KELVYNN JOSÉ DA SILVA

MICHEL BEZERRA

SOLANGE DE ALMEIDA SILVA

TIAGO GRANATTA

SISTEMA DE GESTÃO DE CHAMADOS DE SUPORTE E MANUTENÇÃO PARA PEQUENAS E MÉDIAS EMPRESAS

Relatório Técnico apresentado como requisito parcial/final ao Projeto Integrador do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Cascavel, Paraná.
2021

SUMÁRIO

1. INTRODUÇÃO

Apresenta-se nessa proposta uma ferramenta que proporcione e auxilie aqueles cuja a necessidade seja de pedir apoio para resolução de algum problema, ou realizar alguma solicitação de acesso, sendo possível detalhar e organizar conforme sua prioridade e facilitar à gestão de recursos de solução em suporte pelos responsáveis pelo suporte em pequenas e médias empresas (com equipe técnica de manutenção e/ou TI internas), podendo facilitar o processo tanto para aqueles que irão realizar os chamados (conhecidos como *tickets* em certos sistemas), quanto os atendentes envolvidos.

Com o principal objetivo de facilitar o atendimento de chamados, como permitir a visualização e compreensão do que acontece dentro dos setores relacionados, esse software almeja englobar a realização de chamados pelo usuário, acompanhamento de chamados realizados do próprio usuário, de maneira a identificar os problemas a serem resolvidos, de maneira esclarecedora, e que possibilite o preparo e a tomada de ação por algum agente de manutenção/suporte técnico; Prestando informações relacionadas ao próprio chamado, onde a ação é necessária, e possibilite a inserção de imagens e documentos anexados a determinados chamados além da definição e alteração de Status dos chamados, e nível prioritário, de modo a permitir o acompanhamento do solicitante, e permitir o controle de prioridades dos atendentes, que poderão exclusivamente ter acesso à visualização e controle de prioridade do chamado.

Ao longo da construção do software, alguns itens poderão ou não ser implementados, em vista da complexidade e da limitação de algumas ferramentas utilizadas; uma dessas possíveis adições é a troca de mensagens entre atendentes e usuários comuns, de modo a possibilitar um canal de comunicação e facilitar as interações relacionadas com um chamado, poderá ser implementado em uma futura versão.

O Sistema há de possibilitar sua própria implementação sem grandes dificuldades, permitindo a adequação com ecossistemas e ambientes de utilização variados, podendo atender diferentes segmentos de empresa, com seus setores, divisões e seções e suas particularidades sem maiores dificuldades.

2. FUNDAMENTAÇÃO TEÓRICA

Quando são discutidos os processos de controle de suporte e manutenção em pequenas e médias empresas, observa-se a necessidade de um sistema de gestão de solicitações e suporte com problemas técnicos internos; em algumas situações, onde por exemplo não existe uma ferramenta específica instaurada para esse fim e devidamente implementada, são perceptíveis transtornos como a falta de clareza da definição do problema e a falta de informação sobre determinada solicitação, o que gera inconsistências nos atendimentos e consequentemente na qualidade do serviço prestado, além do aumento no *downtime*[1] das atividades da organização.

Cohen comenta sobre a utilização de subprodutos da tecnologia para a otimização do ambiente de suporte técnico-informacional dos dias atuais, os quais vem se tornando cada vez mais exigentes e imediatistas.

> "O crescimento acelerado do uso da tecnologia, nos últimos tempos, para aumento de eficiência e competitividade arrastou tais departamentos para um crescimento substancial nos atendimentos de suporte." (Roberto Cohen, 2008:15)

É comum a utilização de sistemas de gerenciamento de chamados em empresas de grande porte e conglomerados de terceirização de serviços de atendimento ao consumidor, com sistemas complexos de atendimento, que podem possuir canais de comunicação com centenas ou até milhares de consumidores, chamados de *CRM*, os gerenciadores de relação com o consumidor. Já por outro lado, em caráter de atendimento interno, empresas de menor porte tendem a utilizar e-mails corporativos para controle e gerenciamento de chamados de suporte ou manutenção, que podem se tornar confusos ou até difíceis de controlar e gerenciar.

