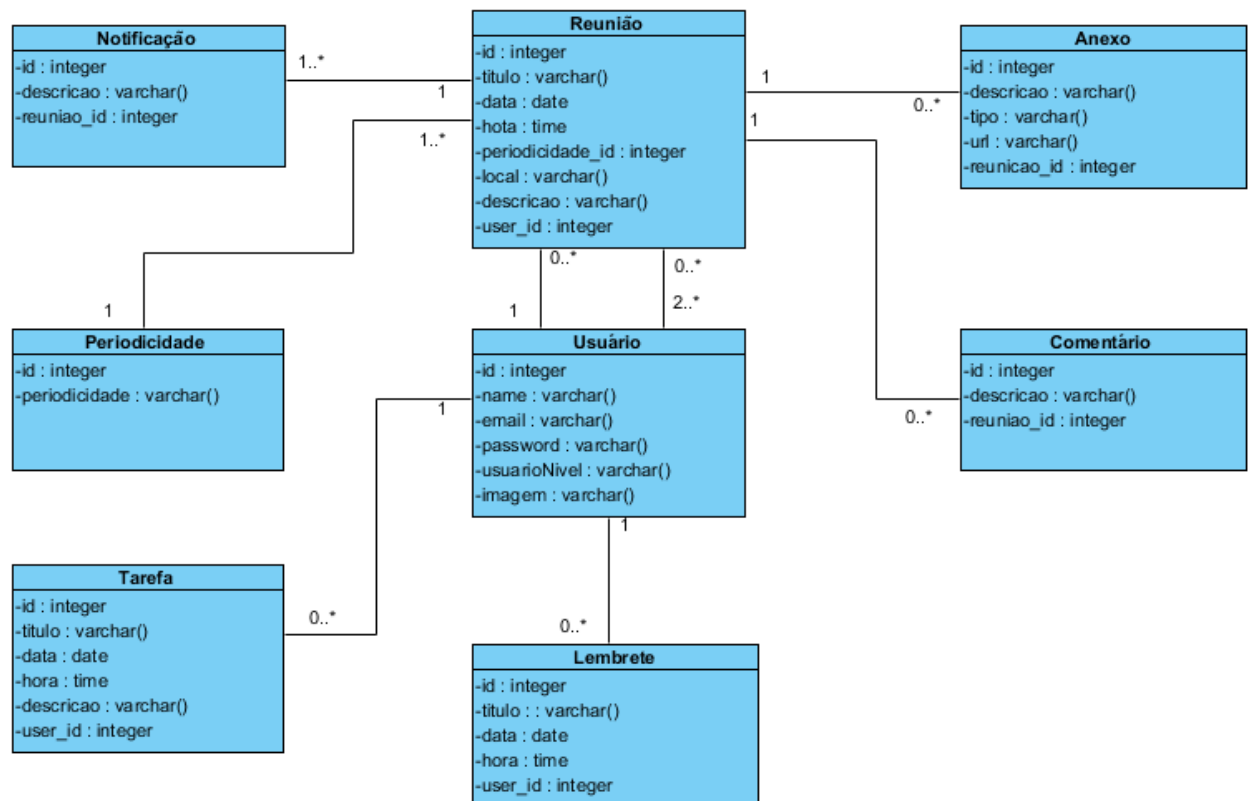


Figura 3-1: Diagrama de classe. Fonte: Elaboração própria

A Figura 3.2.1 mostra o diagrama de classe do sistema, descreve o modelo geral de informação do sistema e é composto pelos seguintes elementos: Classes de objectos, relação de associação e composição, multiplicidade. O diagrama é composta por seis classes de objecto (Usuário, Calendário, Tarefa, Departamento e Acta) com os seus respectivos atributos, com relações de associação e composição, ligado a multiplicidade variado de 0..\* por 0..\*, 0..\* por 1, 1 por 1..\* e 1 por 1.

