ISSN 1808-6136

ISSN on-line 2674-7499

TICKET: AVALIAÇÃO DO PROCESSO DE CRIAÇÃO DE UM SISTEMA DE GERENCIAMENTO DE CHAMADOS

MIGUEL PEREIRA MEDEIROS[1], LUCIANA ROCHA CARDOSO[2]; EZEQUIAS FERREIRA DE SOUZA[3]; LUDMILA FURTADO BREDER[4], ANDRÉIA ALMEIDA MENDES[5], RITA DE CÁSSIA FERREIRA PEDROSA LAZARONI[6]

1 Acadêmico de Tecnologia em Análise e Desenvolvimento de Sistemas do Centro Universitário UNIFACIG, Manhuaçu – MG, 2010563@sempre.unifacig.edu.br.
2 Mestre em Ciência da Computação pela Universidade Federal de Viçosa (UFV), Centro Universitário UNIFACIG, Manhuaçu – MG, luroca@sempre.unifacig.edu.br.
3 Mestre em Desenvolvimento Local pelo Centro Universitário Augusto Motta (UNISUAM), Centro Universitário UNIFACIG, Manhuaçu - MG, ezequias.souza@sempre.unifacig.edu.br.
4 Mestre em Informática pela Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Centro Universitário UNIFACIG, Manhuaçu – MG, Ludmila@sempre.unifacig.edu.br
5 Doutora em Linguística pela Universidade Federal de Minas Gerais (UFMG), Centro Universitário UNIFACIG, Manhuaçu-MG, andreialetras@yahoo.com.br.
6 Doutora em Educação pela Universidad del Mar, Faculdade Univertix, Matipó-MG, lazaroni@uai.com.br

RESUMO

O trabalho com projetos surge como uma possibilidade de não só trabalhar com conteúdos de forma significativa, como também de desenvolver habilidades e competências nos alunos que serão necessárias à sua vida profissional. O artigo em questão objetiva, justamente, apresentar o processo de criação de um desses projetos, criados para conclusão de um dos semestres letivos do curso. Trata de um sistema de documentação do Ticket, um *software* de gerenciamento de chamados feito para o auxílio no envio e recebimento de chamados em uma organização, prestando controle, gerência e métricas sobre o suporte e atendimento de clientes. Trata-se de uma pesquisa de caráter descritivo, desenvolvida com base em pesquisa exploratória e fundamentado a partir de levantamento bibliográfico, baseado primordialmente nos levantamentos de Pinheiro e Magalhães (2007) e Welligton (2011) sobre o funcionamento e importância dos sistemas de suporte e nas ideias de Pressman (2016) sobre todas as etapas da construção de software, desde o levantamento de requisitos até a entrega da aplicação final. Observou-se que a utilização da metodologia de projetos como estratégia de ensino e como forma de avaliação dos alunos favorece a aprendizagem, uma vez que contextualiza os conteúdos fundamentais trabalhados no decorrer do semestre, bem como desenvolve a autonomia dos alunos, contribuindo para uma formação crítica e reflexiva.

Palavras-chave: Sistema de Suporte; Atendimento ao Cliente; Chamados; Tickets; Projetos.

TICKET: EVALUATION OF THE PROCESS OF CREATING A CALL MANAGEMENT SYSTEM

Recebido em 21/08/2021 e aprovado para publicação em 23/08/2021

ABSTRACT

Working with projects emerges as a possibility to not only work with content in a meaningful way, but also to develop skills and competencies in students that will be necessary for their professional life. The article in question aims, precisely, to present the process of creating one of these projects, created for the conclusion of one of the academic semesters of the course. It deals with a ticket documentation system, a call management software made to assist in sending and receiving calls in an organization, providing control, management and metrics on customer support and service. This is a descriptive research, developed based on exploratory research and based on a bibliographic survey, based primarily on the surveys of Pinheiro and Magalhães (2007) and Welligton (2011) on the functioning and importance of support systems and pressman's ideas (2016) on all stages of software construction, from requirements gathering to delivery of the final application. It was observed that the use of the project methodology as a teaching strategy and as a way of evaluating students favors learning, since it contextualizes the fundamental contents worked during the semester, as well as develops the autonomy of students, contributing to a critical and reflective training.

Keywords: Support System; Customer Service; Called; Tickets; Projects.

1 INTRODUÇÃO

O cenário educacional tem desafiado a todos os educadores a repensarem a sua prática pedagógica, uma vez que a atual realidade da docência desafia os professores na adoção de metodologias inovadoras e diversos outros recursos de aprendizagem que se adequem à sociedade do conhecimento. “Professores e alunos em profunda aliança precisam aprender não só como ter acesso à informação, mas, principalmente, como desenvolver espírito crítico com vistas à produção de conhecimento.” (BEHRENS, 2000, p.95).

Além da antiga preocupação de se formar um profissional competente, torna-se necessário, na atual conjuntura, formar um cidadão autônomo e criativo que seja capaz de solucionar problemas e também de questionar-se, constantemente, pela velocidade das informações, sendo capaz de transformar a sociedade.

Dentro dessa nova realidade, o trabalho com projetos surge como uma possibilidade de não só trabalhar com conteúdos de forma significativa, como também de desenvolver habilidades e competências nos alunos que serão necessárias à sua vida profissional. Além disso, desenvolve-se também a autonomia para aprender, de forma criativa e inovadora.

Assim,

> um projeto pode organizar-se seguindo um determinado eixo: a definição de um conceito, um problema geral ou particular, um conjunto de perguntas interrelacionadas, uma temática que valha a pena ser tratada em si mesma... Normalmente, superam-se os limites de uma matéria. Para abordar esse eixo em sala

> de aula, se procede dando ênfase na articulação da informação necessária para tratar o problema objeto de estudo e nos procedimentos requeridos pelos alunos para desenvolvê-lo, ordená-lo, compreendê-lo e assimilá-lo (HERNÁNDEZ, VENTURA, 1998, p.61).

Como se vê, ao se optar por se trabalhar com projetos na construção do conhecimento escolar, valoriza-se uma prática pedagógica que estimula dos alunos "através da pesquisa, desenvolve o respeito às diferenças pela necessidade do trabalho em equipe, incentiva o saber ouvir e expressar-se, o falar em público e o pensamento crítico autônomo." (OLIVEIRA, 2006, p. 14). Esse tipo de trabalho promove a autonomia intelectual, "uma vez que, que vai sendo conquistada através da pesquisa, com toda a diversidade de caminhos percorridos e as competências que os alunos vão desenvolvendo através de tal prática" (OLIVEIRA, 2006, p. 14).