Como o volume de atendimentos internos nessas empresas é maior, a utilização de sistemas *helpdesk* é atraente, mas os sistemas que empresas de atendimento ao público/cliente não são a melhor opção, devido aos preços e a extrema escala de utilização que são comuns com esses programas.

---

[1] *Downtime:* Período no qual a produção, ou máquinas não estão trabalhando devido à falhas ou impossibilidade de uso.

3. MATERIAIS E MÉTODOS

As ferramentas utilizadas, foram escolhidas com rápido desenvolvimento em mente, possibilitando a implementação de maneira dinâmica, juntamente com uma visão que traz em base em alguns conceitos diferentes trazidos das metodologias *Kaizen* e *PDCA*, de outras áreas de atuação, como, a engenharia de produção e gestão de eventos e processos. Essas ferramentas permitem um processo de definição, planejamento, execução e conferência ao longo de todo o processo de desenvolvimento, de modo dinâmico, não serão utilizados em sua totalidade, devido ao escopo de produção que geralmente é voltado para a área industrial, e de produção em larga escala, o que difere da documentação, definição e engenharia de software, nos quais essas ferramentas são utilizadas de maneira mais acatada e limitada, como por exemplo na definição de um problema criado pela necessidade de certo requisito de sistema pretendido, em alguns passos do modelo *PDCA*, aliado à filosofia *Kaizen*, o altruísmo do conceito *trial and error*[2] e processo de eliminação para a resolução de certos problemas na implementação podem ser mais indicados para sistemas que não possuam caminhos de construção definidos, porém, neste caso, utilizados com certos cuidados para não produzir o retrabalho.

Uma ferramenta completa e eficiente o método do ciclo PDCA é um dos mais conhecidos para ajudar na execução do planejamento estratégico nas empresas. Antes de se construir um ciclo PDCA precisa-se que a estratégia seja previamente desenvolvida, para que assim os objetivos se desenvolvam fora do papel.

O ciclo PDCA também é conhecido como Shewhart ou ciclo Deming, o engenheiro Walter Shewhart foi o criador do método nos Estados Unidos na década de 1920, onde era composto por três passos repetidos continuamente (especificação, produção e inspeção). Mas só mais tarde em 1951 esse método ficou mais conhecido através de William Edwards Deming que inseriu mais um passo, e assim à medida que os anos foram passando o método foi evoluindo e acabou se tornando mundialmente conhecido, onde também William ficou conhecido como pai do controle de qualidade nos processos produtivos.

[2] *Trial and Error: expressão conhecida como "tentativa e erro"*

O método apresenta 4 Passos a serem seguidos:

P (do inglês – Plan) = Planejamento
D (do inglês – Do) = Execução
C (do inglês – Check) = Verificação
A (do inglês – Act) = Atuar/Agir

Plan - A ordem começa pelo planejamento, onde se analisa os objetivos e metas, para assim se elaborar um plano para resolver os problemas encontrados, nessa fase se identifica o problema, observa, analisa e coloca ele em ação.

Do - Execução essa etapa é uma das mais importantes pois sem sua realização não é possível colocar em prática as etapas seguintes. Para iniciar a etapa da execução é fundamental que o planejamento esteja completo e que o que precisa ser feito esteja claro para os envolvidos.

Check - Verificação uma etapa em que é avaliado o que foi feito durante a execução, procurando identificar se deu certo ou não, geralmente, verifica-se se as atividades planejadas foram feitas corretamente, se o resultado esperado foi atingido e quais foram os pontos positivos e negativos na execução do plano.

Act - Atuar/Agir essa é uma etapa que requer maior atenção, nessa etapa se observar se os resultados foram alcançados, se sim se incorpora o método caso contrário se identifica as falhas e reinicia o ciclo novamente.