### 3.2.1. Diagrama de Classe com estereótipo

Abaixo na Figura 3-2, se mostra o diagrama de classe com estereótipo

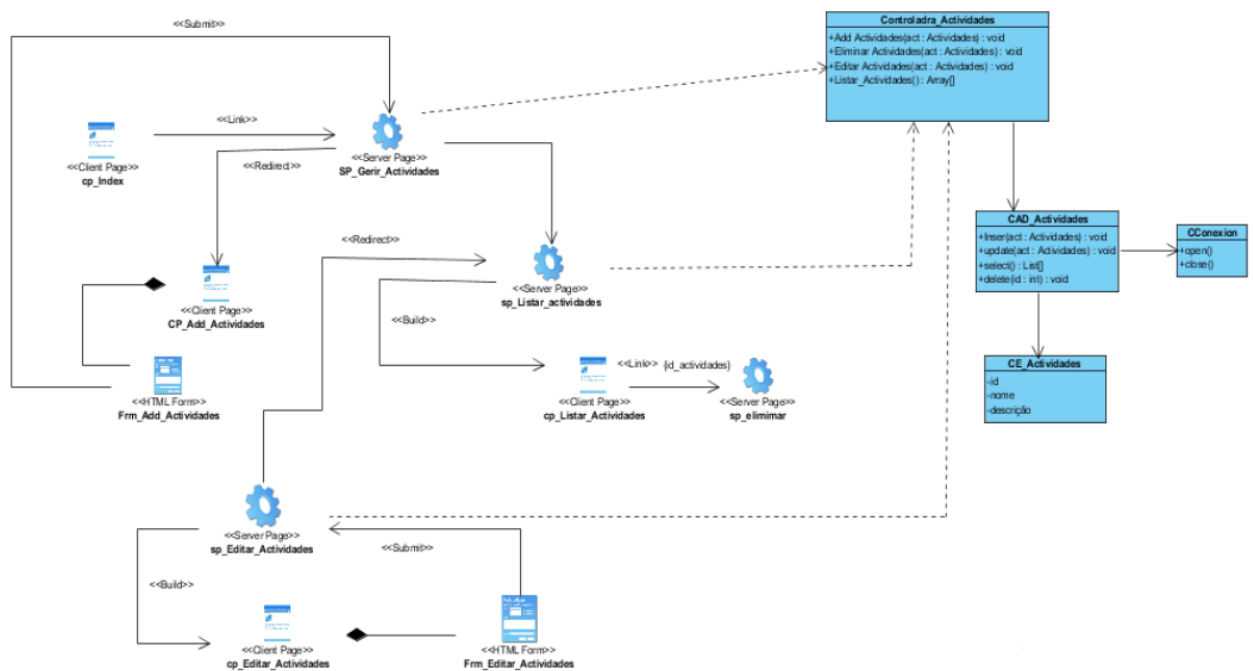


Figura 3-2: Diagrama de classe com estereótipo. Fonte: Elaboração própria

### 3.3. Estilo arquitetônico

A arquitetura de um sistema tem diversos elementos como: elementos utilitários, elementos de interação, elementos que fazem parte do domínio do problema, elementos de conexão, elementos de persistência, etc. Dessa forma, na arquitetura sempre definimos os seus elementos que precisarão ser utilizados no software e como eles se conectam. Uma arquitetura complexa exige mais tempo para desenvolvimento, porém, através de geração automática de aplicações isso pode se tornar mais produtivo. Por isso algumas equipes definem um framework para uma determinada aplicação, assim podemos utilizar muitas coisas pré-prontas que facilitam o desenvolvimento.

No entanto, alguns padrões arquiteturais já foram pensados para resolver problemas corriqueiros. Alguns projetos ou organizações combinam esses padrões arquiteturais, pois atendem melhor às suas necessidades ou deram certo para o seu tipo de aplicação. Por isso é sempre interessante entender as características básicas de cada um dos estilos e escolher ou combinar aqueles que atendem melhor às necessidades de um projeto específico, isso tudo, deve ser feito após uma análise do sistema a ser desenvolvido. Entre as arquiteturas existentes temos: Cliente-servidor, P2P - Peer to Peer, Dados compartilhados, Máquina virtual, Camadas, MVC e muitos outros.

Para essa proposta da solução seleciona-se o padrão arquitetônico MVC.

**Modelo Vista Controlador (MVC):** é nada mais que um padrão de arquitetura de software, separando sua aplicação em 3 camadas. A camada de interação do usuário (view), a camada de manipulação dos dados (model) e a camada de controle (controller).

**Modelo:** Sempre que você pensar em manipulação de dados, pense em modelo. Ele é responsável pela leitura e escrita de dados, e também de suas validações.

**Vista:** Simples, a camada de interação com o usuário. Ela apenas faz a exibição dos dados, sendo ela por meio de um HTML ou XML.

**Controlador:** O responsável por receber todas as requisições do usuário. Seus métodos chamados actions são responsáveis por uma página, controlando qual modelo usar e qual vista será mostrado ao usuário. (NUGROHO, 2017)

Abaixo se mostra na Figura 3-3, o diagrama MVC, representando de forma resumida a interação entre os componentes.

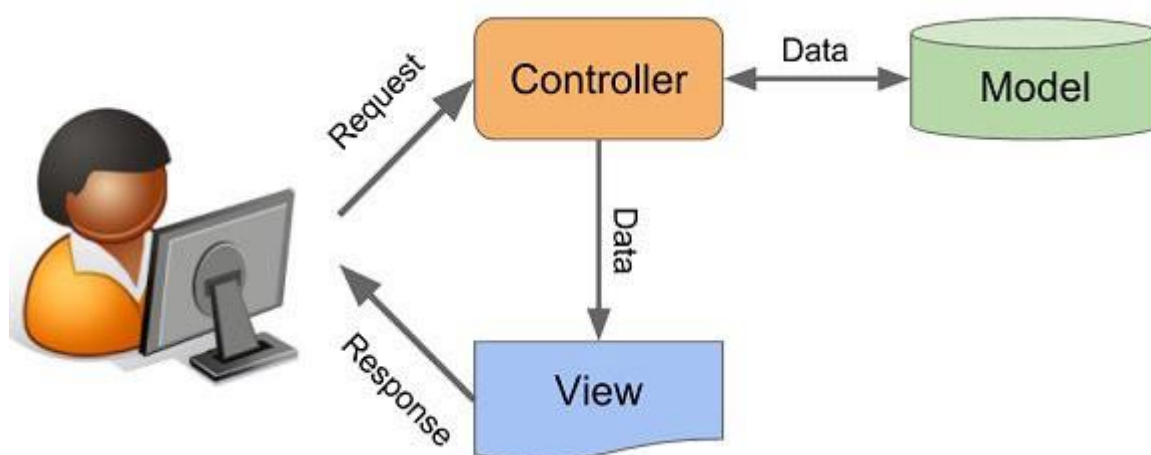


Figura 3-3: Diagrama Modelo Vista Controlador (MVC). Fonte: (Vilarinho, 2018)

### 3.4. Padrão de desenho utilizado

Padrão de desenho (design pattern) é uma solução reutilizável para um problema que ocorre com frequência dentro de um determinado contexto no projeto de software. É um modelo (template) de como resolver um problema que pode ser usado em muitas situações diferentes.

É uma representação abstrata que pode ser instanciada de várias maneiras. Entre os padrões que se utilizam neste contexto encontra-se o padrão geral de *software* para atribuição de responsabilidades (general responsibility assignment software patterns) (GRASP) (SOMMERVILLE, 2011).