Ressalta-se que, para haver aprendizagem, é necessário que o tema e o produto a ser desenvolvido no projeto seja significativo aos alunos e professores. "Os projetos abrem para a possibilidade de aprender os diferentes conhecimentos construídos na história da humanidade de modo relacional e não-linear, propiciando às crianças aprender através de múltiplas linguagens, ao mesmo tempo em que lhes proporcionam a reconstrução do que já foi aprendido" (BARBOSA, HORN, 2008, p.35).

Atentando para todos esses benefícios trazidos pela metodologia de projetos, o curso de Análise e Desenvolvimento de Sistemas de um Centro Universitário da Zona da Mata Mineira instituiu um currículo totalmente baseado no desenvolvimento de habilidades e competências dos alunos, tendo como base principal de avaliação o desenvolvimento de projetos interdisciplinares.

O artigo em questão objetiva, justamente, apresentar o processo de criação de um desses projetos, criados para conclusão de um dos semestres letivos do curso. Trata de um sistema de documentação do Ticket, um *software* de gerenciamento de chamados feito para o auxílio no envio e recebimento de chamados em uma organização, prestando controle, gerência e métricas sobre o suporte e atendimento de clientes. Durante o artigo, é relatada a importância de um software de atendimento de chamados, sendo relatadas diversas das etapas da engenharia de software como o levantamento de requisitos, modelagem de dados e diagramação do sistema, além de apontar as tecnologias que serão utilizadas no processo de desenvolvimento do sistema. Por meio do desenvolvimento do software, o aluno conseguiu demonstrar todo o aprendizado desenvolvido no decorrer da disciplina.

2 REFERENCIAL TEÓRICO

O avanço tecnológico, assim como a constante divulgação das informações, colocou o cenário corporativo a enfrentar situações atípicas decorrentes de todos os recursos funcionais envolvidos. Nesse novo cenário, o cliente encontra-se hoje muito mais exigente do que nunca, tendo a sua disposição uma miríade de opções; dessa maneira, hoje a relação com o cliente não pode mais se basear apenas na oferta de um serviço ou produto, ou, nas palavras de Bentes (2012, p.19), "ter qualidade não é mais diferencial competitivo, já que suas práticas são domínio de ampla parte do mercado". Assim, no panorama competitivo, o relacionamento com o cliente é tão importante quanto a oferta de um bom produto, não sendo apenas um dos grandes diferenciais que uma organização pode ter em meio às demais opções de escolha de um cliente, mas sim uma necessidade mercadológica.

> Lembre-se de que um bom serviço de atendimento ao cliente não é uma opção; nos mercados dinâmicos atuais, é uma obrigação procurar apresentar um desempenho superior ao da concorrência (WELLINGTON, 2011, p.11).

Assim, para o autor, no mundo atual, um bom relacionamento com o cliente faz parte de um processo de gerenciamento de todos os recursos funcionais envolvidos. Nesse sentido, a estrutura exigida da organização deve ressaltar não apenas os processos precedentes e decorrentes do processo de compra, mas também o atendimento ao cliente durante um período indeterminado de tempo. O cuidado e o acompanhamento do cliente fornecem uma experiência de consumo que o fideliza e o transforma em um divulgador da organização.

2.1 A importância dos *softwares* de suporte técnico

Atualmente, a ampla utilização de tecnologias no domínio organizacional já se tornou prática regular do mercado; em função disso, ferramentas de *Enterprise Resource Planning* (ERP) e *Customer Relationship Management* (CRM) se demonstram indispensáveis para qualquer corporação.

> A eficiência, eficácia, a efetividade e a economicidade no suporte aos serviços de TI e aos seus usuários são fatores críticos para o sucesso no alcance dos objetivos estratégicos traçados pela organização (MAGALHÃES, 2007, p.109)

É neste aspecto que, segundo o autor, ferramentas de gestão de chamados se encontram, sendo de vital importância na construção de um suporte eficaz e responsivo, no qual sua implementação permite a centralização da comunicação entre suporte, usuário e cliente, diminuindo os atrasos na troca de informações.

2.2 A implementação dos *softwares* de suporte técnico

Um *software* de gerência de chamados deve, de acordo com Magalhães (2007, p. 117), "atuar como ponto único de contato (*Single Point of Contact*) entre os usuários e os clientes dos serviços de TI e os diversos processos relacionados com o gerenciamento dos serviços de TI.", ou seja, o ponto principal deste tipo de serviço é a centralização da informação, evitando redundância e repetição na troca de informações, ao mesmo tempo entregando ferramentas de suporte comunicacional.

> Destaca-se, ainda, como objetivo, garantir o encerramento do maior número de incidentes e consultas dentro do seu nível de atendimento no processo de gerenciamento de incidentes, evitando a sobrecarga das demais equipes que atuam no processo, bem como realizar a interface entre usuários e clientes (MAGALHÃES, 2007, p.112).

2.3 Tecnologias utilizadas no desenvolvimento do *software*

As seguintes escolhas de tecnologias foram utilizadas para o desenvolvimento do Ticklet*:*

2.3.1 Javascript

A linguagem escolhida para ser utilizada no desenvolvimento do Ticklet é o Javascript, a linguagem padrão utilizada pelos navegadores *Web* e com uma ampla gama de bibliotecas e *frameworks* que serão utilizados para auxiliar diversas etapas do desenvolvimento. O *Javascript* foi escolhido por seu foco em desenvolvimento *web*, ampla documentação em português e seus diversos complementos e módulos disponíveis.

> JavaScript (às vezes abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O

> JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (MOZILLA, 2021, on-line).

2.3.2 NodeJs

O *Javascript*, em si, é uma linguagem voltada a ser utilizada unicamente por navegadores *web*, logo, para que a utilização da linguagem possa ser feita em servidores para o tratamento dos dados do banco de dados e renderização de páginas, é necessário a utilização do *NodeJs*, que é, de acordo com a OpenJS Foundation, "As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications "[1].(OPENJSFOUNDATION, 2021, on-line), ou seja, com o uso do NodeJS, é possível utilizar o *Javascript* em servidores para o desenvolvimento de qualquer *software*.

> Node (ou formalmente Node.js) é um ambiente em tempo de execução open-source (código aberto) e multiplataforma que permite aos desenvolvedores criarem todo tipo de aplicativos e ferramentas do lado servidor (backend) em JavaScript (MOZILLA, 2021, on-line).

