Agendaí

Critiana de Paulo
ENGENHARIA DE SOFTWARE
UFERSA
Pau dos Ferros, Brasil
cristiana.paulo@alunos.ufersa.edu.br

José Lucas Santana de Andrade
ENGENHARIA DE SOFTWARE
UFERSA
Pau dos Ferros, Brasil
jose.andrade82065@alunos.ufersa.edu.br

Klebson Davi de Sousa Magalhães
ENGENHARIA DE SOFTWARE
UFERSA
Pau dos Ferros, Brasil
klebson.magalhaes@alunos.ufersa.edu.br

Romulo Ismael Pereira Sobreira
ENGENHARIA DE SOFTWARE
UFERSA
Pau dos ferros, Brasil
sobreiraromuloes@gmail.com

I. Introdução

Nos dias de hoje, a administração do tempo é uma habilidade fundamental para a maioria das pessoas, especialmente aquelas que precisam lidar com muitas tarefas e responsabilidades diferentes. Para ajudar nesse processo, vários sistemas de gerenciamento de tarefas foram desenvolvidos, oferecendo aos usuários uma variedade de recursos e funcionalidades.

Neste artigo, apresentamos um sistema de agendamento de tarefas que visa facilitar a organização e o gerenciamento de atividades diárias, semanais e mensais. O sistema tem como objetivo oferecer uma solução simples, eficiente e acessível para as pessoas que precisam gerenciar suas tarefas com eficiência, sem complicações.

O `Agendaí` é um sistema de agendamentos feito para ajudar a vida de pessoas com um cotidiano muito agitado, a fim de facilitar seus compromissos, sejam eles pessoais ou profissionais. Servindo como um método para organizar a rotina, permitindo cadastrar itens como: atividades diárias, lembretes de horários de treino, estudo, reuniões de trabalho, etc. Agregando a estas tarefas, o usuário também poderá definir prioridades para cada atividade cadastrada, assim como estabelecer prazos para elas e acompanhá-las ao longo do tempo.

Ao longo deste artigo, descreveremos em detalhes a arquitetura e o funcionamento do sistema de agendamento de tarefas, bem como suas principais funcionalidades e recursos. Além disso, discutiremos as tecnologias utilizadas no desenvolvimento do sistema e trabalhos futuros a respeito do assunto.

Esperamos que este artigo possa contribuir para o desenvolvimento de soluções eficazes e acessíveis para o gerenciamento de tarefas, facilitando a vida das pessoas e aumentando a produtividade.

No decorrer deste trabalho, será apresentada uma visão de como o desenvolvimento do sistema deve se comportar e quais artifícios devem ser utilizados na visão arquitetural.

II. Referencial teórico

Nesta seção, são discutidos os seguintes assuntos:

- A tecnologia da informação no cotidiano das pessoas
- Engenharia de Software
- Gerência de configuração e mudanças
- O desenvolvimento de sistemas web
- Frameworks para desenvolvimento web

A. Tecnologia da informação no cotidiano das pessoas

Segundo [1], A tecnologia da informação desempenha um papel cada vez mais importante no cotidiano das pessoas, afetando significativamente a forma como elas vivem, trabalham, se comunicam e se relacionam.

Para [2], através da tecnologia tem-se uma nova possibilidade de acesso a produtos, transmissão, arquivos e a informação. Toda essa transformação acaba influenciando a economia, política e o cenário social. A criação destas novas tecnologias no setor da informática, segundo as autoras, gerou um mercado cada vez mais competitivo e especializado do qual precisa estar sempre atualizado com os padrões de mercado.

[2] ainda afirmam que a comunicação entre pessoas e empresas se tornou algo essencial, pois tem-se a possibilidade de acessar uma grande quantidade de informações e trocar estes dados em tempo real. Por meio dos computadores, os serviços foram melhorados e teve-se uma substituição de uma porcentagem da mão de obra pela tecnologia que proporcionou novas profissões no setor da informática e das comunicações.

