Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

# Grata - Gerenciamento de Reuniões e ATAs

Autor: Victor Hugo Lopes Mota
Orientador: Dr. Wander Cleber Maria Pereira da Silva

Brasília, DF
2019

Victor Hugo Lopes Mota

# Grata - Gerenciamento de Reuniões e ATAs

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Wander Cleber Maria Pereira da Silva

Brasília, DF

2019

Victor Hugo Lopes Mota
Grata - Gerenciamento de Reuniões e ATAs/ Victor Hugo Lopes Mota. – Brasília, DF, 2019-
24 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Wander Cleber Maria Pereira da Silva

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Gerenciamento de Reuniões. 2. Otimização. I. Dr. Wander Cleber Maria Pereira da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Grata - Gerenciamento de Reuniões e ATAs

CDU 02:141:005.6

Victor Hugo Lopes Mota

# Grata - Gerenciamento de Reuniões e ATAs

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Trabalho aprovado. Brasília, DF, 14 de abril de 1912:

---

**Dr. Wander Cleber Maria Pereira da Silva**
Orientador

---

**Membro convidado 1**
Convidado 1

---

**Membro convidado 2**
Convidado 2

Brasília, DF
2019

*Dedicatória ficara aqui*

# Agradecimentos

AQUI VÃO FICAR OS AGRADECIMENTOS

# Resumo

Resumo do projeto

**Palavras-chaves**: Gerenciamento de Reuniões, Otimização.

# Abstract

Here go to abstract

**Key-words**: Meeting management,Optimization.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

Grata, *Gerenciamento de Reuniões e ATAs*

MAS, *Síndrome de Aceitação Sem Sentido*

NMIL, *Núcleo de Modernização da Informação Legislativa*

TCC, *Trabalho de Conclusão de Curso*

MVC, *Model-View-Controller*

# Sumário

# 1 Introdução

## 1.1 Apresentação do Tema

Um dos instrumentos mais fundamentais e que são cada vez mais crescentes na vida organizacional de uma empresa são as reuniões. Segundo (ALLEN, 2016), já se gastam até 15% do tempo coletivo da organização com reuniões.

Encontros institucionais que não são produtivas e sem sentido, é o que (DAVID, 2013) chama de "Síndrome de Aceitação Sem Sentido"(MAS). David define o MAS como "um reflexo involuntário em que uma pessoa aceita um convite de reunião sem sequer saber o porquê. Uma doença comum entre o escritório e trabalhadores em todo mundo". Atividades que deveriam ser simples e rápidas se tornam complicadas com várias reuniões para a execução completa delas. Reuniões não são nenhum ponto de prazer entre dentro de uma instituição, contudo se os próprios funcionários não conseguem ver o sentido da reunião e os tópicos abordados, mostra que a empresa como um todo está fadada ao fracasso.

Nesse viés, reuniões é um dos meios usados para programar uma atividade, reunir pessoas em busca de uma solução relacionada ao problema X. Composta por vários encontros sem tópicos e objetivos específicos, sem *feedbacks* aos gerentes e sem atas sobre o que foi discutido, as reuniões infelizmente se tornam um problema para gerentes e seus funcionários para as instituições. Problemas como estes podem ser pela falta de investimento disponível para aplicação na área tecnológica ou até mesmo pela priorização de outras necessidades.

Este trabalho visa propor uma solução de *software* para auxiliar a condução de reuniões, utilizando dos conhecimentos adquiridos no curso de Engenharia de Software para o desenvolvimento de uma solução que se adeque às reais necessidades empregadas aos gerentes dos projetos organizacionais.

## 1.2 Justificativa

Tendo como premissas os problemas em reuniões apresentados no tópico anterior (1.1), uma das soluções para aumentar a produtividade em reuniões é através de um sistema *web* que auxilie os gerentes e líderes de reuniões a gerenciar seus encontros de forma rápida, intuitiva e gratuita para que qualquer organização possa usar os recursos do *software* com o objetivo de melhorar a organização de sua empresa.

O Sistema GRATA, vem oferecer a solução prática para a melhoria do controle das

informações e qualidade dos serviços. Tendo como a principal funcionalidade o registro das Atas de reuniões de forma simples e intuitiva, tanto para quem gerencia como para quem participa. Além da automação dos processos essenciais da organização, o sistema fornece relatórios gerenciais e analíticos, que podem ser usados para identificação de pontos de melhoria ou até mesmo para dar visibilidade a questões específicas.