2.3.3 *NextJS*

Juntamente com a utilização do *Javascript* como linguagem, também será utilizada o *NextJs*, um *framework* para *React* que, por sua vez, é uma biblioteca voltada a construção de páginas a partir do *Javascript*. O *NextJs* é um *framework* desenvolvido para ser utilizado através do *ReactJs* e que introduz novos conceitos e tecnologias ao *ReactJs* e ao *Javascript* (MICROSOFT, 2021, on-line). Com a utilização do *NextJ,s* é possível aproveitar-se de renderização do lado do servidor, pré-renderização híbrida, roteamento de sistema de arquivos e diversos novos incrementos.

> Next.js é uma estrutura para criar aplicativos JavaScript renderizados pelo servidor com base em React.js, Node.js, Webpack e Babel.js. Ele é basicamente um projeto genérico para o React, criado de acordo com as práticas recomendadas, que permite criar aplicativos Web "universais" de forma simples e consistente, praticamente com qualquer configuração. (MICROSOFT, 2021, on-line).

2.3.4 *Tailwindcss*

[1] Com um tempo de execução JavaScript baseado em eventos assíncronos, o Node.js foi projetado para construir aplicativos de rede escalonáveis (OPENJSFOUNDATION, 2021, on-line, tradução nossa).

Para a estilização das páginas, será utilizado o *Tailwindcss*, um *framework* para o *Cascading Style Sheets* (CSS), que utiliza a abordagem de *utility-first* na estilização das páginas, substituindo o método padrão de construção de classes pela utilização de classes providas pelo próprio *framework* tornando o processo de implementação do *design* mais dinâmico. Ou nas palavras da equipe do *Tailwindcss:*

> Utility classes help you work within the constraints of a system instead of littering your stylesheets with arbitrary values. They make it easy to be consistent with color choices, spacing, typography, shadows, and everything else that makes up a well-engineered design system (TAILWINDCSSS, 2021, on-line)[2].

2.3.5 PostgreSQL

Para o banco de dados, foi escolhido o *PostgreSQL*, um sistema gerenciador de banco de dados relacional, feito com código aberto e completamente gratuito. A escolha do *PostgreSQL* se dá pelo seu amplo histórico funcional de utilização em sistemas de grande porte, além de ser altamente documentado e de comunidade ativa no mundo de *software* livre.

> O PostgreSQL vem com muitos recursos destinados a ajudar os desenvolvedores a criar aplicativos, administradores para proteger a integridade de dados e construir ambientes tolerantes a falhas e ajudá-lo a gerenciar seus dados, não importa quão grande ou pequeno seja o conjunto de dados. Além de ser livre e de código aberto,o PostgreSQL é altamente extensível (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2007, on-line).

2.3.6 Prisma

Outra tecnologia fundamental escolhida para ser utilizada durante o desenvolvimento e com relação ao banco de dados é o Prisma, um mapeador relacional de objetivos baseado em *NodeJs*, que será responsável por intermediar o caminho entre o banco de dados e a aplicação *web*. “Easy to integrate into your framework of choice, Prisma simplifies database access, saves repetitive CRUD boilerplate and increases type safety” (PRISMA, 2021, on-line)[3].

[2] As classes utilitárias ajudam você a trabalhar dentro das restrições de um sistema, em vez de encher suas folhas de estilo com valores arbitrários. Eles tornam mais fácil ser consistente com as escolhas de cores, espaçamento, tipografia, sombras e tudo o mais que compõe um sistema de design bem projetado. (TAILWINDCSS, 2021. tradução nossa)

[3]Fácil de integrar na estrutura de sua escolha, o Prisma simplifica o acesso ao banco de dados, salva CRUD repetitivos e aumenta a segurança de tipo. (PRISMA, 2021. tradução nossa)

3 METODOLOGIA

Trata-se de uma pesquisa de caráter descritivo, uma vez que objetiva apresentar um projeto de um software como avaliação final de um semestre letivo no curso de Análise e Desenvolvimento de Sistemas. O projeto foi desenvolvido com base em pesquisa exploratória e fundamentado a partir de levantamento bibliográfico, baseado primordialmente nos levantamentos de Pinheiro e Magalhães (2007) e Welligton (2011) sobre o funcionamento e importância dos sistemas de suporte e nas ideias de Pressman (2016) sobre todas as etapas da construção de software, desde o levantamento de requisitos até a entrega da aplicação final.

4 REQUISITOS E MODELAGEM DO SISTEMA

4.1 Levantamento dos requisitos do sistema

Seguindo a concepção de modelagem e planejamento proposta por Sommerville (2016, on-line), "todos os aspectos do *software* a ser construído deve ser especificado antes de o projeto começar", ou seja, antes de qualquer construção ou planejamento de *software* ser feita, a primeira tarefa a ser executada deve ser o levantamento e a análise dos requisitos do sistema, sendo esses requisitos as funções e as expectativas de funcionamento sobre o sistema a ser desenvolvido.

E, com base em pesquisas on-line, observações sobre outros sistemas e conversas com profissionais de TI foram levantados os demais requisitos funcionais e esperados para o posterior planejamento e desenvolvimento do Ticklet*:*

- O sistema deve ter um cadastro de usuário on-line;
- O sistema deve suportar a criação de múltiplos setores organizacionais;
- O usuário deve ser capaz de criar *tickets*;
- O usuário deve ser capaz de responder *tickets*;
- Os *tickets* devem ter alguma atribuição de *status* entre pendente e completo;
- O usuário de tipo suporte deve ser capaz de redefinir o *status* de qualquer *ticket* a qualquer momento;
- O sistema deve ter suporte a respostas prontas para o suporte;
- O suporte pode criar categorias para organizar os chamados.

Além dos requisitos funcionais, também foram especificados os requisitos não funcionais, os requisitos que dizem respeito a especificações de detalhes importantes, mas não essenciais ao funcionamento do sistema.

- Pesquisar por chamados;
- Alta gama de filtros para chamados;
- Formatação dos textos do chamado;
- Criar e exportar relatórios.

4.2 Modelagem de *Software*

A partir da análise dos requisitos levantados, é possível estipular as funções necessárias para que o *software* atenda às necessidades de seus usuários e modelar tais processos de forma unificada e consistente através da utilização da *UML*.