Inicialmente o ciclo foi desenvolvido para ser aplicado na administração da qualidade da gestão, sendo um método contínuo de aprimoramento da qualidade dos produtos e serviços de empresas.

O método PDCA passou por diversas alterações nos mais de 90 anos de existência e pode ser adaptado para qualquer tipo de empresa, esse método é utilizado para solucionar problemas que não são facilmente visualizados.

Esse processo é um método interativo, onde cada repetição chega a um resultado diferente e esses resultados serão utilizados nas vezes seguintes de forma acumulativa, assim pode se dizer que o PDCA é um ciclo que traz resultados diferentes a cada utilização.

Para que essas vantagens do PDCA sejam bem utilizadas é essencial que os colaboradores envolvidos no processo tenham cuidado ao formar a equipe responsável, que tenha treinamento adequado.

O modelo PDCA é muito importante porque possibilita a identificação de falhas, o que torna o processo de reparação mais fácil, se tornando um método que ajuda na implementação de novas ideias quanto à solução dos problemas.

Mesmo em um ambiente de documentação e construção de software; Essa abordagem é possível pelo alto nível trazido pelas tecnologias utilizadas, que permitem a reconstrução de certas partes do código e funções, sem atrasar severamente o andamento do processo de análise ou o próprio código.

3.1 Componentes de Construção e Ferramentas internas

1. Backend

Para a construção da base do software proposto, escolhe-se a linguagem de programação Python, devido ao alto nível de construção de código, e sua versatilidade e simplicidade quanto à criação de aplicativos independentes de maneira mais ágil

Python é uma linguagem de programação de alto nível, orientada a objetos, funcional, de tipagem dinâmica e forte, foi lançada por Guido van Rossum em 1991. A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Possui tipagem[3] dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens. Devido às suas características, ela é utilizada, principalmente, para processamento de textos, dados científicos e criação de CGI[4]s para páginas dinâmicas para a web.

[3] Tipagem: Maneira com que determinada linguagem de programação lida com variáveis e tipos de dados.

[4] *CGI (Common Gateway Interface)* Interface que utiliza um modelo de conexão Servidor – Clientes para a comunicação e transferência de dados.

Python suporta a maioria das técnicas da programação orientada a objeto, qualquer objeto pode ser usado para qualquer tipo, e o código funcionará enquanto houver métodos e atributos adequados, o conceito de objeto na linguagem é bastante abrangente: classes, funções, números e módulos são todos considerados objetos, unida ao *framework*[5] Django, pode ser facilmente utilizada para desenvolvimento web *Back-End* [6].

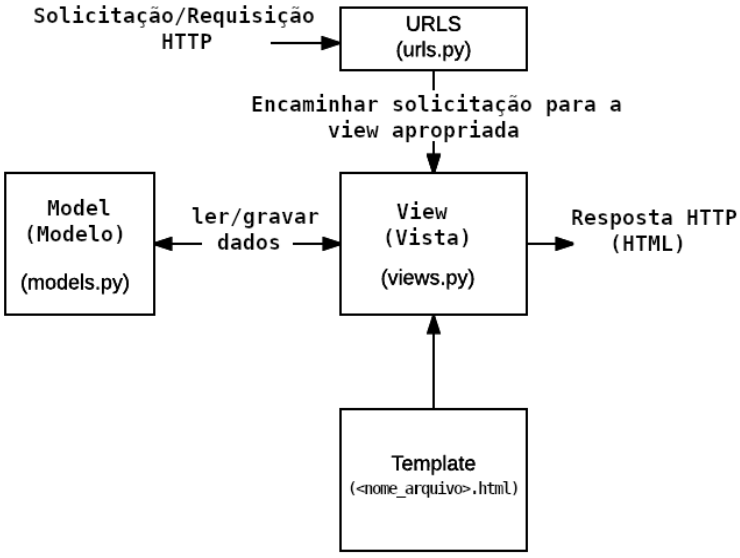
O *framework* Django foi criado originalmente como sistema para gerenciar um site jornalístico na cidade de Lawrence, no Kansas. Tornou-se um projeto de código aberto e escrito em Python que foi publicado sob a licença BSD em 2005. Tem como seus co-criadores os desenvolvedores Adrian Holovaty e Simon Wilkinson, o nome Django foi inspirado no músico de jazz Django Reinhardt, considerado um dos melhores e mais influentes guitarristas de todos os tempos.