B. Engenharia de software

Segundo [3] a Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software [4]. A engenharia de software foca no software como produto. Não entra neste escopo os softwares construídos apenas para passarem o tempo dos programadores [5].

No desenvolvimento de um projeto de software, quanto mais complexo é o software, maior é o empenho que o engenheiro de software deve fazer para o desenvolver e tem que ter maior gerenciamento [6]. A engenharia de software abrange diversas áreas de conhecimento, como engenharia de requisitos, design de software, programação, testes de software, gerenciamento de projetos de software e manutenção de software.

Sabendo disso, todos os conceitos retratados no artigo em questão envolvem a produção de diagramas para auxiliar na produção do produto de software e sua manutenibilidade de uso. A engenharia de software está em constante evolução, e novas técnicas, ferramentas e processos estão sendo desenvolvidos continuamente para ajudar a melhorar a qualidade e a eficiência do processo de desenvolvimento.

C. Gerência de configuração e mudanças

De acordo com [7] a gerência e configuração de mudanças são processos importantes em qualquer organização que busca implementar mudanças eficazes em seus sistemas e processos. Esses processos envolvem a identificação de necessidades de mudança, a avaliação de seu impacto, a criação de planos de implementação, a realização de testes e a implementação propriamente dita.

A Gerência de Configuração de Software é formada pelas atividades de Controle de Versão, Controle de Mudanças e Integração Contínua. O controle de versão (ou versionamento) é uma prática comum em desenvolvimento de software que visa gerenciar as alterações feitas em um conjunto de arquivos ao longo do tempo. O objetivo principal é permitir que várias pessoas trabalhem no mesmo conjunto de arquivos sem conflitos, garantindo a integridade do código-fonte e a rastreabilidade das mudanças. [7]

O controle de mudanças envolve a criação de um processo formal para gerenciar as mudanças, incluindo a identificação das mudanças, avaliação dos impactos, autorização, implementação, verificação e documentação das mudanças realizadas. O processo pode variar dependendo do tamanho e complexidade do projeto, mas geralmente envolve a criação de uma equipe responsável pelo gerenciamento das mudanças, a definição de critérios para a avaliação das mudanças e a criação de uma política de controle de mudanças que estabeleça as diretrizes para o processo. [7]

Já a integração contínua é baseada em um processo automatizado que envolve a construção e a execução de testes em cada mudança de código-fonte, garantindo que o software funcione corretamente em todas as etapas do processo de desenvolvimento. [7]

Tendo isso em vista, é de grande importância a aplicação de gerência e configuração de mudanças em qualquer projeto, ainda que sejam seus conceitos ou no mínimo a aplicação do controle de versões. Isso ajudará a equipe no desenvolvimento, mantendo o software alinhado com os objetivos e imprevistos possíveis de ocorrer.

D. O desenvolvimento de sistemas web

De acordo com [8], ocorreu uma grande transformação nos meios de comunicação nas últimas décadas, impulsionada pela chegada da Internet, o que levou a significativas mudanças no desenvolvimento de sistemas de informação. Com o avanço tecnológico, a Internet passou de um meio de divulgação de informações para uma plataforma de comunicação e desenvolvimento.

[9] estabeleceram uma relação entre este contexto e o surgimento de um novo modelo de desenvolvimento de sistemas: a criação de soluções web. No início, as aplicações web eram compostas de páginas com conteúdo estático, institucional e textual. No entanto, hoje em dia, os sistemas evoluíram para soluções maiores e mais complexas, capazes de fornecer dados dinâmicos, acesso a bancos de dados e integração com outras aplicações. As aplicações web necessitam de uma estrutura para a apresentação de suas informações, e para isso é usada a *HyperText Markup Language* (HTML) [10].

De acordo com [10], a HTML é considerada a linguagem padrão para acessar páginas na Internet. Os navegadores interpretam as linhas de código em HTML e apresentam o resultado para o usuário. A linguagem HTML é composta por textos e códigos especiais, chamados de *tags*. Para visualizar os resultados de um código em HTML, basta ter um navegador capaz de interpretá-lo.