> The OMG's Unified Modeling Language™ (UML®) helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. (You can use UML for business modeling and modeling of other non-software systems too.) Using any one of the large number of UML-based tools on the market, you can analyze your future application's requirements and design a solution that meets them, representing the results using UML 2.0's thirteen standard diagram types.(OMG, 2005, on-line)[4]

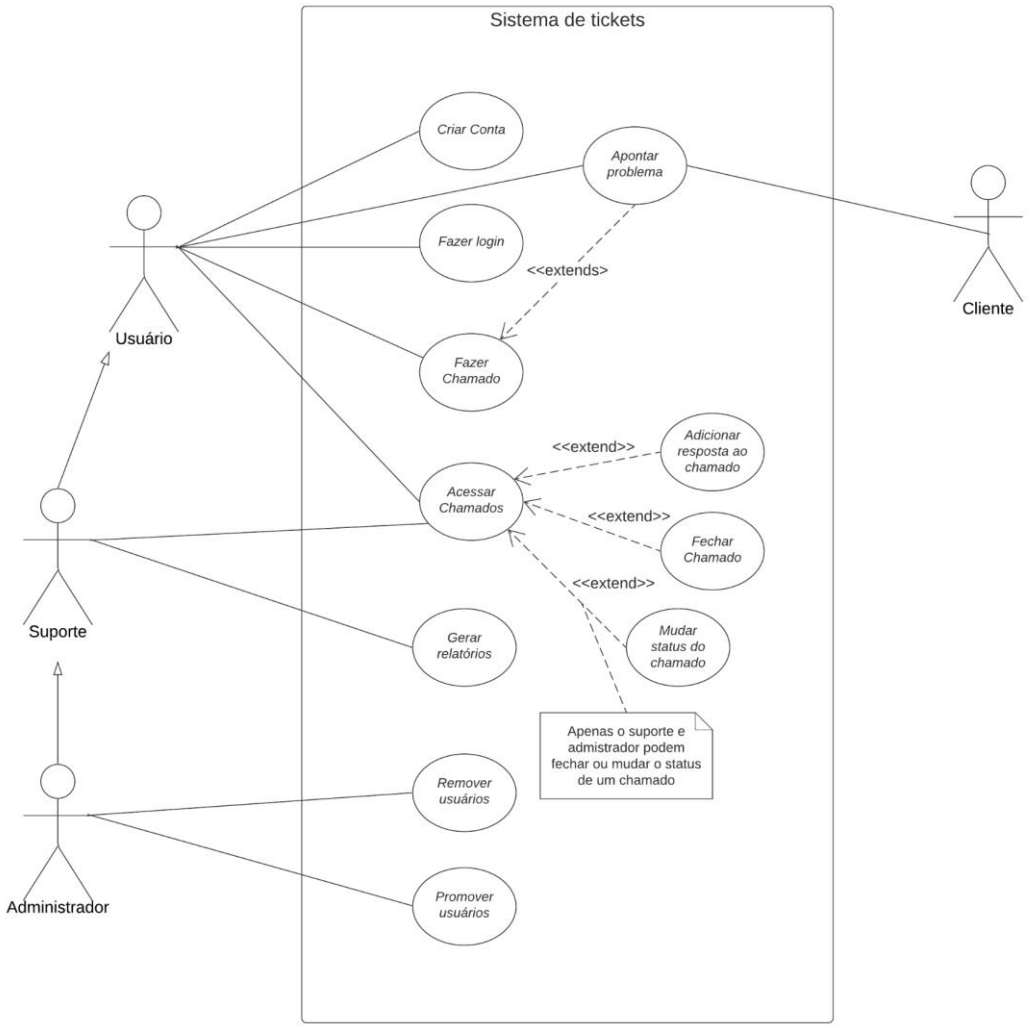
4.3 Diagrama de casos de uso

O diagrama de casos de uso da figura 1 descreve as funcionalidades e as relações do sistema no mais alto nível, abstraindo todos os detalhes de implementação e exemplificando o uso do sistema a partir de atores e casos de uso. A partir desse diagrama, é possível visualizar

[4] A UML da OMG ajuda a especificar, visualizar e documentar modelos de sistemas de software, incluindo sua estrutura e design, de uma forma que atenda a todos esses requisitos. (Você também pode usar UML para modelagem de negócios e modelagem de outros sistemas que não sejam de software.) Usando qualquer uma das muitas ferramentas baseadas em UML do mercado, você pode analisar os requisitos de seu aplicativo futuro e projetar uma solução que os atenda, representando os resultados usando os treze tipos de diagrama padrão da UML 2.0.(OMG, 2005, tradução nossa)

o escopo do projeto, as entidades envolvidas e funcionalidades do sistema. É possível ter uma descrição mais clara através do apêndice A.