#### 3.4.1. Padrão Grasp

Os padrões GRASP englobam uma série de princípios baseados em conceitos de orientação a objetos. Partindo de análises que procuram definir quais as obrigações dos diferentes tipos de objetos em uma aplicação, estes *patterns* disponibilizam uma série de recomendações que procuram favorecer a obtenção de sistemas melhor estruturados. Partindo de práticas consagradas no desenvolvimento de sistemas orientados a objetos, os padrões GRASP procuram fornecer diretrizes para a construção de aplicações bem estruturadas e que possam ser facilmente adaptáveis diante da necessidade de mudanças. A consequência direta das recomendações propostas por estes patterns é um código melhor organizado, de fácil manutenção e ainda, capaz de ser compreendido por diferentes desenvolvedores sem grande dificuldades (Devmedia, 2019). Os utilizados no desenho do sistema são os seguintes:

- **Controlador (*Controller*):** Quando aplicado atribui a responsabilidade por lidar com eventos do sistema a uma classe que não esteja relacionada a interface com o usuário. Na presente investigação este padrão pode ser observado na controladora Reunião (ReuniaoController).
- **Especialista (*Information Expert*):** propõe dar as responsabilidades as classes de acordo a informação que contém as mesmas, cumprindo assim um princípio básico e intuitivo da programação orientada a objectos. Na presente investigação este padrão pode ser observado quando queremos obter a lista de todas as tarefas, onde é necessário a model Tarefa e a criação do objecto tarefa para fornecer todos os dados.

### **3.4.2. O Padrão GoF**

Repertório de soluções e princípios que ajudam os desenvolvedores a criar software, e que são codificados em um formato estruturado, consistindo de:

- Nome
- Problema que soluciona
- Solução do problema

GoF é a sigla em inglês para o termo “Gang of Four”, um apelido dado ao esforço conjunto de 4 autores (Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides). Basicamente está dividido em três seções:

**Padrão de Criação:** Relacionados à criação de classes e objetos, ligados ao processo de instanciação.

**Padrão Estrutural:** Tratam da alteração da estrutura de um programa, e das associações entre classes e objetos.

**Padrão Comportamental:** Observam a maneira com que classes e objetos podem interagir.

## **3.5. Análise e desenho do banco de dados**

### **3.5.1. Modelo físico de banco de dados**

A modelagem física lida com o design do banco de dados real com base nos requisitos reunidos durante a modelagem lógica do banco de dados. Todas as informações coletadas são convertidas em modelos relacionais e modelos de negócios. Durante a modelagem física, os objetos são definidos em um nível denominado nível de esquema.

Um esquema é considerado um grupo de objetos que estão relacionados entre si em um banco de dados. Tabelas e colunas são feitas de acordo com as informações fornecidas durante a modelagem lógica. Chaves primárias, chaves exclusivas e chaves estrangeiras são definidas para fornecer restrições. Índices são definidos. (ELMASRI, 2011)

Abaixo se mostra na Figura 3-4, o modelo físico exibindo as relações entre as tabelas.

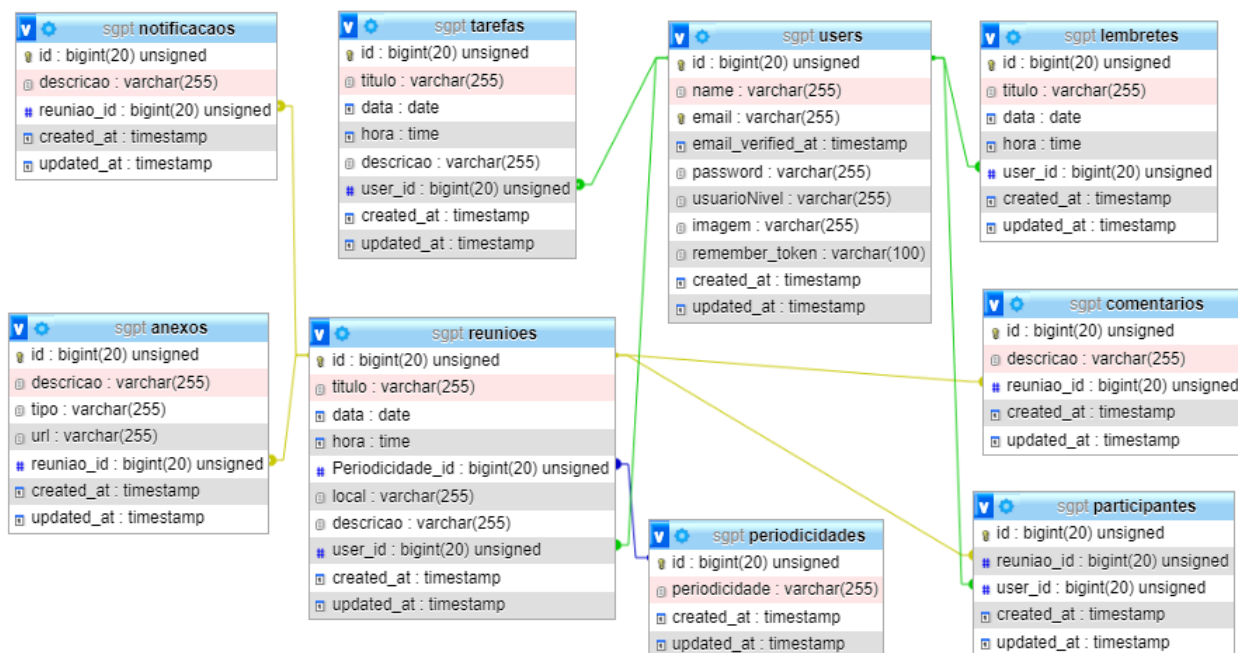


Figura 3-4: Modelo físico de banco de dados. Fonte: Elaboração própria

### 3.5.2. Descrição das Entidades

A Tabela 12 mostra a descrição das entidades.

Tabela 12: descrição das entidades

| Entidade       | Descrição   |
|----------------|---|
| useres         | Esta tabela armazena os dados dos usuários que acedem o sistema.                            |
| reunioes       | Esta tabela armazena os dados referente aos reuniões.                                       |
| periodicidades | Esta tabela armazena as periodicidades das reunioes.  |
| participantes  | Nesta tabela são armazenado os identificadores de cada usuário participante em uma reunião. |

|              |   |
|--------------|---|
| anexos       | Esta tabela armazena os documentos que serão anexados à uma determinada reunião |
| comentarios  | Esta tabela armazena os comentarios de cada reunião                             |
| notificações | Esta tabela armazena as notificações referente as reuniões                      |
| Tarefas      | Esta tabela armazena as tarefas do usuário                                      |
| lembretes    | Esta tabela armazena os lembretes do usuário                                    |

### 3.6. Modelo de despliegue(Implanação)

Os diagramas de instalação ou também de implantação mostram a disposição e descrevem a configuração de elementos de suporte de processamento, e de componentes de software, processos e objectos existentes nesses elementos (Videira, 2001) .