Além disso, [10] destaca a importância do JavaScript como uma linguagem de *script* capaz de auxiliar na criação de páginas web interativas. Por meio dela, é possível exibir mensagens e efeitos dinâmicos em uma página. Para utilizar o JavaScript, é necessário ter apenas um navegador que suporte essa linguagem.

No entanto, para facilitar o desenvolvimento *web*, é comum utilizar *frameworks*, que serão explicados a seguir.

E. Frameworks para desenvolvimento web

Conforme destacado por [11], um *framework* é uma coleção de classes que trabalham em conjunto para simplificar o processo de desenvolvimento de uma solução, evitando a necessidade de utilizar diversas soluções para um mesmo problema.

Dentre as vantagens do uso de *frameworks*, [12] destaca a padronização no uso de convenções, classes e bibliotecas, o que torna a manutenção do sistema mais fácil, mesmo que o código tenha sido desenvolvido por outra equipe. Além disso, o autor destaca a automação de tarefas para o reaproveitamento de conteúdo, como outra vantagem oferecida pelos *frameworks*.

1) NestJs: NestJS é um *framework* para construção de aplicativos Nodejs escaláveis e eficientes. Ele foi criado com base no *framework* popular Angular e compartilha muitos conceitos e características com ele. O NestJS utiliza TypeScript como linguagem principal e possui uma arquitetura modular e flexível, permitindo que os desenvolvedores escolham as ferramentas e bibliotecas mais adequadas para cada projeto.

O *framework* possui uma série de recursos que facilitam o desenvolvimento de aplicativos, como o uso de injeção de dependência para gerenciamento de objetos e serviços, a possibilidade de criar *middlewares* personalizados para tratar requisições HTTP, suporte a *WebSockets* e a utilização de *decorators* para a criação de controladores e módulos.

2) React: React é uma biblioteca de JavaScript para a construção de interfaces de usuário [13]. Desenvolvido e mantido pelo *Facebook*, *React* permite que os desenvolvedores

criem interfaces de usuário complexas e interativas de forma eficiente [14].

A principal característica do *React* é a utilização de componentes reutilizáveis, que possibilitam a criação de interfaces modulares e escaláveis. Cada componente é uma peça isolada de código que pode ser utilizado em diversas partes da aplicação, permitindo que os desenvolvedores criem interfaces complexas sem a necessidade de reescrever código repetitivo [14].

Outra característica importante do *React* é o seu modelo de programação baseado em estado. Com o *React*, cada componente pode ter um estado interno que pode ser atualizado dinamicamente, o que possibilita a criação de interfaces dinâmicas e responsivas [14].

Além disso, o *React* possui uma comunidade ativa e grande quantidade de recursos disponíveis, incluindo bibliotecas, ferramentas e documentação. Isso torna mais fácil para os desenvolvedores aprenderem e utilizarem o *React* em seus projetos [14].

3) Ionic: O Ionic é um *framework* de desenvolvimento de aplicativos híbridos que segundo [15], está em alta no mercado devido à sua grande produtividade e facilidade na utilização de seus recursos, já que grande parte dos programadores conhecem as linguagens exigidas pelo *framework* (HTML, CSS, Javascript). Por meio destas linguagens, é possível desenvolver um código que será exportado como aplicativo para ser executado em plataformas como *Android*, *IOS* a partir do Ionic.

III. O Sistema: AgendaÍ

Nesta seção, são descritas as principais características do sistema, cuja arquitetura deve seguir e quais restrições deve obedecer.

A. ARQUITETURA DO SISTEMA

Um software pode se tornar uma estrutura complexa; por isso, precisa-se representá-lo de diferentes formas, que venham a auxiliar quem o desenvolveu e quem poderá contribuir futuramente com ele [16]. Neste sentido, o sistema precisa de diferentes representações para possibilitar diferentes visões, de forma a melhorar o entendimento e tomada de decisões, em diferentes contextos [16]. As visões podem ser modeladas de diferentes formas; uma delas é o uso de diagramas, buscando descrever o funcionamento e interações da aplicação.