FIGURA 1: Diagrama de casos de uso

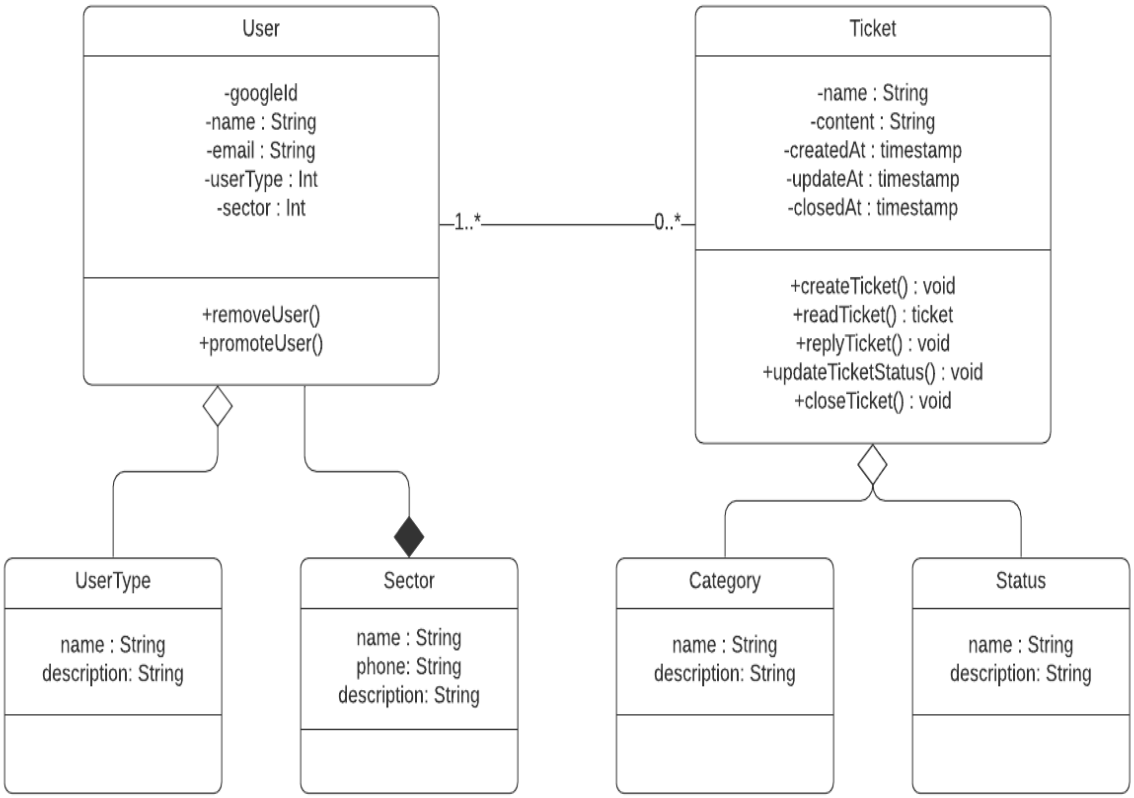


Fonte: Acervo pessoal

4.4 Diagrama de classes

A proposta deste diagrama de classes representado na figura 2 é apresentar as relações entre os artefatos do sistema a partir de uma estrutura relacional de classes de objetos. As principais classes do sistema são os *tickets*, ou também denominados chamados e o usuário, além dessas, há alguns objetos complementares para o funcionamento pleno do sistema.

FIGURA 2: Diagrama de classes

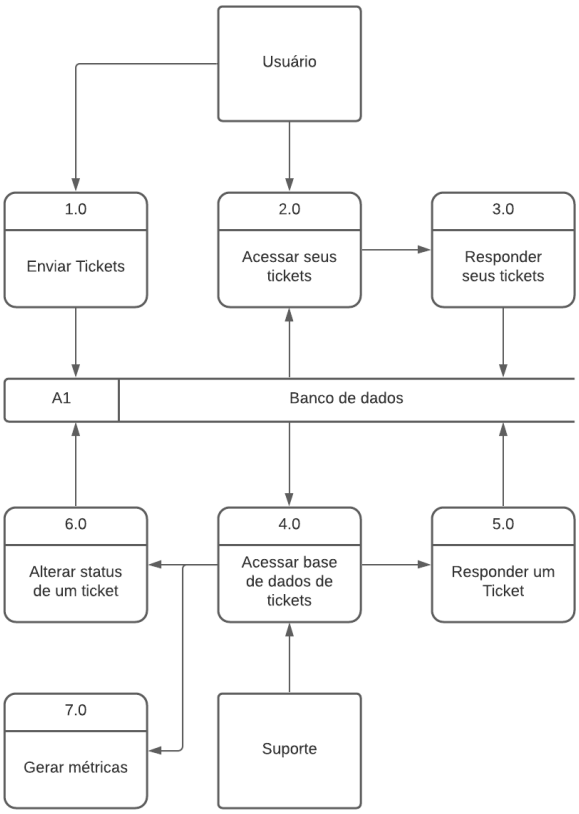


Fonte: Acervo pessoal.

4.5 Diagrama de Fluxo de dados

Com os casos de uso e classes do sistema bem definidos, a próxima etapa é estruturar o fluxo de dados e operações que se sucedem na execução do sistema. A figura 3 representa o fluxo de operações que podem ser realizadas pelo usuário e pelo suporte no envio, recebimento e resposta dos chamados.