## 1.3 Problema de Pesquisa

O crescente problema com reuniões mal gerenciadas seja por gerentes não capacitados, ou por falta da especificação prévia dos tópicos a serem abordados, levam diretamente a reuniões mal sucessididas e com isso desperdício de tempo e dinheiro. Estima-se que empresas gastam em média US $ 37 bilhões anualmente em reuniões (DRAKE, 2014). O custo real desses encontros impulsionou a (HARVARD, 2016) a criar uma calculadora que ajuda gerentes a calcularem o verdadeiro custo de um encontro.

Nessa viés e utilizando a justificativa desenvolvida no tópico 1.2, é possível se ter o problema de pesquisa. A problemática a ser resolvida neste projeto é: *Como desenvolver um sistema para auxiliar a condução de reuniões em organizações?*

### 1.3.1 Metodologia

A metodologia abordada neste trabalho teve sua escolha baseada após a realização de pesquisas comparativas entre metodologias tradicionais como o (PMBOK, 2012) e a metodologia ágil de *software*.

Por conta de um conhecimentos maior sobre a metodologia e com entregas frequentes em menos tempo, a metodologia escolhida para este projeto no desenvolvimento do sistema foi a metodologia ágil, juntamente com algumas práticas do *Scrum* e baseado nos valores do *Lean Software Development*.

### 1.3.2 Requisitos

Requisito não é um termo usado apenas pela Engenharia de Software . Há casos em que requisitos são apenas uma declaração abstrata em alto nível de um serviço ou restrição que um sistema deve oferecer.

(SOMMERVILLE, 2011) os define como: "Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações."

Requisitos podem ser definidos em duas categorias: Requisitos Funcionais e Requisitos Não-Funcionais, ambos serão definidos a seguir.

#### 1.3.2.1 Requisitos Funcionais

Os requisitos funcionais descreve o que o sistema deve de fato ser. Requisitos funcionais podem ser tão específicos quanto necessário,por exemplo, podem ter sistemas com requisitos funcionais gerais e outros que além de refletir os sistemas, também abrangem as formas de trabalho de uma organização. Requisitos funcionais de um sistema deve ser completo, isso quer dizer que todos os serviços requisitados pelo usuário devem ser definidos.

#### 1.3.2.2 Requisitos Não-Funcionais

Requisitos não-funcionais são requisitos que são relacionados as propriedades do sistema como confiabilidade, tempo de espera, desempenho, segurança e até restrições do sistema. Requisitos não-funcionais podem possui tanta relevância quanto os requisitos funcionais, pois em uma reunião de levantamento de requisitos, o cliente sonha o mundo e não está atento se os recursos os próprios recursos e os recursos da emprega conseguem atender ao requisito. Um requisito não-funcional não atendido pode inclusive inutilizar um projeto. Exemplo disso é caso um sistema de uma aeronave não consiga atingir a confiabilidade necessária, não será dado o certificado de segurança para operar, sendo assim a aeronave não poderá voar.

### 1.3.3 Processo de Desenvolvimento de Software

Segundo (SOMMERVILLE, 2011), esse processo pode ser definido como "Um processo de *software* é um conjunto de atividades relacionadas que levam à produção de um produto de *software*."

Neste trabalho, foram definidas as principais atividades a serem realizadas para alcançar o objetivo final de ter um sistema gratuito que auxilie os gerentes a otimizar suas reuniões por meio computacional:

- Especificação do *software*: funcionalidades e restrições do *software*;
- Projeto e implementação do *software*: as especificações que o *software* deve atender;
- Validação de *software*: para que atenda as expectativas do cliente, o *software* deve ser validado pelo mesmo;
- Evolução do *software*: o *software* deve ser capaz de ser extensível a mudanças, tendo assim seu código aberto.

É nessa fase que são definidas a arquitetura e a linguagem de *software*.

### 1.3.4 Arquitetura de Software

A arquitetura de *software* é como o sistema deve ser organizado com a estrutura geral do projeto. A arquitetura possui um valor alto dentro da construção de um *software*, pois nela se tem o elo entre o projeto e a engenharia de requisitos. Possui o dever identificar os principais componentes estruturais no sistema e o relacionamento entre eles. Neste projeto a arquitetura é o MVC *(model-view-controller)*.

#### 1.3.4.1 Model-View-Controller

O padrão arquitetural MVC é responsável de responsabilidades em camadas. A primeira é *Model*(modelo), que é responsável pela manipulação de dados, ou seja, leitura, escrita de dados e também suas validações é de responsabilidade da Model. A segunda camada é a *View*(visão), que possui a responsabilidade de interação com o usuário. Por último se tem a *Controller*(controladora), responsável por receber as aquisições do usuário. A controller também tem o dever de disponibilizar os dados para a *view* e assim ocorrer a interação com o usuário.