Em uma arquitetura de software existem basicamente três categorias de visões: a visão de módulos, componente e conector e de implantação [16]. A visão de módulo tem como característica apresentar os elementos estruturais do sistema. Por sua vez, a visão componente e conector possui por base a representação dos elementos, que podem ser reutilizáveis em sistemas e definem elementos de mais alto nível e suas interligações/comunicações ao longo da aplicação. A visão de implantação, por sua vez, está voltada para definir como os elementos são implantados ou mesmo atribuídos às equipes de desenvolvimento [16].

Neste trabalho, a documentação das diferentes visões da arquitetura da aplicação é feita por meio da *Unified Modeling Language* UML [17], a Linguagem de Modelagem Unificada; uma linguagem visual para modelar softwares, em especial, aqueles baseados no paradigma de orientação a objetos [18]. Trata-se de uma linguagem de uso difundido, que possui uso amplo na industria voltada ao desenvolvimento de software [17].

B. SISTEMA DE VERSIONAMENTO

Para o sistema de versionamento, será utilizado o *Github* que é um serviço baseado em nuvem que hospeda um sistema de controle de versão inglês *Version Control System*(VCS) chamado *Git*. Ele permite que os desenvolvedores colaborem e façam mudanças em projetos compartilhados enquanto mantêm um registro detalhado do seu progresso [19].

C. JUSTIFICATIVA PARA A ESCOLHA DO SISTEMA DE VERSIONAMENTO

Existem alguns motivos para os quais se deve usar o Github dentre eles são:

- Controle de Versão: É um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas mais tarde. O GitHub é uma plataforma de controle de versão que permite fazer justamente este acompanhamento facilitando a identificação de problemas e correção dos mesmos. [20]
- Colaboração: O GitHub permite que várias pessoas trabalhem em um projeto ao mesmo tempo, colaborando e contribuindo para o código e documentação. Isso é particularmente útil para equipes de desenvolvimento de software, pois ajuda a aumentar a produtividade e a qualidade do código. Esta configuração oferece muitas vantagens, especialmente sobre VCSs locais. Por exemplo, todo mundo sabe, até certo ponto o que todo mundo no projeto está fazendo. Os administradores têm controle refinado sobre quem pode fazer o que. [20]
- Comunidade: O GitHub é uma excelente ferramenta para o trabalho em equipe. Quando se trata de um software ou um site que precisa ser criado em conjunto, por exemplo, a plataforma online facilita, sem dúvidas, a gestão do projeto [21]. Por este motivo facilita na contribuição de projetos que já existem e na criação de novos.
- Integração: O GitHub se integra facilmente com outras ferramentas de desenvolvimento, como editores de código, ferramentas de construção e integração contínua. Você pode instalar integrações em sua conta pessoal ou em organizações que possui. Você também pode instalar GitHub Apps a partir de um repositório específico em um repositório específico em que você tem permissões de administrador ou que pertencem à sua organização.Tudo isso facilita a integração com os sistemas de desenvolvimento. [22]
- Acesso Remoto: O GitHub permite acessar seu código de qualquer lugar do mundo, desde que você tenha uma

conexão com a internet. Uma URL remota é outra forma de o Git dizer ”o lugar onde seu código é armazenado”. A URL poderia ser seu repositório no GitHub, ou a bifurcação de outro usuário, ou até mesmo em um servidor totalmente diferente. [22]

Em resumo, o GitHub é uma plataforma muito útil para desenvolvedores de software que desejam controlar seu código-fonte, colaborar com outras pessoas, receber *feedback* e ajuda da comunidade, integrar ferramentas de desenvolvimento e acessar seu código de qualquer lugar.