FIGURA 3: Fluxo de dados de um chamado



Fonte: Acervo pessoal

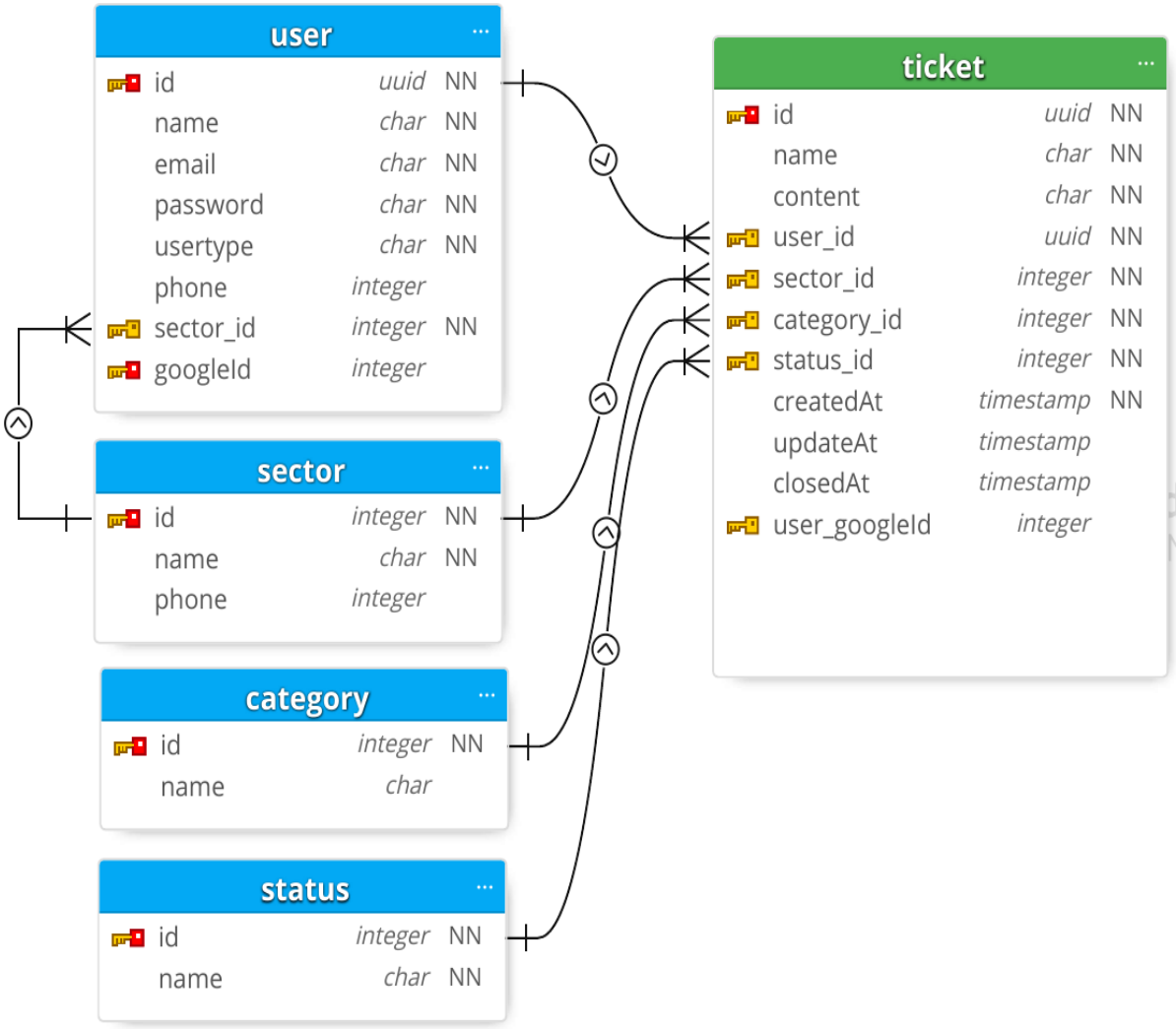
4.6 Modelagem de dados

Tendo em mente que o Ticklet será um *software* responsável por criar, enviar e consultar informações relativas a um chamado de suporte, a modelagem de dados se deu com o objetivo de elucidar quais os dados necessários para o funcionamento dessas operações e quais seriam as relações e dependências entre esses dados.

O chamado, ou *ticket*, como é intitulado internamente pelo sistema, é o principal artefato presente em torno do sistema, então é a partir dele que é planejada toda a modelagem e o fluxo de dados. No sistema, o chamado carrega consigo as informações de seu remetente, setor de envio, data de envio, título, conteúdo da mensagem, informação de status, data da última vez que foi modificado e data da conclusão. Caso o chamado tenha sido atendido e concluído, alguns desses dados como datas de atualização e conclusão podem parecer irrelevantes inicialmente para a gerência dos chamados, mas serão úteis para criar métricas.

Além do *ticket*, o sistema armazena informações básicas sobre outros artefatos como o próprio usuário e seus dados como nome, e-mail, telefone e setor, artefatos intermediários que servem para prover informações complementares para os usuários e chamados. A figura 4 é uma representação visual das tabelas e relacionamentos que existiram no banco de dados.

FIGURA 4: Tabelas do banco de dados



Fonte: Acervo pessoal

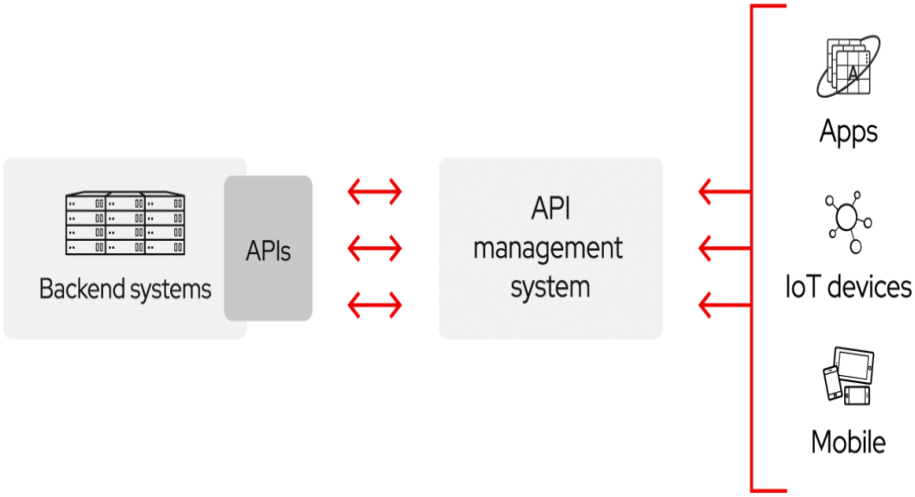
4.7 *Endpoints* da API

Uma *Application Interface Protocol* (API) é acordo com a REDHAT:

> An API is a set of definitions and protocols for building and integrating application software. API stands for application programming interface. APIs let your product or service communicate with other products and services without having to know how they're implemented. This can simplify app development, saving time and money. When you're designing new tools and products—or managing existing ones—APIs give you flexibility; simplify design, administration, and use; and provide opportunities for innovation. (REDHAT, 2021, on-line)[5]

Para o desenvolvimento do Ticklet, será criado uma API que proverá acesso para leitura e alteração de dados no banco de dados. A API utiliza o protocolo de requisições *Hypertext Transfer Protocol* (HTTP).

FIGURA 5: Funcionamento de uma API



Fonte: Redhat.

A figura 5 representa, de modo geral, o funcionamento de uma *API* e, na figura 6, são especificadas as rotas e os métodos que serão utilizados para prover acesso entre a aplicação e o banco de dados.

[5] Uma API é um conjunto de definições e protocolos para construir e integrar software de aplicativo. API significa interface de programação de aplicativo. As APIs permitem que seu produto ou serviço se comunique com outros produtos e serviços sem precisar saber como eles são implementados. Isso pode simplificar o desenvolvimento de aplicativos, economizando tempo e dinheiro. Quando você está projetando novas ferramentas e produtos - ou gerenciando os existentes - as APIs oferecem flexibilidade; simplificar design, administração e uso; e oferecer oportunidades de inovação.(REDHAT,2021, on-line, tradução nossa)

FIGURA 6: Rotas da API

Fonte: Acervo Pessoal

É também importante especificar que todas rotas do tipo *GET*, *PATCH* e *DELETE* possuem métodos para filtrar os dados que serão recebidos, alterados e deletados.