### 1.3.5 Linguagem de Software

Linguagem de programação são instruções passadas de maneira que o computador entenda e apresente um retorno. Existem diversas linguagens de programação, desde a mais baixo a alto nível.

Linguagens de *software*, como também podem ser chamadas, são divididas em duas frentes: *front-end* e *back-end*. Ambas serão explicadas nos tópicos a seguir.

#### 1.3.5.1 Front-end

A programação de um *software* pelo ponto de vista do *front-end* é a visão final do usuário com o sistema. *Front-end* é a *View* do MVC, como explicado no tópico 1.3.4.1. Existem diversos tipos de *frameworks* que auxiliam os desenvolvedores a trabalhar com essa frente, como:

- *Bootstrap*
- *Materialize*
- *Material UI*
- *Angular 4*

A linguagem *front-end* escolhida para este projeto, foi a *Angular 4*, pois além de facilitar o desenvolvimento e interação com usuário final, é uma das mais utilizadas ao redor do mundo, então é facilita uma manutenção futura do *software*.

#### 1.3.5.2 Back-end

A programação *back-end* possui as responsabilidades da *model* e *controller* no padrão arquitetural MVC. Em conjunto o *Angular 4*, que é a linguagem *front-end* deste projeto, a *back-end* possui o dever de tratar os dados, valida-los e fomenta-los a visão do usuário.

Existem diversas linguagens *back-end* que auxiliam os desenvolvedores a trabalhar em uma linguagem que o computador entende, como:

- *Python Django-Rest*
- *Java*
- *Ruby on Rails*
- *C*

A linguagem *back-end* escolhida para este projeto, foi a *Python Django-Rest*, pois tem uma ótima conexão com a linguagem *front-end*, e por ser muito utilizada, possibilita assim uma manutenção futura.

## 1.4 Objetivos

### 1.4.1 Objetivos Gerais

### 1.4.2 Objetivos Específicos

# 2 Fundamentação Teórica

## 2.1 Reuniões Tradicionais vs Reuniões no Modelo Ágil

### 2.1.1 Reuniões Tradicionais

### 2.1.2 Reuniões Ágeis

# 3 Objetivos

## 3.1 Objetivo Geral

O objetivo do projeto é propor novos processos de reuniões e gerenciamento dos documentos gerados no ciclo de desenvolvimento dos projetos, contando com o suporte de um software gratuito e de código aberto para automatizar alguns dos trabalhos manuais, tornando os processos mais ágeis e com armazenamentos seguros.

## 3.2 Objetivos Específicos

# 4 Metodologia

# 5 Proposta de Solução

## 5.1 Definição dos Requisitos

### 5.1.1 Requisitos Funcionais

### 5.1.2 Requisitos Não-Funcionais

# 6 Planejamento de Desenvolvimento

## 6.1 Cronograma TCC

# 7 Conclusão

# Referências

ALLEN, L.-W. Meetings as a positive boost? how and when meeting satisfaction impacts employee empowerment. *Journal of Business Research*, 2016. Acesso em: 25.04.2019. Citado na página 13.

DAVID, G. *How to save the world (or at least yourself) from bad meetings*. 2013. Disponível em: <https://www.ted.com/talks/david_grady_how_to_save_the_world_or_at_least_yourself_from_bad_meetings>. Acesso em: 22.04.2019. Citado na página 13.

DRAKE, B. 37 billion is lost every year on these 12 meeting mistakes. *Business Insider*, 2014. Disponível em: <https://www.businessinsider.com/37-billion-is-lost-every-year-on-these-meeting-mistakes-2014-4>. Acesso em: 29.04.2019. Citado na página 14.

HARVARD, B. *Estimate the Cost of a Meeting with This Calculator*. 2016. Disponível em: <https://hbr.org/2016/01/estimate-the-cost-of-a-meeting-with-this-calculator>. Acesso em: 29.04.2019. Citado na página 14.

PMBOK, P. *Um Guia do Conhecimento em Gerenciamento de Projetos 5ª edição*. [S.l.]: Saraiva, 2012. Acesso em: 30.04.2019. Citado na página 14.

SOMMERVILLE, I. *Engenharia de Software 9ª edição*. [S.l.]: Pearson Education do Brasil, 2011. Acesso em: 01.04.2019. Citado 2 vezes nas páginas 14 e 15.