D. REQUISITOS FUNCIONAIS

Para [4], o levantamento de requisitos é importante para identificar as principais funcionalidades e características a serem incluídas no sistema. As etapas de elicitação e análise de requisitos trabalham com o domínio do problema e tentam determinar “o que” o software deve fazer e se é realmente possível desenvolver o software solicitado. O sistema proposto neste trabalho, consiste em uma aplicação web que funciona como uma agenda para tarefas. Os requisitos levantados são descritos a seguir:

- **[RF001] Cadastro de Tarefas**: O sistema deve permitir que o usuário insira suas atividades diárias, como compromissos, reuniões, tarefas domésticas, dentre outras.
- **[RF002] Edição de Tarefa**: O sistema deve permitir que o usuário edite as tarefas que cadastrou.
- **[RF003] Excluir Tarefas**: O sistema deve permitir que o usuário exclua as tarefas cadastradas.
- **[RF004] Listar Tarefas**: O sistema deve permitir que o usuário possa listar tarefas cadastradas, inclusive por filtro de data.
- **[RF005] Priorização de Tarefas**: O sistema deve permitir que o usuário possa definir a ordem de importância de cada tarefa, para que ele possa priorizar cada tarefa cadastrada. .
- **[RF006] Definição de prazos**: O sistema deve permitir que o usuário estabeleça um prazo para a conclusão de cada tarefa cadastrada, a fim de organizar e priorizar as atividades de acordo com sua urgência.
- **[RF007] Alertas e notificações**: O sistema deve enviar alertas e notificações para lembrar o usuário das atividades que precisam ser realizadas, bem como as que estão atrasadas, garantindo que o mesmo não as esqueça.
- **[RF008] Acompanhamento de progresso**: O sistema deve permitir que usuário possa acompanhar o progresso de suas tarefas cadastradas, para que o mesmo possa avaliar se está cumprindo seus objetivos e realize ajustes, quando necessário.
- **[RF009] Visualização de calendário**: O sistema deve apresentar as tarefas do usuário em um calendário, facilitando a visualização e o gerenciamento das atividades.
- **[RF010] Integração com outras aplicações**: O sistema deve permitir se integrar com outros aplicativos e serviços, como e-mail, calendário e notas, para garantir que o usuário possua todas as informações importantes em um só lugar.
- **[RF0011] Login de Usuário**: O sistema deve permitir que o usuário faça login no sistema utilizando a conta *Google* ou com sua conta já cadastrada.
- **[RF0012] Cadastro de Usuário**: O sistema deve permitir que o usuário se cadastre utilizando a conta *Google* ou com formulário com campos Nome, E-mail e Senha.
- **[RF0013] Edição de Usuário**: O sistema deve permitir que o usuário possa alterar seu Nome, E-mail ou senha.
- **[RF0014] Exclusão de Usuário**: O sistema deve permitir que o usuário exclua sua conta do sistema.

E. REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais descrevem propriedades dos sistemas, tais como restrições de serviços, funções do sistema, tempo, desenvolvimento e padrões [4].

Como se trata de um sistema *web*, que manipula lembretes e tarefas do dia a dia de uma pessoa, os requisitos não funcionais identificados se referem a usabilidade, atratividade, inteligibilidade, segurança e manutenção, que são descritos a seguir.

- Usabilidade: o usuário deve executar as interações com o sistema em uma velocidade aceitável, com poucas ocorrências de erros no uso do sistema.
- Atratividade: O sistema deve ter elementos que facilitem a interação do usuário.
- Inteligibilidade: O sistema deve conter uma interface interativa, ou seja, de fácil uso, a exemplo de cores consistentes e ícones intuitivos.
- Segurança: O sistema deve garantir a segurança dos dados, pois se tratam, muitas vezes, de informações do dia a dia do usuário e pode ser considerado sigiloso sua rotina.
- Manutenção: O sistema deve permitir fácil manutenção e entrega, quando solicitado, ou correção de algum problema identificado.