Abaixo se mostra na Figura 3-5, o modelo de implantação do sistema.

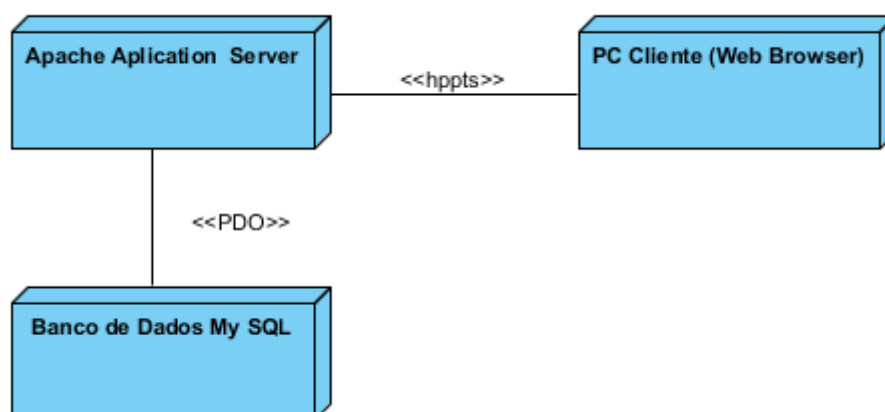


Figura 3-5: modelo de implantação. Fonte: Elaboração própria



### 3.6.1. Descrição do modelo de implantação

A Tabela 13 mostra a descrição do modelo de implantação do sistema.

Tabela 13: descrição do modelo de implantação

| Nós/Dispositivos           | Descrição  |
|----------------------------|--|
| PC_Cliente (Web Browser)   | Acedem a aplicação através de um computador, onde é executada mediante um navegador, sobre qualquer sistema operativo.   |
| Apache Application Server  | Servidor de Aplicações (em inglês Applications Server), é um servidor que disponibiliza um ambiente para a instalação e execução de certas aplicações, centralizando e dispensando a instalação nos computadores clientes. |
| Servidor de Banco de Dados | Neste se mantêm os dados que a aplicação usa e o sistema gestor de banco de dados é o MySQL.   |

A Tabela 14 mostra a descrição dos conectores usados no modelo de implantação do sistema.

Tabela 14: descrição dos conectores usados no modelo de implantação

| Conectores | Descrição   |
|------------|---|
| HTTPS      | HTTPS (sigla do inglês, Hyper Text Transfer Protocol Secure, protocolo de transferência de hipertexto seguro) Trata-se de um protocolo TCP/IP utilizado pelos servidores Web para a transferência e |

|     |   |
|-----|---|
|     | visualização de conteúdo Web de forma segura para o cliente (navegador).  |
| PDO | O PDO (PHP Data Object) é uma extensão da linguagem PHP para acesso a banco de dados. Totalmente orientado a objetos ele possui diversos recursos importantes, além de suporte a diversos mecanismos de banco de dados. |

### 3.7. Validação do Sistema

Quando se constrói um sistema informático antes que este venha ser usado ou entregue ao cliente, precisa ser testado de modos a verificar se cumpri com o propósito pelo qual foi projectado e funciona correctamente.

O teste faz parte de um processo no desenvolvimento do programa, podendo ser feito pelos próprios desenvolvedores ou, em alguns casos, feito por profissionais especializados na área. O procedimento tem como objetivo antecipar e corrigir falhas e bugs que apareceriam para o usuário final. (MONITORA, 2019)

#### **Tipos de testes**

Existem diferentes tipos de testes que podem ser aplicados para verificar se o sistema de gestão de plano de trabalho para a direção do instic atende a todos os requisitos identificados e o funcionamento correto do sistema de gestão foi atingido . Aqui estão os tipos de testes usados: teste funcional e o unitário.

#### **Método de teste**

Os Métodos de Teste de Software têm o objetivo de projectar testes que descubram diferentes tipos de erros com menos tempo e esforço. Existem dois fundamentos: Caixa Branca (White Box) para verificar a estrutura e Caixa Preta (Black Box) para a funcionalidade do software. No processo de teste em questão, o método de teste Black Box é usado (DEV MEDIA, 2012).

#### **3.7.1. Provas unitárias**

São provas realizadas pela equipa de desenvolvimento verificam que os componentes unitários estão codificados sob condições de robustez, suportando a entrada de dados

errados e inesperados e demonstrando assim a capacidade de tratar erros de maneira controlada. Neste teste unitário foi usada a prova de caixa branca, onde se aplicou o teste de caminho básico (CUNHA, 2021).

#### 3.7.1.1. Caixa branca

O analista de testes tem acesso ao código fonte, conhece a estrutura interna do produto sendo analisado, possibilita que sejam escolhidas partes específicas de um componente para serem avaliadas. Esse tipo de teste, também conhecido como teste estrutural, é projetado em função da estrutura do componente e permite uma averiguação mais precisa do comportamento dessa estrutura. Perceba que o acesso ao código facilita o isolamento de uma função ou ação, o que ajuda na análise comportamental das mesmas.

O teste de caminho básico é técnica onde calcula-se a complexidade lógica do software e utiliza esta medida como base para descobrir o caminho básico do software e exercendo o teste de modo que todos os caminhos sejam efetuados (CUNHA, 2021)

A seguir se mostra o fragmento do código na figura 3.6 onde aplicou-se o teste de caixa branca da presente solução:

```
1 public function convocar($id) {  
2     if (!$reuniao = Reuniao::find($id)) {  
3         return redirect()->back();  
4         $useres = User::orderBy('name')->paginate(20);  
         return view('reuniao_convocar', compact('reuniao', 'useres'));  
5     }
```

Figura 3-6: Código do teste de caixa branca. Fonte: Elaboração própria

A seguir, se mostra na figura 3.7 o gráfico de fluxo de caixa branca feito no sistema.