Desde a sua versão 1.0 até a 2.0 lançada em 2017 teve várias correções proporcionando cada vez mais qualidade para o framework, com sua comunidade empenhada em trazer o melhor para qualquer tipo de Website. Suas características consistem em um desenvolvimento versátil, seguro, escalável, sustentável e portátil; alguns sites que utilizam o Django são: Disqus, Instagram, Knight Foundation, Mozilla, National Geographic, Pinterest e Openstack.

[5] *Framework:* conjunto de bibliotecas utilizadas para criar uma base onde as aplicações são construídas. Tem como principal objetivo resolver problemas recorrentes, ele permite ao desenvolvedor focar nos problemas a serem resolvidos e não na arquitetura e configurações.

[6] *Back-End*: Termo utilizado para referenciar a construção que é responsável pela lógica e processamento de dados em um sistema.

Aplicativos web feitos em Django geralmente agrupam o código que manipula cada uma dessas etapas em arquivos separados conforme o organograma a seguir.



Modelo MTV do Django; Neste tema existe uma pequena divisão na comunidade, uns dizem que o Django é MVC (Model View Controller), já outros dizem que é MTV (Model [7]*Template*[8] *View*[9]), e na verdade ambos estão certos, pois podem ser usados de duas formas, a questão é que no Django temos a ausência do Controller, porém podemos associar esta responsabilidade a View do Django. Sendo assim podemos encarar os *templates* como a View do MVC, e o Modelo permanece sendo os próprios modelos. Já no caso do MTV, que tem o funcionamento parecido com o MVC, ele descreve realmente o fluxo do Django: *models*, *templates* e *views*.

Para o gerenciamento de banco de dados, foi utilizado o SQLITE, uma biblioteca Transacional do tipo SQL, apresenta tipagem dinâmica e distribuição em domínio público, de

[7] *Model:* Modelo de mapeamento do banco de dados para um projeto que permite a manipulação de dados facilitada.

[8] *Template*: Páginas para visualização de dados, normalmente, é aqui que fica o HTML que será apresentada nos navegadores;

[9] *View:* Lógica de negócio, determina o que irá acontecer no projeto, programação em si.

tipo de armazenamento contido em arquivo e documentação organizada e detalhada, que facilita o processo de versionamento e testes com os dados do sistema. Trata-se de uma ferramenta extremamente confiável, não requer configurações pré-utilização e disponibiliza robusta estabilidade com mínimos empecilhos, que são corrigidos a cada nova versão (A grande parte de constituição do seu código Open Source[10] é dedicado ao teste para prevenção de falhas e outros problemas; mantém as propriedades *ACID*[11] mesmo após falhas consideradas catastróficas e quedas de energia enquanto é utilizada).

2. Front-End

Para a construção do *Front-end* [12] escolhe-se o ***Javascript***, que permite a interação do usuário com a aplicação, e permite a alteração de elementos de maneira dinâmica, e possibilita coisas que o Back-End não permite sozinho.

O **Javascript** é uma linguagem de programação e ferramenta de desenvolvimento, que no Front-End permite complexas mudanças e controle de comportamento de páginas web, permite a alteração de texto, mídia e elementos HTML e propriedades CSS de maneira rica e específica. Esse também permite a adição de bibliotecas de código pré-escrito, que facilitam o desenvolvimento.

[10] *Open Source*: Software que está em domínio público, de custo zero.