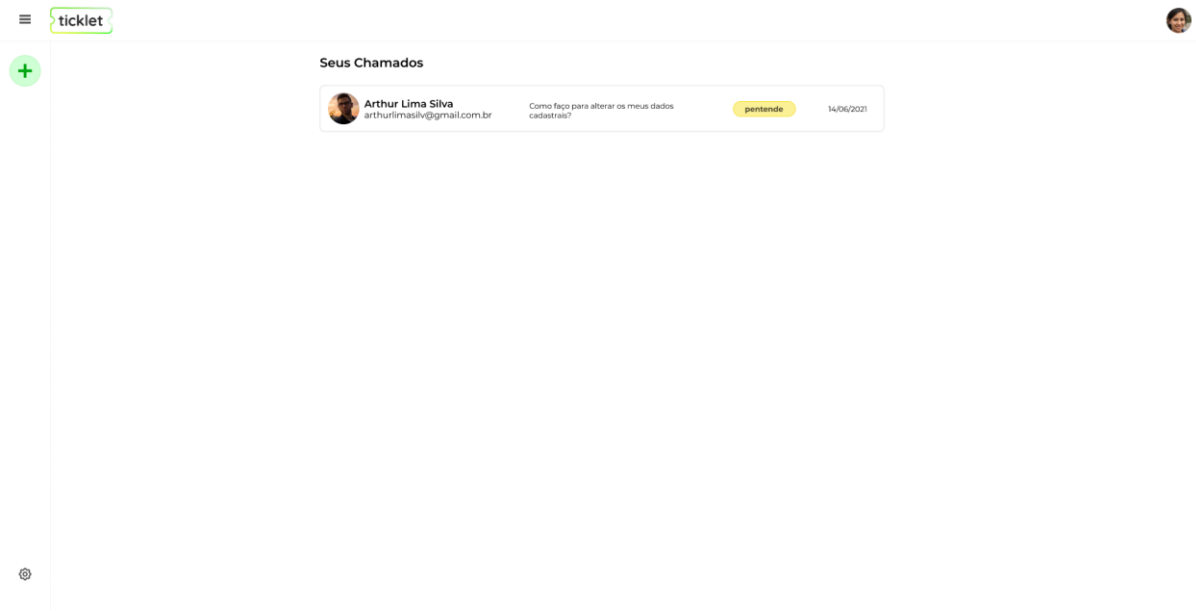
4.8 Interface gráfica

Ao propor a utilização de um novo *software* dentro de qualquer organização, por via de regra, sempre surgirá o atrito e a resistência na adoção da nova ferramenta por parte do usuário. Para amenizar tal resistência, a interface de usuário do Ticklet foi projetada para parecer familiar e de simples adaptação para o usuário médio, utilizando-se de um *design* que se assemelha a um leitor de e-mails e seguindo diversas das heurísticas propostas por Nielsen.

Como a proposta de utilização do Ticklet é feita para dois tipos de usuários, a interface se altera de acordo com o usuário em questão, apresentando opções e menus diferentes para o cliente e para o suporte, mas sempre mantendo a integridade visual e conceitual.

Este é o menu principal do sistema mostrado na figura 7 a seguir, assemelhando-se a um cliente de e-mail tradicional, ele agrupa todos os chamados em uma lista vertical no centro da tela; no canto esquerdo, tem-se um menu lateral com acesso rápido a opções disponíveis ao usuário e, no topo, tem-se, além de um menu hamburger[6] para a expansão das opções disponíveis na esquerda, tem-se a logo do sistema e, por fim, no canto superior direito, é visto um círculo com a foto de perfil do usuário.

Figura 7: Tela principal do usuário

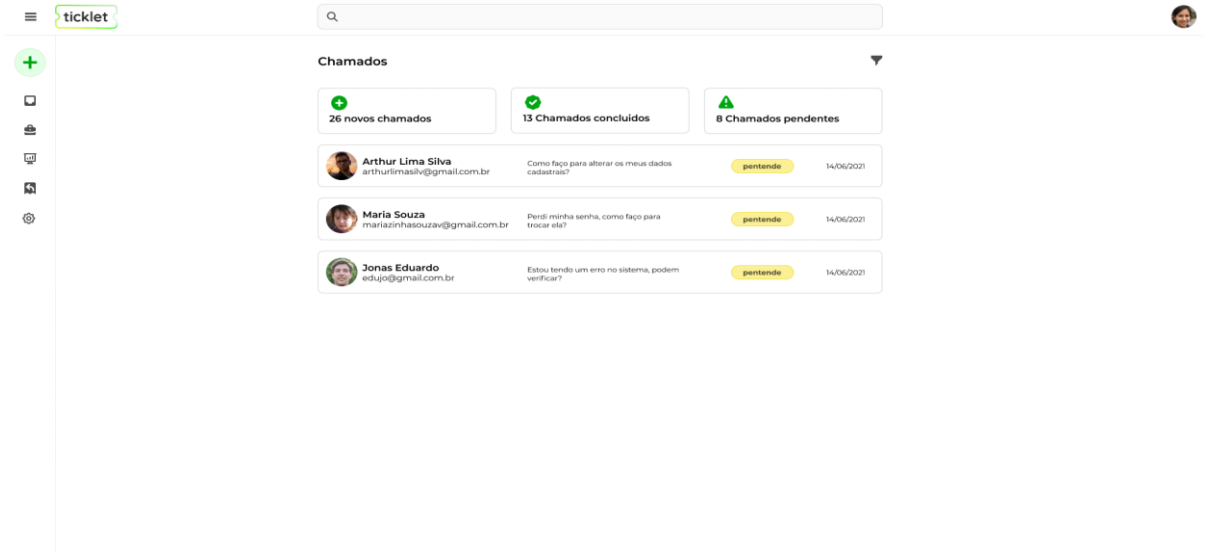


Fonte: Acervo pessoal

[6] Ícone com três linhas horizontais, comumente chamado de menu hambúrguer.

A interface do menu principal se mantém a mesma para o suporte na figura 8, apenas apresentando algumas opções extras na esquerda e funções extras para filtragem de chamados, além de um menu de pesquisa na parte superior da tela.

Figura 8: Tela principal do suporte



Fonte: Acervo pessoal.

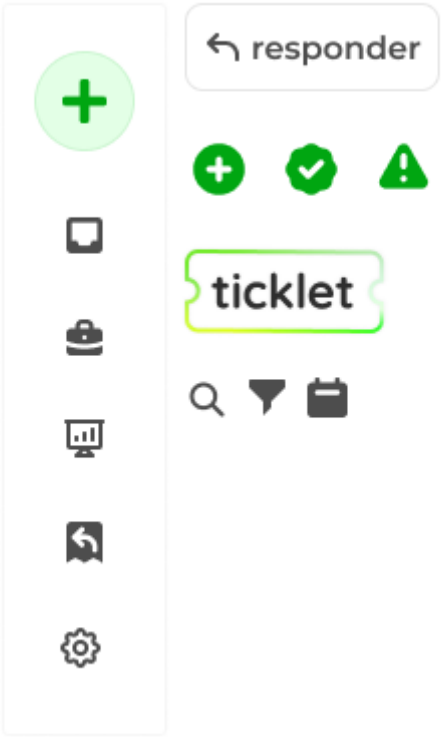
Já a tela de visualização de um chamado se assemelha a uma aplicação de fórum *on-line*, no qual o chamado se apresenta como um tópico com descrição e, logo abaixo, o suporte e cliente podem trocar informações para a resolução do problema descrito. A figura 9 mostra a tela de leitura de chamados.