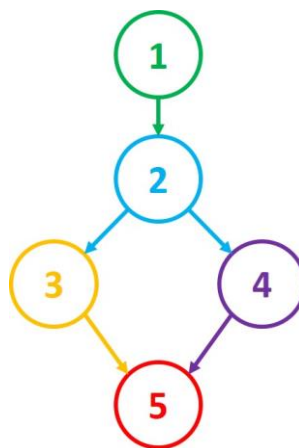


Figura 3-7: Gráfico de fluxo, teste de caixa branca. Fonte: Elaboração própria

### 3.7.2. Provas funcionais

Provas de funcionamento também chamado testes funcionais (Caixa preta) são métodos e técnica dinâmicas, que envolve executar uma implementação de software com os dados de teste e examinar suas saídas e comportamento operacional, a fim de verificar se ele está funcionando conforme o esperado (Souza, 2008). O teste de caixa preta tenta encontrar erros das seguintes categorias:

- Funções incorretas ou ausentes,
- Erros de interface,
- Erros nas estruturas de dados ou acesso a bancos de dados externos,
- Erros de desempenho e erros de inicialização e finalização.

Este método é implementado por meio de técnicas de provas, um das quais é a técnica de partição equivalente.

Técnica de Partição Equivalente trata-se de uma técnica de testes funcionais que propõe a separação das possíveis entradas em categorias diferentes. Partições de equivalência podem ser encontradas em dados válidos e inválidos (valores que deveriam ser rejeitados, por exemplo). As partições podem ser identificadas para valores de saída, valores relativos ao tempo (antes ou depois de um evento), bem como valores internos ao processo (Costa, 2013).

#### **Desenho de caso de teste**

Os casos de teste são um conjunto de ações com resultados esperados com base nos requisitos de especificação do sistema. Conjuntos de dados válidos e inválidos foram usados

para realizar os testes aplicando a técnica de particionamento de equivalência. Em seguida, o caso de teste é mostrado graficamente projectando ao SGPT-INSTIC associada ao caso de uso “Cadastrar usuário” .

Tabela 15: Provas de Caixa Preta

| Condições de entrada                        | Classes de equivalência válida         | Classes de equivalência não válidas  |
|---|--|--------------------------------------|
| O campo nome é obrigatório                  | nome = "Ivânia Quiosa"                 | nome = " "                           |
| O campo e-mail é obrigatório                | E-mail=<br>"ivypatricya1213@gmail.com" | Email= " "                           |
| O campo Tipo de usuário é obrigatório       | Tipo de usuário = "funcionario"        | Tipo de usuário = " "                |
| O campo senha é obrigatório                 | Senha = " jvd568 "                     | Senha= " "                           |
| O campo confirmação da senha é obrigatório. | confirmação da senha = "<br>jvd568 "   | <u>confirmação</u> da senha = "<br>" |

**Teste valido do cadastro de usuário:**

**Antes:** preenchendo todos os campos, que são: nome, email, tipo de usuário, senha e confirmação de senha.

**Bem vindo/a, Administrador do sistema**  
Area administrativa

**Cadastrar Usuário**

Nome completo: Ivânia Quiosa

E-mail: ivypatricia1213@gmail.com

Tipo de usuário: Funcionário

Senha: [obscured]

Confirmação da senha: [obscured]

**Salvar**

Figura 3-8: prova funcional de cadastrar usuário. Fonte: Elaboração própria

**Depois:** preenchido todos os campos e clicando no botão cadastrar, o sistema mostra uma mensagem informando sucesso ao cadastrar este usuário.

**Bem vindo/a, Administrador do sistema**  
Area administrativa

**Usuários**

| Usuário | Nome                     | E-mail                    | Nível de acesso | Ação                   |
|---------|--------------------------|---------------------------|-----------------|------------------------|
|         | Administrador do sistema | admin@gmail.com           | Admin           | <a href="#">Editar</a> |
|         | Inácio Antônio José      | inacio@ak.com             | Funcionario     | <a href="#">Editar</a> |
|         | Ivânia Quiosa            | ivypatricia1213@gmail.com | Funcionario     | <a href="#">Editar</a> |
|         | Kaisa de Almeida         | kaisa@hotmail.com         | Funcionario     | <a href="#">Editar</a> |
|         | Kapinge de Almeida       | kapingealmeida@gmail.com  | Funcionario     | <a href="#">Editar</a> |
|         | Laravel 8 com bootstrap  | a@a.com                   | Admin           | <a href="#">Editar</a> |

Figura 3-9: prova funcional de cadastrar usuário. Fonte: Elaboração própria

### Teste inválido de cadastro de usuário:

Quando não se preenche os dados o sistema mostra a seguinte mensagem: “\*O campo x é obrigatório”, como se vê na Figura

The screenshot displays the administrative interface of the SGPT-INSTIC system. The top header includes the logo 'SGPT-INSTIC', a welcome message 'Bem vindo/a, Administrador do sistema', and the text 'Area administrativa'. A sidebar on the left contains navigation links: 'Painel de controle', 'Usuários' (with sub-links 'Cadastrar' and 'Usuários'), and 'Minha conta'. The main content area features a 'Cadastrar Usuário' form with the following fields: 'Nome completo' (containing 'Ivânia Quilosa'), 'E-mail' (containing 'E-mail' and a red error message '\*O campo e-mail é obrigatório.'), 'Tipo de usuário' (a dropdown menu with 'Funcionário' selected), 'Senha' (containing 'Senha'), and 'Confirmação da senha' (containing 'Confirmar senha'). A blue 'Salvar' button is positioned at the bottom of the form.

Figura 3-10: prova funcional de cadastrar usuário. Fonte: Elaboração própria

## Resultados das Provas

Ao aplicar as provas de funcionalidades, foram elaborados 80 casos de provas. Depois de uma primeira iteração se obtiveram 60 não conformidades das quais 20 foram resolvidas, na segunda iteração se obtiveram 40 não conformidades das quais resolveram-se 10 conformidades, na terceira iteração se obtiveram 30 não conformidades das quais 20 foram resolvidas, na quarta iteração se obtiveram 10 não conformidades e foram resolvidas 10. Depois de corrigidas estas deficiências se procedeu a realização de uma quinta iteração de provas obtendo resultado satisfatório, dando por concluída as provas.

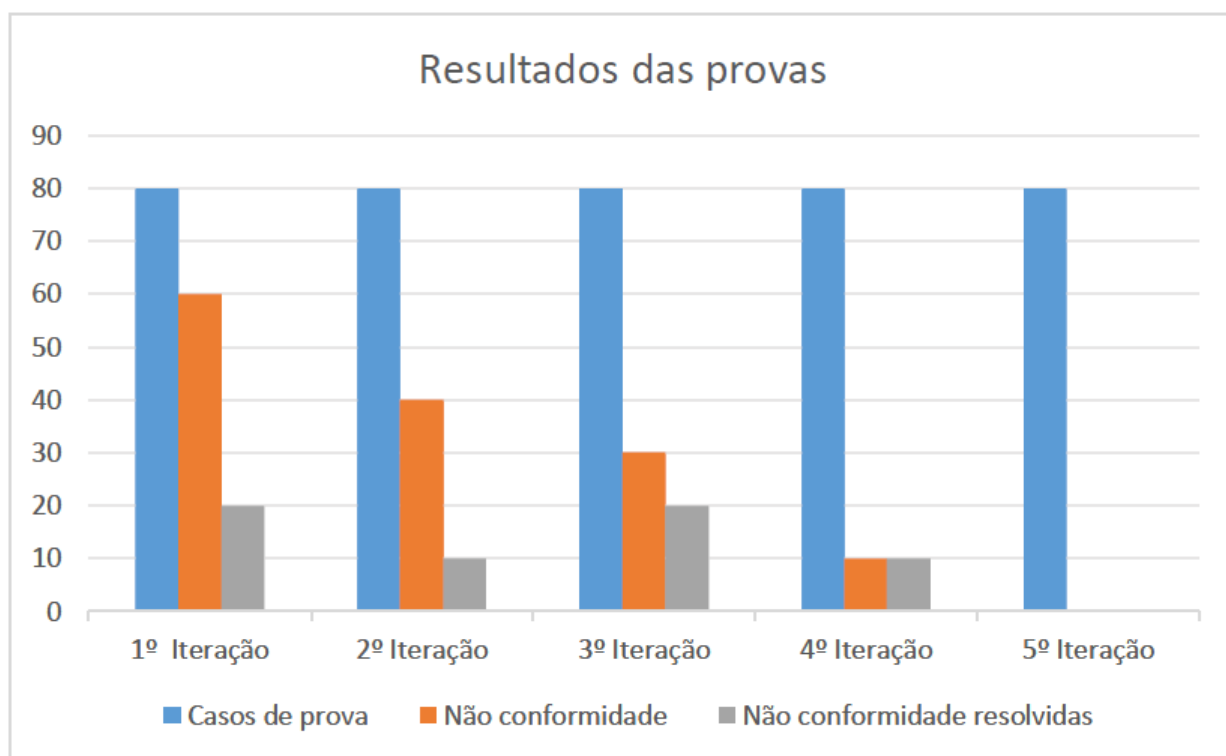


Figura 3-11: Resultados das provas. Fonte: Elaboração própria

### 3.8. Conclusões parciais do capítulo

Depois de varias análises aprofundadas, desencadeadas nos diagramas estabelecido, como a de classes simples e com estereótipos, a de componentes, que por sua vez facultou o surgimento do banco de dados e, baseando no estudo da implantação nos seus respectivos diagramas, nota – se que em base neste deu a origem ao surgimento do Sistema de Pano de Trabalho para a Direção do INSTIC, tendo sido aprovado com êxito e cumprindo com todos os requisitos que se esperava; logo pode – se dizer que este software dará solução a problemática encontrada na gestão de Plano de Trabalho do INSTIC.



## 2. Conclusões

Com a elaboração deste trabalho e desenvolvimento desta investigação que foi apresentada no início, por meio do cumprimento de seus objectivos específicos conduziu as seguintes conclusões:

1. O estudo do marco teórico-metodológico da investigação e dos sistemas similares para a gestão de Plano de Trabalho Para a Direção do INSTIC , permitiu definir uma proposta que facilitasse dar soluções as necessidades apresentadas pela Direção do INSTIC.
2. Com base as metodologias, ferramentas e tecnologias usadas, o sistema foi implementado seguindo a metodologia OpenUp e as demais ferramentas utilizadas permitiram que o sistema fosse desenvolvido, alcançando assim os objectivos por este oferecidos. Usou-se a linguagem de programação php 7.1 com a utilização do framework Laravel 8 para a parte de back-end e o bootstrap 4 para poder fazer a parte voltada ao front-end, usou-se ferramenta UML, Visual Paradigm 13.1 e o My SQL 6.
3. Os requisitos registados durante a investigação, permitiram ter uma visão mais clara sobre as necessidades a serem implementadas e o funcionamento do sistema proposto. O processo de modelagem do negócio e do sistema executado resultou nos diagramas e artefactos necessários para orientar o desenvolvimento do sistema de gestão de Plano de Trabalho Para a Direção do INSTIC.
4. A etapa de desenho e implementação da solução desenvolvida permitiram extrair os elementos fundamentais para o funcionamento do sistema de gestão de Plano de Trabalho Para a Direção do INSTIC e permitiu também cumprir na totalidade com os requisitos funcionais identificados.
5. As provas realizadas permitiram detectar falhas, e melhora-las possibilitando ter um sistema seguro e com qualidade.

### 3. Recomendações.

Após a conclusão do trabalho, constatou-se que os objectivos pelos quais foi criado foram alcançados, para a continuidade recomenda-se o seguinte:

- Implementar uma versão mobile.

## 4. Referências bibliográficas.

**1STWEBDESIGNER. 2019.** PHP vs Ruby vs Python: The Three Programming Languages in a Nutshell. [Online] 2019. [Citação: 1 de Junho de 2019.] <https://1stwebdesigner.com/php-vs-ruby-vs-python/>.

**ALMEIDA, Guilherme Augusto Machado de. 2017.** ALMEIDA, Guilherme Augusto Machado de. 2017. Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis. ed. São Paulo: Departamento de Engenharia de Produção, 2017. [Online] 2017.

**ANDRADE, Ana Paula De. 2019.** O que é Laravel?. [Online], 16 de Maio de 2019. [Citação: 2 de junho de 2021.] <https://www.treinaweb.com.br/blog/o-que-e-laravel/>. [Online] 2019.

**BARBARÁ, Saulo. 2008.** *Gestão por processos: fundamentos, técnicas e modelos de implementação*. Rio de Janeiro : Qualutymark, 2008.

**BATISTA, Emerson de Oliveira. 2004.** *Sistema de Informação: o uso consciente da tecnologia para o gerenciamento*. São Paulo : Saraiva, 2004.

**Bruno Feitosa. 2019.** Comece a usar XP — eXtreme Programming no seu time Ágil. <https://medium.com/>. [Online] 9 de janeiro de 2019. [Citação: 22 de Maio de 2021.] <https://medium.com/@brunofeitosa/comece-a-usar-xp-extreme-programming-no-seu-time-%C3%A1gil-5664cf0da100>.

**BUILDER. 2017.** *Conheça quais são os principais tipos de métodos ágeis. Recuperado em 03 de Agosto de 2018, de Project Builder: https://www.devmedia.com.br/processo-de-teste-agil-x-tradicional/36854*. [Online] 2017.

**CALDEIRA, Carlos Pampulim. 2011.** *Introdução aos sistemas de gestão*. Évora : Universidade de Évora, 2011.

**Costa, Pedro. 2013.** Partição de equivalência em testes de software. Base2. [Online] 12 de Setembro de 2013. [Citação: 13 de Março de 2018.] <http://www.base2.com.br/2013/09/12/particao-equivalencia-teste-software/#>. [Online] 2013.

**CUNHA, Simone. 2021.** Estratégias de testes. [Online], 2021. [Citação: 5 de Julho de 2021.] <http://testwarequality.blogspot.com/p/estrategias-de-testes.html>. [Online] 2021.

**DEVMEDIA. 2013.** *DEVMEDIA. (2013). Artigo SQL Magazine 42 - Modelagem de dados com a Visual Paradigm - Do modelo de classes à criação do banco de dados. Recuperado el 13 de 07 de 2021, de Devmedia: <https://www.devmedia.com.br/artigo-sql-magazine-42modelagem-de-dados-com-a>. [Online] 2013.*

— **2016.** Introdução ao VISUAL STUDIO CODE. [Online],2021. [Citação: 30 de Maio de 2021.] <https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418> . [Online] 2016.

— **2012.** Testes funcionais de software. [Online], 2012. [Citação: 5 de Julho de 2021.] <https://www.devmedia.com.br/testes-funcionais-de-software/23565>. [Online] 2012.

— **2018.** Integrando XP as principais metodologias ágeis. <https://www.devmedia.com.br>. [Online] 2018. [Citação: 22 de maio de 2021.] <https://www.devmedia.com.br/integrando-xp-as-principais-metodologias-ageis/30989>.

**Editorial Conceito. 2020.** Conceito de Agenda, definição e o que é. <https://queconceito.com.br/>. [Online] 2020. [Citação: 19 de Junho de 2021.] <https://queconceito.com.br/agenda>.

**Editorial Conceitos. 2016.** Plano de Trabalho - Conceito, o que é, Significado. <https://conceitos.com>. [Online] 11 de novembro de 2016. [Citação: 22 de maio de 2021.] <https://conceitos.com/plano-trabalho>.

**ELMASRI, Ramez. 2011.** *ELMASRI, Ramez.2011. Fundamentals of Database Systems, 6th edition. . 2011.*

**Erich Gamma,Richard Helm, Ralph Johnson e john vissides. 1994.** *Design Patterns: Elements of Reusable Object-Oriented Software". 1994.*

**Fundação Nacional da Qualidade. 2018.** *Sistemas de Gestão.* São Paulo : Fundação Nacional da Qualidade, 2018.

**HALL, Pearson. 2011.** Modelagem. Nova Jersey. Citação (Hall, 2011). 2011.

**L3SOFTWARE. 2017.** *Visual Paradigm Modeler | L3 Software. Recuperado el 03 de Agosto de 2018, de L3 Software: <http://l3software.com.br/produto/visual-paradigmmodeler/>. [Online] 2017.*

**MACHADO, Felipe Nery Machado. 2015.** *Análise e Gestão de Requisitos de Software - Onde Nascem Os Sistemas - 3ª Edição. Rio de Janeiro.Editora:SARAIVA. 2015.*

**MELGOZA, J. 2016.** 6 buenas razones para usar symfony. Recuperado el 13 de 06 de 2021, de jonathanmelgoza: <https://jonathanmelgoza.com/blog/6-buenas-razones-para-usar-symfony/>. [Online] 2016.

**Monitora. 2020.** As principais metodologias de desenvolvimento de software que você precisa saber. <https://www.monitoretec.com.br>. [Online] Monitora, 20 de agosto de 2020. [Citação: 22 de maio de 2021.] <https://www.monitoretec.com.br/blog/metodologias-de-desenvolvimento-de-software/>.

**MONITORA, Equipa. 2019.** os tipos de testes de software.[Online] 11 de Fevereiro de 2019. [Citação: 5 de Julho de 2021.] <https://www.monitoretec.com.br/blog/quais-os-tipos-de-testes-de-software-e-por-que-automatiza-los/>. [Online] 2019.

**NTCHOSTING. 2019.** PHP Programming Language. [Online] 2019. [Citação: 1 de Julho de 2019.] <https://www.ntchosting.com/encyclopedia/scripting-and-programming/php/>.

**NUGROHO, PRASETYA. 2017.** PHP Codeigniter MVC Concept For Dummies. [Online] 1 de Julho de 2017. [Citação: 24 de Julho de 2021,] <https://seegatesite.com/codeigniter-mvc-concept-for-dummies-with-simple-example/>. [Online] 2017.