[11] ACID: Propriedades de transações que garantem a garantia de segurança e confiabilidade de dados encontradas em sistemas de gerenciamento de banco de dados (Atomicidade, Consistência, Isolamento e Durabilidade)

[12] *Front-End:* Programação que controla a aparência e funcionamento no lado cliente da aplicação. Geralmente é aqui onde o design e interação ocorre.

Algumas dessas bibliotecas foram escolhidas para esse projeto, sendo elas:

Bootstrap: Estilização de página, permite o controle visual da página web.

Datatables: Controle, filtragem e organização de tabelas de dados, visualização e pesquisa de dados dentro do Front-end.

Jquery*:* Controle facilitado de elementos HTML/CSS aliado ao Javascript

MomentJS: Utilizado para a formatação e organização de data/hora dentro de tabelas

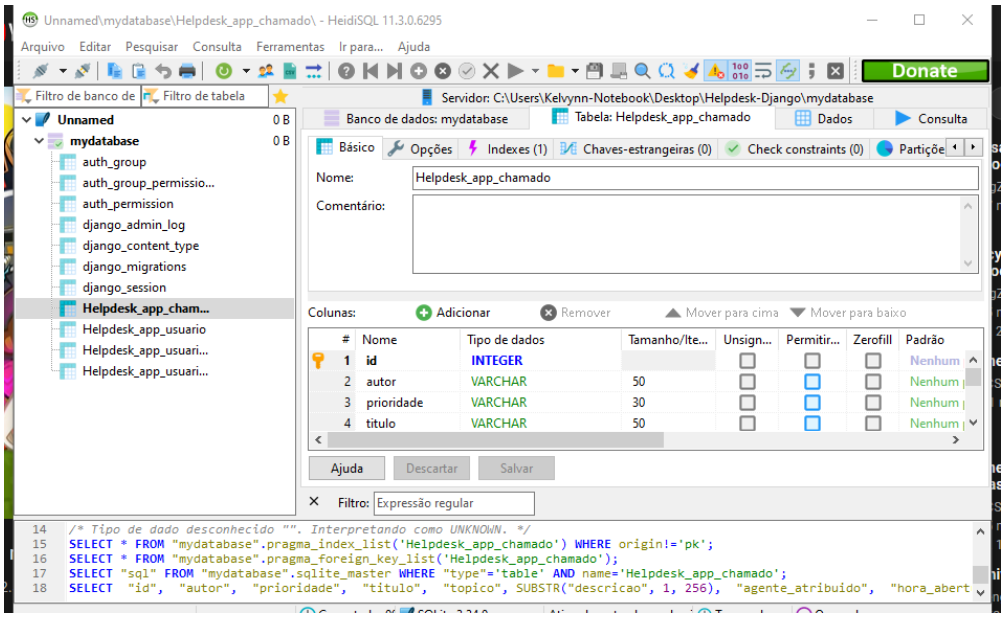
ToastrJs: responsável pelas notificações e alertas visuais no front-end (junto com a ferramenta "Messages" do Django)

FontAwesome: Utilizado para a adição de ícones dentro do ambiente visual do Sistema.

3.2 Definição De Ambiente De Desenvolvimento

Algumas das Ferramentas de desenvolvimento utilizadas foram:

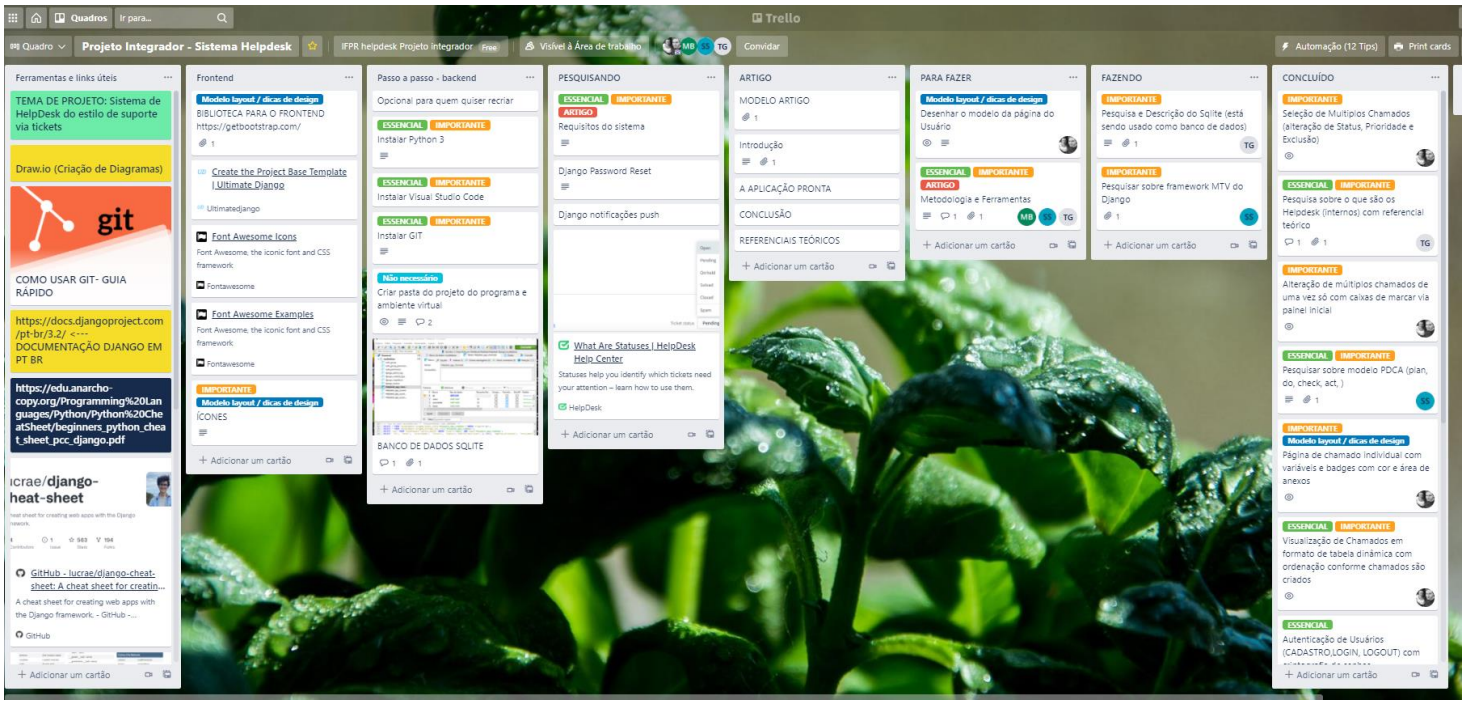
HeidiSQL (Gerenciamento, consultas e controle de banco de dados **SQLITE**),