F. DIAGRAMAS

- Diagrama de Classes

Os elementos estruturais do sistema podem ser representados por um diagrama de classes. Buscando identificar as principais classes que formam o sistema, foi utilizada a técnica de Análise Textual de Abbot (ATA) [23], empregando, para isso, a descrição dos requisitos que compõem o sistema proposto.

- Entidade Relacionamento

De acordo com [24], o Modelo Entidade Relacionamento (também chamado Modelo ER, ou simplesmente MER), como o nome sugere, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

Em geral, este modelo representa de forma abstrata a estrutura que o banco de dados da aplicação possuirá. É possível que o banco de dados contenha várias outras entidades, tais

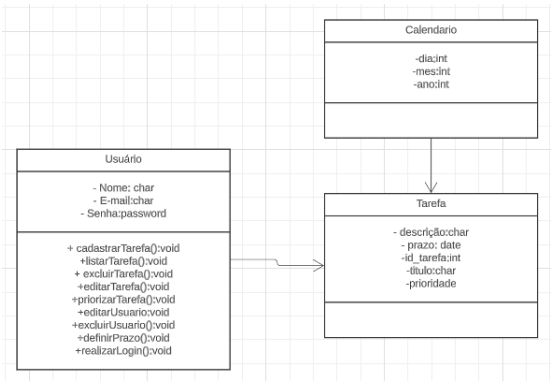

Figura 1. Diagrama de Classes

como: chaves e tabelas intermediárias, que podem só fazer sentido no contexto de bases de dados relacionais.

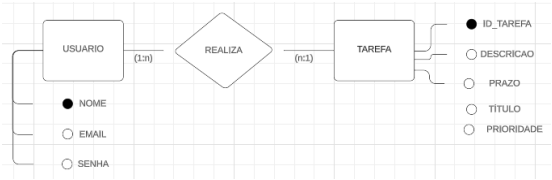

Figura 2. Modelo Entidade Relacionamento

G. RESTRIÇÕES

As restrições são decisões importantes para o desenvolvimento do sistema e que a arquitetura do sistema deve suportar [16].

O sistema proposto é projetado para ser utilizado na Web e deve ficar disponível para qualquer usuário que deseje organizar sua rotina com o `Agendaí`. Para o desenvolvimento da aplicação, algumas restrições estão presentes:

- O sistema deve ser projetado usando HTML5, JavaScript para o *front-end*, juntamente com os *frameworks React* e Ionic;
- O sistema deverá ser projetado usando NestJS e Firebase para o *back-end*.

No contexto de aplicações *web*, o *front-end* é uma área de desenvolvimento que constrói a parte que interage de forma direta com o usuário [25]. Como exemplo de tecnologias empregadas para o desenvolvimento do lado *front-end* da aplicação, inclui-se o HTML5, que é uma linguagem de marcação usada para definir a estrutura da página; o *CSS*, que define a estética dos elementos e o JavaScript, que é a parte que irá dar funcionalidade aos elementos definidos. Mesmo com essas tecnologias, vários *frameworks* podem ser utilizados para auxiliar no desenvolvimento *front-end*, tais como o *React*.

Seguindo o processo, tem-se o *back-end*, que vem a ser a parte lógica da aplicação, alimentando a *front-end* do sistema [25]. O *back-end* trabalha também a interação com o modelo de dados da aplicação e várias tecnologias podem ser usadas no desenvolvimento *back-end*, dentre elas Nodejs e tecnologias que usam a linguagem Java, tais como Servlets, JSP e JSF. No tocante, ao modelo de dados, podem ser utilizados MySQL, Firebase e afins.

IV. TRABALHOS RELACIONADOS

Esta seção, apresenta uma análise de algumas propostas sobre sistemas de agendamentos presente trabalho.

1) Desenvolvimento de um sistema de agenda online para consultório médico utilizando Laravel [26]: Neste trabalho, é apresentado o desenvolvimento de um sistema de agenda online utilizando o *framework* Laravel. O sistema possui recursos de agendamento, cancelamento e confirmação de consultas, além de permitir a geração de relatórios e o envio de lembretes por e-mail.