**OLIVEIRA, Bruno Carazato. 2018.** *Gaia protótipo: um modelo de prototipação Para processos de desenvolvimento de Software*. Londrina : s.n., 2018.

**OLIVEIRA, Djalma de Pinho Rebouças de. 2002.** *Sistemas de informação gerenciais: estratégias, táticas, operacionais*. São Paulo : Atlas, 2002.

**PEREIRA, Maria Dayane Macario. 2018.** *ANÁLISE DE MÉTODOS DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: UMA METODOLOGIA DE PESQUISA COMPARATIVA ENTRE MÉTODOS ÁGEIS E TRADICIONAIS*. CASTANHAL – PA : s.n., 2018.

**PRESSMAN, Roger S. 2016.** *Engenharia de Software, Uma Abordagem Profissional*, 8º ed. Porto Alegre : AMGH, 2016.

**REZENDE, Denis Alcides e ABREU, Aline França de. 2000.** *Tecnologia da informação aplicada a sistemas de informação empresariais: o papel estratégico da informação e dos sistemas de informação nas empresas*. São Paulo : Atlas, 2000.

**Rodrigues, Nadja e Jr., Jorge Dias. 2015.** *Modelagem de Negócio - A importância de entender o negócio antes de começar o desenvolvimento de projetos de software*.

ResearchGate. [Online] 29 de julho de 2015. [Citação: 06 de Março de 2018.] <https://www.devmed.2015>.

**RODRIGUEZ, V.R Martius. 2010.** *Gestão empresarial: organizações que aprendem*. Rio de Janeiro : Qualitymark, 2010.

**SCRIPTCASE. 2021.** Linguagem de Programação WEB. [Online], 2021. [Citação: 30 de Maio de 2021.] <http://www.scriptcase.com.br/linguagem-programacao-web/> . [Online] 2021.

**Silva, Alberto Manuel Rodrigues da e Videira, Carlos Alberto Escaleira. 2001.** *SILVA, Alberto Manuel Rodrigues da e Videira, Carlos Alberto Escaleira. 2001. UML, Metodologias e Ferramentas CASE. Lisboa/Portugal : Centro Atlântico, 2001. 972-842636-4. 2001.*

**SOMMERVILLE, I. 2011.** *Engenharia de Software*. 9. ed. São Paulo : Pearson Prentice Hall, 2011.

**Souza, Ellen Polliana Ramos. 2008.** *RBTPProcess: Modelo de Processo de Teste de Software baseado em Riscos*. [Dissertação de Mestrado] Recife : s.n., 2008. CDU(681.300:681.3.06). 2008.

**The PHP Group. 2019.** What is PHP? [Online] 2019. [Citação: 1 de Julho de 2019.] <https://www.php.net/manual/en/intro-what-is.php>.

**TIESPECIALISTAS. 2015.** *OpenUP: uma visão geral*. Recuperado el 13 de 03 de 2018, de tiespecialistas: <https://www.tiespecialistas.com.br/2010/09/openup-uma-visao-geral/>. [Online] 2015.

**Universidade Federal de Uberlândia. 2020.** O que é um Plano de Trabalho? | PROPLAD/UFU. <http://www.proplad.ufu.br/>. [Online] 4 de agosto de 2020. [Citação: 22 de maio de 2021.] <http://www.proplad.ufu.br/perguntas-frequentes/o-que-e-um-plano-de-trabalho>.


**VENTURA, Plinio. 2019.** O que é UML (Unified Modeling Language). [Online], 31 de Janeiro 2019. [Citação: 10 de junho de 2021. ] <https://www.ateomomento.com.br/diagramas-uml/>. [Online] 2019.

**Videira, Alberto Manuel Rodrigues da Silva e Carlos Alberto Escaleira. 2001.** *UML, Metodologias e Ferramentas CASE. Porto - Lisboa : s.n., 2001. 972-8426-36-4. 2001.*

**Vilarinho, Leonardo. 2018.** MVC – framework: usando a arquitetura sem código de terceiros. *medium.com*. [Online] medium.com, 27 de 02 de 2018. [Citação: 25 de 10 de 2021.] <https://medium.com/trainingcenter/mvc-framework-usando-a-arquitetura-sem-c%C3%B3digo-de-terceiros-bf95a744c66d>.





**ZENARO, Flávia dos Santos. 2012.** *A utilização do Scrum em um sistema web: um estudo de caso*. 2012.


# Anexos





Bem vindo/a, **Kapinge de Almeida**  
Departamento de ...


Nova actividade ▾



 Painel de controle

 Actividades >

 Estatística >

 Minha conta >

**Reuniões**

Discussão de projecto dos finalistas  
23 de agosto de 2021

Jornada científica  
10 de novembro de 2021

Encontro com os delegados de turma  
15 de outubro de 2021

[Mostrar todas](#) →

**Tarefas**

Despache nº 234  
23 de agosto de 2021

Escrever informe  
10 de novembro de 2021

Criar horário  
15 de outubro de 2021

[Mostrar todas](#) →


**Lembretes**

Entrega do projecto  
25 de outubro de 2021

Pre-defesa  
Novembro de 2021





Defesa  
Dezembro de 2021

[Mostrar todos](#) →



Bem vindo/a, **Kapinge de Almeida**  
Departamento de ...

Nova actividade ▾



**Actividades**

Reuniões

Tarefas

Lembretes

**Estatística**


Estatística


**Minha conta**

Perfil

Sair

**Discussão de projecto dos finalistas**

**Informações** 

**Anexos** 

2021-11-28


10:57:15

Sala de reuniao do Instic


Kapinge de Almeida


Não se repete


**Sobre a reunião**  
presença indispensável


**Participantes** 

4 participantes

 **Inácio António José**  
inacio@ak.com

 **Ivânia Quiosa**  
ivypatricia1213@gmail.com

 **Kaia de Almeida**  
kaia@hotmail.com

 **Kapinge de Almeida**  
kapingealmeida@gmail.com

60