Figura 9: Tela de leitura de chamados

Fonte: Acervo pessoal.

Um cuidado feito ao longo de toda a interface é constantemente utilizar ícones no lugar de texto para botões e opções do sistema, representados na figura 10, isto gera, no usuário, uma correspondência entre as funcionalidades e opções do *software* com representações físicas do mundo real, e isto, consequentemente, traz ao usuário facilidade no uso já que o faz ser mais dependente do reconhecimento das opções e menos da memorização das partes do sistema.

FIGURA 10: Iconografia do Ticklet



Fonte: Acervo pessoal

A seguir, são apresentados os diagramas de caso de uso, descritos através na tabela abaixo, neles encontram-se todas as ações tomadas dentro do sistema a partir de seus utilizadores. Os casos de uso são uma forma clara de especificar os recursos que deverão ser implementados para o bom funcionamento do software.

TABELA 1: Diagrama de Casos de uso

Número de ordem	Caso de uso	Descrição
1	Criar conta	O usuário poderá criar uma conta através de autenticação com o google
2	Fazer login	O usuário poderá acessar sua conta através de autenticação com o google
3	Fazer chamado	O usuário poderá criar e enviar chamados contento dúvidas ou problemas para uma equipe de chamados resolver
4	Acessar chamado	O usuário poderá acessar seus chamados para poder acompanhar a resolução dos seus problemas e o suporte poderá acessar qualquer chamado a fim de resolver os problemas enviados
5	Responder chamado	O usuário poderá adicionar respostas ao chamado em uma estrutura semelhante a um bate papo entre ele e o suporte. O mesmo vale para o suporte
6	Fechar chamado	O suporte poderá dar um chamado com concluído e encerrar o atendimento.
7	Alterar o status do chamado	O suporte poderá alterar o status do chamado a fim de especificar em que ordem está a resolução do chamado
8	Gerar métricas	O suporte poderá gerar métricas a partir dos dados sobre os chamados
9	Remover usuário	O administrador do sistema poderá remover usuários indesejados
10	Promover usuário	O administrador do sistema poderá promover ou rebaixar qualquer usuário a suporte ou usuário.

Número de ordem	Ator	Definição
1	Usuário	O usuário tem permissões para criar, abrir e responder seus chamados
2	Suporte	O suporte além das permissões do usuário também é capaz de ter acesso a qualquer chamado, encerrar chamados, alterar status dos chamados, gerar métricas
3	Administrador	O administrador além de ter todas as outras permissões também é capaz de remover, promover e rebaixar usuários
4		

Fonte: Acervo pessoal

5 CONSIDERAÇÕES FINAIS

Através da documentação elaborada, é possível elucidar todos os processos necessários para o desenvolvimento do Ticket e ter uma visão clara dos recursos e procedimentos exigidos para a conclusão do projeto, assim essa documentação se prova útil como ferramenta de planejamento para o futuro projeto do sistema de chamados que está em fase de desenvolvimento.

Observou-se, durante todo o semestre, que a utilização da metodologia de projetos como estratégia de ensino e como forma de avaliação dos alunos favorece a aprendizagem, uma vez que contextualiza os conteúdos fundamentais trabalhados no decorrer do semestre, bem como desenvolve a autonomia dos alunos, contribuindo para uma formação crítica e reflexiva.

5 REFERÊNCIAS

ABOUT NODEJS. **OpenJS Foundation**. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 29 jun.2021.

BARBOSA, Maria C. S.; HORN, Maria da Graça S. **Projetos pedagógicos na Educação Infantil.** Porto Alegre: Artmed, 2008.

BEHRENS, M. A. Projetos de aprendizagem colaborativa num paradigma emergente. In: MORAN, J. M. MASETTO, M. T; BEHRENS, M A. **Novas tecnologias e mediação pedagógica**. Campinas: Papirus, 2000.

HERNÁNDEZ, Fernando; VENTURA, Montserrat. **A organização do currículo por projetos de trabalho**: o conhecimento é um caleidoscópio. 5. ed. Porto Alegre: Artmed, 1998.

MICROSOFT. Introdução ao Next.js no Windows. **Microsoft**. Disponível em: <https://docs.microsoft.com/pt-br/windows/dev-environment/javascript/nextjs-on-wsl>. Acesso em: 29 jun. 2021.

MOZILLA. Introdução Express/Node. **MOZILLA,** 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introduction>. Acesso em: 11 jul. 2021.

MOZILLA. Javascript. **MOZILLA,** 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 29 jun. 2021.

OLIVEIRA, Cacilda Lages. **Significados e contribuições da afetividade, no contexto da Metodologia de Projetos, na Educação Básica**. 2006. Dissertação (Mestrado) – CEFET – MG, Belo Horizonte MG, 2006. Disponível em: OLIVEIRA, Cacilda Lages - Significado e contribuições da afetividade, no contexto da Metodologia de Projetos, na Educação Básica (tecnologiadeprojetos.com.br) Acesso em: 20 ago. 2021.

PINHEIRO, Walfrido Brito; MAGALHÃES, Ivan Luizio. **Gerenciamento de serviços de TI na prática:** uma abordagem com base na ITIL. São Paulo: Novatec, 2007.

PRESSMAN, Roger. **Engenharia de Software**. 8. ed. São Paulo: McGraw Hill Brasil, 2016.

PRISMA. Next-generation Node.js and TypeScript ORM. **PRISMA**, 2021. Disponível em: <https://www.prisma.io/>. Acesso em: 21 jun. 2021.

RED HAT. What is an API? . **RED HAT**. Disponível em: <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>. Acesso em: 21 jun. 201

UML. Introduction To OMG's Unified Modeling Language™ (UML®). **UML**, 2005. Disponível em: <https://www.uml.org/what-is-uml.htm>. Acesso em: 21 jun. 2021.

TAILWINDCSS. Rapidly build modern websites without ever leaving your HTML.. **Tailwindcss,** 2021. Disponível em: <https://tailwindcss.com >. Acesso em: 11 jul. 2021

WELLINGTON, Pat. **Atendimento Eficaz ao Cliente**. São Paulo: Clio Editora, 2011.