2) Sistema de gerenciamento de agenda para salão de beleza [27]: Este trabalho descreve o desenvolvimento de um sistema de gerenciamento de agenda para um salão de beleza. O sistema permite que os clientes agendem seus horários pelo site ou aplicativo, e que os funcionários gerenciem suas agendas através de uma interface simples e intuitiva. O sistema também possui recursos para controle de estoque e emissão de relatórios.

3) Sistema de agenda eletrônica para clínica veterinária [28]: Neste trabalho, é apresentado o desenvolvimento de um sistema de agenda eletrônica para uma clínica veterinária. O sistema permite que os clientes agendem consultas e serviços pela internet, além de permitir que os funcionários gerenciem suas agendas e cadastrem informações dos animais e seus proprietários.

4) Desenvolvimento de um sistema de agenda para escritório de advocacia [29]: : Este trabalho descreve o desenvolvimento de um sistema de agenda para um escritório de advocacia. O sistema permite que os advogados gerenciem suas agendas, marquem reuniões e audiências, e recebam lembretes por e-mail. O sistema também possui recursos para emissão de relatórios e controle de prazos.

V. TRABALHOS FUTUROS

Algumas sugestões de trabalhos futuros para o sistema `Agendaí` incluem:

- Adição de recursos de colaboração: uma possível expansão para o sistema de agenda seria a adição de recursos de colaboração, permitindo que os usuários compartilhem suas agendas com outras pessoas, como colegas de trabalho ou membros da família.
- Integração com outros aplicativos: uma melhoria potencial seria a integração com outros aplicativos, como calendários ou plataformas de gerenciamento de tarefas, permitindo que os usuários sincronizem suas informações com outras ferramentas que já utilizam.
- Desenvolvimento de aplicativos móveis: outra possível expansão seria o desenvolvimento de aplicativos móveis para *iOS* e *Android*, permitindo que os usuários acessem suas agendas de qualquer lugar e em qualquer dispositivo.

Essas são apenas algumas possibilidades de trabalhos futuros para o sistema `Agendaí`. É importante lembrar que essas

sugestões devem ser baseadas nas necessidades e expectativas dos usuários, para que o sistema possa contribuir para a melhoria da qualidade de vida dos usuários e aumentar a eficiência no gerenciamento de tarefas e compromissos diários.

VI. CONSIDERAÇÕES FINAIS

Neste trabalho, foi proposta a descrição de desenvolvimento de um sistema de agendamentos com o propósito de ser uma ferramenta útil para ajudar as pessoas a gerenciar suas tarefas e compromissos diários. Motivada pela necessidade e o benefício de ter uma rotina organizada. A metodologia de desenvolvimento permite a criação de uma aplicação web que atende aos requisitos funcionais e não funcionais descritos.

No entanto, o sistema de agenda ainda tem espaço para melhorias futuras, como a inclusão de recursos de colaboração para compartilhamento de agendas entre usuários e a integração com outros aplicativos para sincronização de dados. Além disso, é importante continuar aprimorando a usabilidade e a eficiência do sistema, a fim de fornecer uma ferramenta ainda mais útil para os usuários.

Por fim, pode-se afirmar que o sistema de agenda para pessoas é uma ferramenta valiosa para ajudar os usuários a gerenciar suas tarefas e compromissos diários. Espera-se que este trabalho possa contribuir para a melhoria da qualidade de vida dos usuários, aumentando a sua produtividade e reduzindo o estresse no gerenciamento de suas tarefas diárias.

REFERÊNCIAS

[1] M. Castells, *A Sociedade em REDE*. Paz e Terra, 2009.

[2] K. KOHN and M. C. H., *O impacto das novas tecnologias na sociedade: conceitos e características da Sociedade da Informação e da Sociedade*. Digital. In, 2007.

[3] "Conceitos de software e engenharia de software," https://www.devmedia.com.br/conceitos-de-software-e-engenharia-de-software/15730, acessado em 5 de maio de 2023.

[4] R. S. Pressman, *Egenharia de software*. Pearson, 1995, vol. 3.

[5] P. Filho, *Tecnologia da Informação no Cotidiano das Pessoas*. Editora Érica, 2009.

[6] P. Jalote, *An Integrated Approach to Software Engineering*. Springer, 2005.

[7] "Gerência de configuração e mudanças," https://www.devmedia.com.br/gerencia-de-configuracao-e-mudancas/31327, acessado em 5 de maio de 2023.

[8] V. T. Yoshiura, "Desenvolvimento e implantação de um sistema web para monitoramento da rede de atenção em saúde mental," Master's thesis, Programa de Pós-Graduação Interunidades Bioengenharia - Escola de Engenharia de São Carlos / Faculdade de Medicina de Ribeirão Preto / Instituto de Química de São Carlos da Universidade de São Paulo, São Carlos, 2015.

[9] Y. Deshpande, S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke, and B. White, "Web engineering," *Journal of Web Engineering*, vol. 1, no. 1, pp. 3–17, May 2002.

[10] C. J. COSTA, *Desenvolvimento para web*. Lisboa: Lusocrédito, 2007.

[11] P. Alvim, *Tirando o Máximo do Java EE 6 Open Source com jCompany Developer Suite*. Pearson, Powerlogic Publishing, 2010.

[12] E. L. Minetto, *Frameworks para Desenvolvimento em PHP*. Novatec, 2007.

[13] R. Developers, "React: A javascript library for building user interfaces," *React Documentation*, 2021. [Online]. Available: https://reactjs.org/

[14] A. Banks and E. Porcello, *Learning React: Functional Web Development with React and Redux*. O'Reilly Media, 2017.

[15] A. J. R. Gonçalves, "Desenvolvimento de aplicativos híbridos com o ionic framework," *Picos (PI)*, vol. 1, no. 1, pp. 500–515, jun. 2017.

[16] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-wesley professional, 2012, vol. 3.

[17] G. T. A. Guedes, *UML 2 - Uma Abordagem Prática - 3ª Edição: uma Abordagem Prática*. Novatec, 2018, vol. 3.

[18] "O que é um diagrama uml?" https://www.lucidchart.com/pages/pt/o-que-e-uml, acessado em 5 de maio de 2023.

[19] Andrei L., "O que é github e como usá-lo," https://www.hostinger.com.br/tutoriais/o-que-github, Mar. 2023.

[20] "1.1 começando - sobre controle de versão," https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Sobre-Controle-de-Vers%C3%A3o, acessado em 12 de maio de 2023.

[21] "Entenda de uma vez o que é github e a importância dele num negócio," https://rockcontent.com/br/blog/o-que-e-github/, acessado em 12 de maio de 2023.

[22] "Sobre integrações," https://docs.github.com/pt/get-started/exploring-integrations/about-integrations, acessado em 12 de maio de 2023.

[23] E. Bezerra, *Princípios de análise e projeto de sistemas com UML*. Elsevier, 2007.

[24] "Mer e der: Modelagem de bancos de dados," https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332, acessado em 5 de maio de 2023.

[25] H. K. Ferreira and J. D. Zuchi, *análise comparativa entre frameworks frontend baseados em javascript para aplicações web.* revista interface tecnológica 15.2, 2018, vol. 111-123.

[26] R. Silva, "Desenvolvimento de um sistema de agenda online para consultório médico utilizando laravel," 2019, especialização em Desenvolvimento Web com Interfaces Ricas.

[27] M. Santos, "Sistema de gerenciamento de agenda para salão de beleza," 2020, curso Técnico em Informática.

[28] C. Silveira, "Sistema de agenda eletrônica para clínica veterinária," 2018, curso Técnico em Informática.

[29] A. Rocha, "Desenvolvimento de um sistema de agenda para escritório de advocacia," 2017, curso Técnico em Informática para Internet.