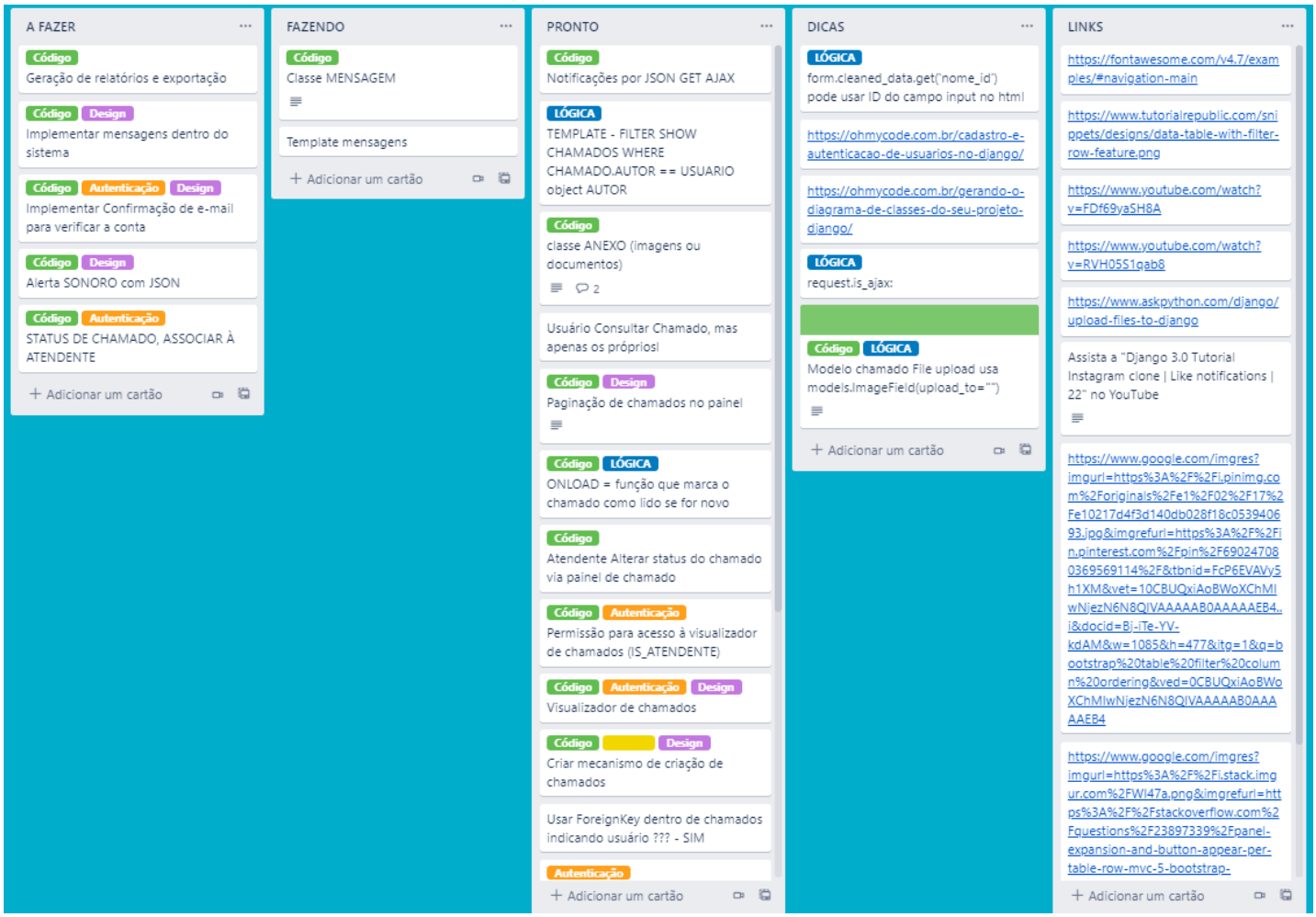
1 Exemplo da Interface HeidiSQL

SublimeText (Utilizado para o Desenvolvimento, utilizado especialmente para construção e edição de arquivos HTML e Javascript), **Visual Studio Code** (Utilizado para o desenvolvimento Python e linguagem de templates do Django), **Draw.io** (Utilizado para a criação de Esquemas gráficos e relacionais) **GIT** e **Github** (Gerenciamento de versões e controle de arquivos via repositório).

Para o controle de conteúdo e tarefas, foi utilizado o web site/aplicativo **Trello,** já que permite o controle de atribuições, planos e escopos de desenvolvimento e solução de problemas



2 Quadro Principal de gestão do projeto



4 Quadro de desenvolvimento

4. REFERÊNCIAS

Implantação de Help Desk e Service Desk (Roberto Cohen, 2008)

“What is the plan-do-check-act (PDCA) cycle?” - https://asq.org/quality-resources/pdca-cycle (acessado em 12/07/2021)

Introdução ao Javascript - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction