UNIVERSIDADE DE TAUBATÉ
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO DE TICKETS DE SUPORTE

TAUBATÉ
2022

BRUNO HOMERO MINORI SATO
JOSÉ TADEU COSTA CAMARGO DE OLIVEIRA
RENAN CASTILHO ALVES

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO DE TICKETS DE SUPORTE

Trabalho de graduação pelo curso de Análise e Desenvolvimento de Sistemas do Departamento de Informática da Universidade de Taubaté,
Area de Concentração: Sistemas da informação
Orientador: Prof. Dawilmar Guimarães Araújo

TAUBATÉ
2022

BRUNO HOMERO MINORI SATO
JOSÉ TADEU COSTA CAMARGO DE OLIVEIRA
RENAN CASTILHO ALVES

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO DE TICKETS DE SUPORTE

Trabalho de graduação pelo curso de Análise e Desenvolvimento de Sistemas do Departamento de Informática da Universidade de Taubaté,
Area de Concentração: Sistemas da informação
Orientador: Prof. Dawilmar Guimarães Araújo

Data: 14/12/2022
Resultado: Aprovado

BANCA EXAMINADORA

Prof. Me. Dawilmar Guimarães Araújo Universidade de Taubaté
Assinatura:_____

Prof. Dr. Márcio Augusto Ernesto Moraes Universidade de Taubaté
Assinatura:_____

Prof. Esp. Edgar Israel Universidade de Taubaté
Assinatura:_____

Grupo Especial de Tratamento da Informação - GETI
Sistema Integrado de Bibliotecas – SIBi Universidade de Taubaté - Unitau

S253d Sato, Bruno Homero Minori
Desenvolvimento de software de gestão de tickets de suporte / Bruno Homero Minori Sato , José Tadeu Costa Camargo de Oliveira , Renan Castilho. -- 2022.
17 f. : il.

Monografia (graduação) – Universidade de Taubaté, Departamento de Informática, 2022.
Orientação: Prof. Dr. Dawilmar Guimaraes de Araujo, Departamento de Informática.

1. Manutenção de sistemas. 2. Controle de manutenção. 3. Sistemas de controle online. I. Oliveira, José Tadeu Costa Camargo de. II. Castilho, Renan. III. Universidade de Taubaté. Departamento de Informática. Graduação em Análise e Desenvolvimento de Sistemas. IV. Titulo.

CDD – 005.1

Ficha catalográfica elaborada **por Shirlei Righeti – CRB-8/6995**

AGRADECIMENTOS

Agradecemos, primeiramente, à Deus, que nos deu energia e condições para desenvolvermos este projeto.

Agradecemos aos nossos familiares que sempre estiveram ao nosso lado e nos apoiaram em toda a jornada universitária.

Agradecemos aos professores que nos guiaram nestes anos e nos incentivaram a melhorar constantemente e buscar o conhecimento.

RESUMO

Sistemas de gestão de tickets, fundamentalmente, tem seu funcionamento baseado em check-ups cujo objetivo é demarcar e facilitar o gerenciamento dos pedidos, prever falhas e diminuir eventuais problemas que podem ocorrer, como falhas de comunicações entre setores e perda de registros. Atualmente, são pouquíssimos os sistemas disponíveis gratuitamente que fazem esse controle de tickets ou que não exigem máquinas mais potentes que suportem a robustez do sistema. Para resolver essas questões, este trabalho propõe como solução o desenvolvimento de um sistema de gerenciamento online de tickets. Identificando necessidades cotidianas dos membros do grupo, buscou-se fazer uso dos vários conceitos abordados ao longo do curso nas diversas matérias vistas em sala de aula, como a linguagem Python juntamente ao framework Flask; ou a biblioteca SQLite para o armazenamento de informações; conceitos de HTML, CSS, Javascript e Bootstrap no desenvolvimento da interface front-end, responsável por toda a responsividade e grande parte da estilização do projeto, facilitando seu desenvolvimento e diminuindo o tempo de codificação. O resultado desejado é um sistema simples e objetivo, que permita a utilização em diversos periféricos, possibilitando a um maior número de usuários diminuir as perdas e aumentar a produtividade.

Palavras-chave: manutenção de sistemas; controle de manutenção; sistemas de controle online.

ABSTRACT

Ticket management systems have their operations based on regular check-ups, whose objective is to ease the management of products, predict failures and reduce possible problems that may occur, such as communication failures between sectors and loss of records. Currently, there are very few systems available for free to do this job or that do not require powerful pcs to support the robustness of the systems. To solve these issues, this work proposes the development of an online ticket management system as a solution. By identifying the daily needs of the group members, it was possible to make use of many concepts covered throughout the course, such as the Python language along with the Flask framework; or the SQLite library for storing information; and the concepts of HTML, CSS, Javascript and Bootstrap in the development of the front-end interface, responsible for all the responsiveness and much of the project's styling, facilitating its development and reducing coding time. The desired result is a simple and objective program, allowing the use by several peripherals configurations, increasing the number of users, reducing their losses and increasing their productivity.

Keywords: systems maintenance; maintenance control; online management.

SUMÁRIO

1. INTRODUÇÃO

A manutenção dos equipamentos é um assunto de grande debate, independente da área de atuação. Ela busca prever possíveis falhas e problemas que possam ocorrer durante o uso dos mais variados equipamentos, através de check-ups periódicos que permitem a identificação de pequenos problemas antes que eles se tornem grandes (ALVES, MARTINS e PAULISTA, 2016).

Várias são as vantagens apresentadas ao executar as manutenções preventivas, visto que existe uma tendência de o desempenho dos equipamentos reduzir com o passar do tempo. Nesse sentido, a manutenção permitirá atenuar os possíveis impactos dessas ações.

Eventuais problemas de produtividade, resultados de diversos fatores como o excesso de arquivos desnecessários, acúmulo de sujeira, envelhecimento dos componentes internos, desatualização de seus softwares etc., também são diminuídos com as manutenções. Para Lopes (2017), dentro de um cenário empresarial que pode abranger uma grande quantidade de equipamentos, tal diminuição na produtividade pode acabar gerando gastos muito elevados e que devem ser evitados.

Visualizando uma possível demanda, realizou-se um levantamento acerca das soluções disponíveis no mercado que fornecessem aos usuários alguma forma de organizar e controlar a manutenção de equipamentos. Como resultado, foi possível observar que existem diversas empresas que disponibilizam sistemas de gerenciamento de ticket, tanto para serviços voltados ao software quanto hardware. O que se notou também é que boa parte dos sistemas acabam sendo pagos – podendo ser um fator limitador para alguns. Ignackuz (2020) salienta a importância dos sistemas de tickets, capazes de rastrear todas as interações dos clientes e acompanhá-los até a finalização do chamado. A autora também reforça, dentre as várias vantagens desses sistemas, a facilidade de elencar prioridades e verificar as tarefas que demandam mais urgência, o aumento da produtividade de todo o time de suporte, a centralização de informações e dados e a descentralização do conhecimento, permitindo a continuidade, sem rupturas, no acompanhamento e finalização de tickets em aberto.

Como apontado previamente por Lopes (2017), a falta de manutenções pode acabar culminando em grandes problemas, tanto materiais como financeiros, com perda de equipamentos e atraso para finalização de projetos e serviços.

Paduelli (2007) também enfatiza que o cenário de desenvolvimento dos softwares tende a se tornar cada vez mais competitivo e que, nos mais diversos ramos, a clientela tende a priorizar cada vez mais a credibilidade e confiabilidade dos produtos e serviços, fomentando assim a busca por uma solução que se destacasse na qualidade destes itens.

1.1. OBJETIVOS

O objetivo principal deste trabalho é atender as necessidades dos profissionais de Help Desk que buscam por um programa responsável pelo gerenciamento de manutenções e que conte também com o controle dos equipamentos, tanto interno como externo para os clientes também ver o andamento do serviço solicitado, através de tickets de serviço.

Para tal, a proposta apresentada será desenvolvida como uma aplicação online de fácil utilização pelos usuários.

1.2. JUSTIFICATIVA

Focado na plataforma web, a aplicação busca proporcionar uma maior facilidade de acesso a todos os usuários. Tal proposta é adotada justamente para que não haja necessidade de aquisição de máquinas com hardware específico sendo necessário uma máquina que consiga rodar um navegador, promovendo uma maior amplitude de usuários possíveis, bem como diminuir eventuais incompatibilidades de sistemas em máquinas mais antigas e mais limitadas. Tal abordagem também tem como objetivo agilizar e facilitar a entrega dos equipamentos, visto que é um sistema compatível tanto com desktop como mobile, através do navegador disponível que contará com recursos de responsividade.

Este trabalho também busca contribuir, diretamente, na questão da baixa disponibilidade de sistemas voltados para esse objetivo específico. Ele também

poderá auxiliar na divulgação de métodos para aprimorar e estimular futuras pesquisas e desenvolvimentos nesta área ao mostrar, de forma clara e objetiva, as melhores práticas no desenvolvimento de sistemas, elaboração de registros e documentação.

1.3. METODOLOGIA

Durante o desenvolvimento da aplicação, serão abordados e aplicados os conceitos e princípios das boas práticas de programação, para que o software final possa ser utilizado como referência em estudos futuros e, também, como estudo de caso prático. Tais princípios a serem abordados irão compreender desde a metodologia adotada para a gestão do projeto, a escolha da arquitetura do software e os testes de qualidade aplicados para garantia do bom funcionamento do programa.

1.3.1. Testes de Qualidade

Para garantir a boa funcionalidade de um programa, é necessário a realização de sua testagem. Ela é essencial para que o produto a ser entregue esteja funcionando dentro das expectativas, com todas as funções funcionando adequadamente e sua usabilidade estar dentro dos parâmetros desejados (LOPES, 2017).

Conforme Vargas e Pereira (2019), o teste de softwares é um processo de checagem aplicado aos programas em fase de desenvolvimento como ilustrado na figura 1, podendo ser realizados para avaliar diferentes aspectos. Dentre os aplicados ao longo do desenvolvimento do presente trabalho, pode-se citar:

- Teste de Usabilidade: aplicado para avaliar a qualidade do software na experiência do usuário, buscando compreender o quão intuitivo, compreensível e inteligível a interface do programa se mostrou para o usuário;
- Teste Funcional: para atestar que a aplicação seria capaz de desempenhar suas funções, optou-se pelo teste de caixa-branca – análise do código fonte do

software, escolhendo partes específicas que permitissem uma busca precisa do comportamento da estrutura;

- Teste de Regressão: para evitar a recorrência de um erro após a correção de bugs e acréscimo de funcionalidades, foram realizadas varreduras para identificar falhas até então inexistentes, evitando instabilidades que pudessem comprometer o sistema;

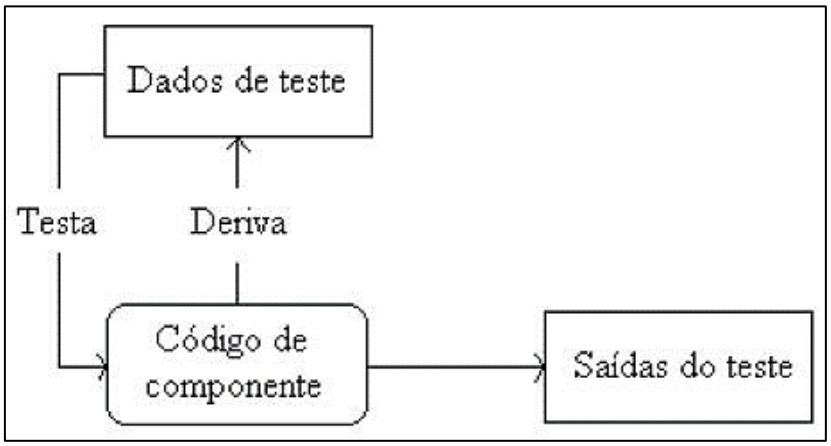

Figura 1 – Processo do teste de softwares.
Fonte: UFPE, 2020.

1.3.2. Metodologia Ágil

Uma metodologia ágil é um conjunto de técnicas e práticas, que podem ser aplicadas na gestão de projetos. Inicialmente englobava somente a área de Tecnologia da Informação, mas pode ser atualmente aplicada em todas as áreas (MIRANDA e ALVES, 2018).

Segundo os autores, os métodos comuns de planejamento, execução e entrega, podem não responder satisfatoriamente às necessidades de velocidade e de constantes alterações demandadas pelos clientes pois consistem em sequências predefinidas de etapas, como análise de requisitos, desenvolvimento, testes, produção e manutenção, podendo tornar os projetos bastante demorados.

Em contrapartida, as metodologias ágeis oferecem rapidez, eficiência e flexibilidade, simplificando os processos e tornando-os mais dinâmicos e iterativos, desde a concepção da ideia até o produto final (FADEL e SILVEIRA, 2010). Isso é possível já que, nessa técnica, os projetos são divididos em pequenas entregas,

compostas por etapas planejadas em um ciclo rápido e eficiente, resultando em uma entrega parcial que permite ao cliente ver os resultados rapidamente e dar seu feedback durante todo o processo. Ao longo do tempo, com a repetição dos ciclos, o produto vai sendo continuamente aprimorado e pode ser testado a cada funcionalidade, entregando mais valor em menos tempo.

Para a implementação deste projeto, optou-se pela utilização da metodologia Scrum – um framework simples adotado constantemente em projetos complexos a partir de pequenos ciclos de atividades, denominados Sprints. Cada sprint, estabelecida em uma semana, permitiu a melhor divisão das tarefas entre os colaboradores, melhor definição das prioridades de estudo e programação, e validação das ações implementadas, bem como a revisão das atividades desenvolvidas, como ilustrado nas Figuras 2(a) e 2(b).

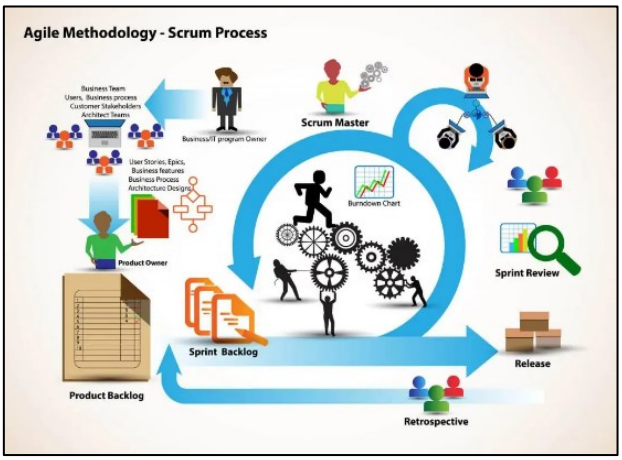

Figura 2(a) - Representações gráficas do fluxo de trabalho dentro da metodologia Scrum, desde a elaboração até a entrega.

Figura 2(b) - Representações gráficas do fluxo de trabalho dentro da metodologia Scrum, com enfoque na Sprint.

Fonte: DIGITÉ, 2022.

1.3.3. Modelo MVC

Outro ponto importante levantado durante a definição dos parâmetros do trabalho foi em relação à escolha do padrão de arquitetura de software mais adequado. De grande relevância, a definição da arquitetura possibilita a otimização do trabalho de design e desenvolvimento, permitindo à aplicação estar dentro dos

padrões necessários para funcionar, com seus componentes responsáveis pela realização de um conjunto específico de funções bem estruturados, gerando muito mais facilidade de manutenção ao longo do tempo (LEMOS et al., 2013). Outra vantagem ao utilizar um padrão é a criação de interfaces bem definidas com componentes e funcionalidades já testados e a possibilidade de reutilização de códigos.

Dentre as várias arquiteturas disponíveis, foi escolhido o MVC, pelo fato de ser a mais conhecida e empregada entre os desenvolvedores. Sua escolha se dá pelo fato de ser uma arquitetura de software responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários. Luciano e Alves (2011) apresentam a arquitetura MVC e sua divisão em três componentes essenciais conforme a figura 3:

- *Model*: a camada responsável por manter e lidar com as informações, gerenciando e controlando as funções, lógica e regras de negócios estabelecidas. Recebe as informações do *Controller*, faz a validação se estão corretas ou não e envia a resposta mais adequada;
- *Controller*: essa camada é responsável por intermediar as requisições enviadas pelo *View* com as respostas fornecidas pelo *Model*, processando os dados informados pelo usuário e repassando para demais camadas;
- *View*: é camada responsável por apresentar as informações de forma visual ao usuário. Se relaciona com a UI (interface) do sistema.

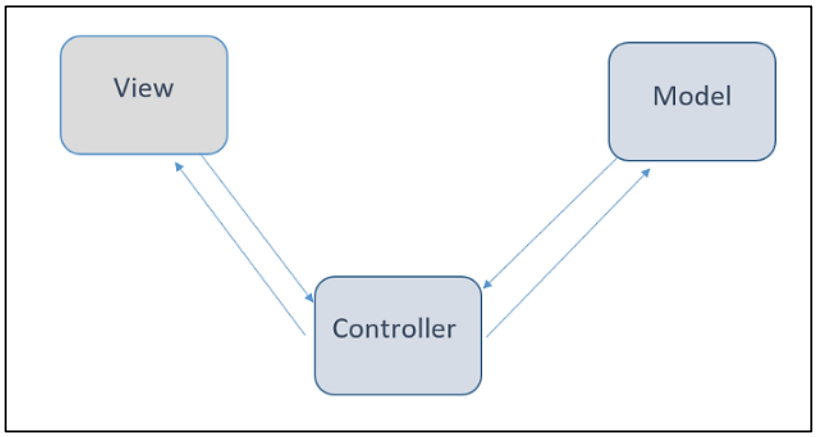

Figura 3 – Modelo de representação do MVC.
Fonte: SVIRCA, 2020.

1.3.4. Cronograma

Com a abordagem proposta no framework *Scrum* foi possível estruturar uma sequência de atividades que podem ser divididas em quatro grandes sprints e quatro entregas, sendo a primeira a Concepção e Planejamento, onde foram levantados os requisitos e identificado as especificidades do projeto, a segunda entrega a Padronização de Ambiente, onde todos os envolvidos adequaram seus equipamentos diversos para todos colaborarem em um mesmo ambiente neutro, a terceira entrega sendo o desenvolvimento de fato do software e a última entrega sendo esta documentação.

As entregas ocorreram em ciclos de trinta dias, sendo este o tempo das sprints realizadas, e foi possível respeitar quase integralmente o escopo planejado, com algumas pequenas alterações nos prazos devido a correções de problemas encontrados, conforme ilustrado na Figura 4 a seguir:

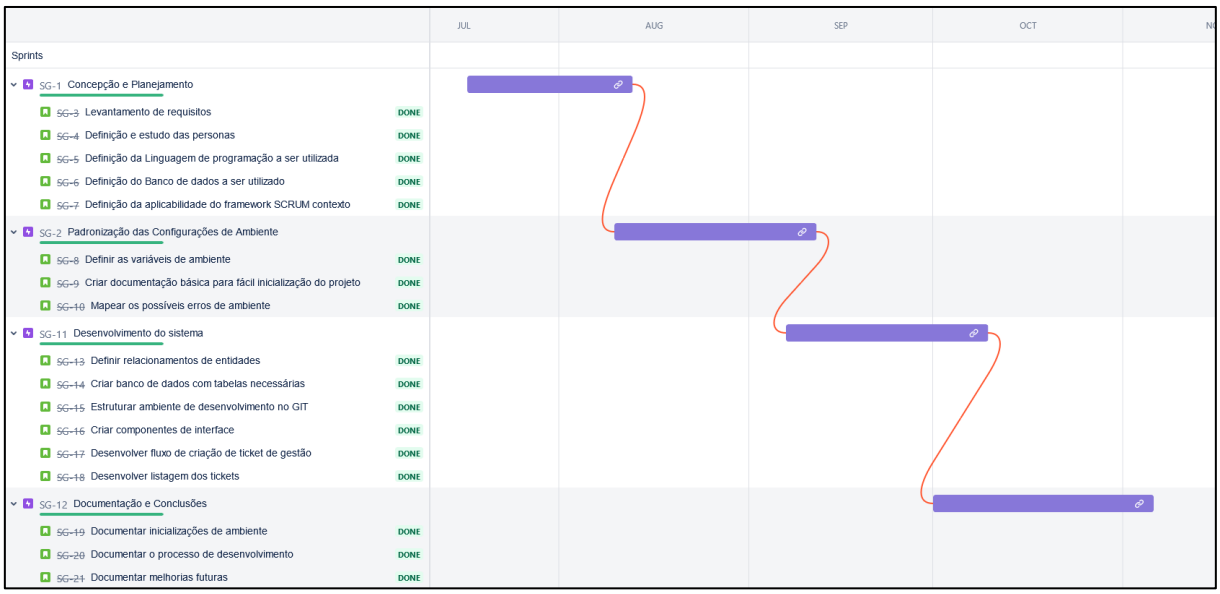

Figura 4 – Representação gráfica do cronograma de planejamento das sprints e realização do projeto.
Fonte: AUTORES, 2022.

2. REVISÃO BIBLIOGRÁFICA

O gerenciamento de processos de negócios pode ser visto como a busca pela melhoria contínua. Varvakis et al. (1998) sustentam esta visão afirmando que a gestão de processos é a definição, avaliação e melhoria contínua dos processos para atender às necessidades e expectativas. Este escritor acrescenta ainda que esta procura de melhoria contínua inclui motivação, criatividade e trabalho árduo. Ele também diz que a gestão de processos busca alcançar as melhores condições para o cliente, com base nos princípios básicos de qualidade, análise de valor, *just in time*, entre outros.

Portanto, Harrington (1993) defende a ideia de que a gestão deve trabalhar com o sistema, enquanto os funcionários devem trabalhar com o sistema. Em outras palavras, a melhoria contínua e os processos de mudança devem ser orientados para o processo e não para a pessoa. “Isso significa que todas as funções devem trabalhar juntas para aumentar a eficiência, eficácia e flexibilidade de todo o processo” (Harrington, 1993, p. 368).

O conceito anterior combina inovação com qualidade e geração de valor. Gonçalves (2000) também destaca a importância da inovação como fonte de competitividade ao afirmar que, no século XX, as empresas japonesas direcionaram 70% de seus recursos de pesquisa e desenvolvimento para a inovação, obtendo, nesse processo, resultados superiores aos americanos. companhia., que aplicou a mesma medida à produção do produto.

2.1. SERVIÇOS DE TICKETS

De um modo geral, um sistema de gerenciamento tickets executa uma função mais automatizada e ajuda a equipe de a gerenciar o ciclo de vida de cada solicitação ou processos. É recomendado para facilitar o giro de pedidos e informações. O gerenciamento de solicitações é um processo de gerenciamento de projetos que envolve o gerenciamento de solicitações de serviços ou produtos. É um processo que é recebido, revisado, aprovado ou rejeitado, e entregue ao cliente rapidamente. Muitas empresas usam o gerenciamento de solicitações para gerenciar a complexidade de um grande número de solicitações de muitos clientes. O processo de gerenciamento

de solicitações exige que a organização estabeleça um processo de aceitação, gerencie os recursos necessários para atender à solicitação e gerencie a resposta do cliente. O gerenciamento de solicitações também pode incluir o rastreamento do andamento das solicitações e a resolução de quaisquer problemas que possam surgir durante o processo.

3. PLANEJAMENTO E CONCEPÇÃO

O planejamento se inicia desde o momento inicial que se passa a aprofundar no problema abordado e na solução proposta, passando desde o levantamento de requisitos, onde são expostos os principais requisitos que o sistema deverá ter, as personas envolvidas e como facilitar para elas o uso do sistema e por fim tomar as decisões de quais tecnologias utilizar para o desenvolvimento.

3.1. LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos acaba sendo uma das mais importantes etapas no processo de desenvolvimento de um software ou sistema é a coleta de necessidades a serem sanadas.

Definir claramente o que um sistema deve fazer é fundamental para o sucesso do projeto, além de economizar tempo e dinheiro no desenvolvimento. Portanto, todos os processos e programas de desenvolvimento devem ser cuidadosamente planejados. Para isso, podem ser utilizados métodos de documentação ricos ou métodos ágeis, levando em consideração as necessidades do cliente e as necessidades do sistema (DILKIN, 2020).

Para cumprir o propósito de sua criação, o software em questão deve ser acessível facilmente e ser simples de usar, para este sanar este primeiro quesito foi definido que será desenvolvido em um ambiente WEB dessa forma não utilizará muito do hardware e memória do computador local já que o sistema é WEB, tendo como requisito de hardware um computador que rode o navegador e a simplicidade foi aplicada no desenvolvimento das telas da aplicação, que buscam pelo minimalismo, dispondo na vista do usuário apenas o necessário para o uso.

3.2. PERSONAS

Uma persona é um personagem semificcional, baseado em dados e comportamentos reais, que representa o cliente ideal de uma marca ou empresa. Também é conhecido como comprador ou avatar. A persona direciona a criação do software e foi também considerada neste escopo de projeto (PEÇANHA, 2020).

A principal persona que merece ser considerada é a dos gestores de setores empresariais, que costumam ser pessoas ocupadas e prezam pela agilidade e simplicidade mais do que recursos complexos e demorados.

Considerando então a Persona que representa nosso usuário final, a interface desenvolvida de forma minimalista e simples gera a maior aceitação de uso e agrega ao usuário facilidade de entendimento de seus recursos.

3.3. HARDWARE

Como se trata de uma aplicação WEB o terminal quanto o servidor precisa somente atender os requisitos impostos pelo navegador, para conseguir rodar o sistema, como por exemplo o navegador Google Chrome que tem como requisitos Windows 7 ou mais recente e um processador Pentium 4 ou mais recente (GOOGLE, 2022).

3.4. ESPECIFICAÇÕES NECESSÁRIAS

Nesta etapa, tendo levantado todos os requisitos do sistema, resta necessário definir as tecnologias que serão utilizadas no desenvolvimento e por fim dar início à fase de produção do software em si.

3.4.1. Definição da Linguagem Python

Foi adotada a linguagem Python para desenvolver o sistema pois se trata de Linguagem que todos os colaboradores envolvidos tinham conhecimento mínimo para contribuir no desenvolvimento e proporcionar um produto que esteja alinhado com o trabalho em equipe desenvolvido.

3.4.2. Definição do Banco de Dados SQLite

Quanto ao banco de dados adotado, restou definido o SQLite a ser utilizado na aplicação, devido a sua baixa complexidade de implementação e pela sua agilidade e pouco peso, que proporcionam ao software um ganho de agilidade tanto no desenvolvimento quanto na etapa de utilização pelo usuário final.

3.4.3. Definição do framework Flask de desenvolvimento

Ao buscar um framework que facilitaria o desenvolvimento do sistema e teria um grau de aprendizado e complexidade fácil ou moderado, uma vez que o tempo previsto para as entregas curto e é necessária aprender para construção do sistema contando com um ambiente simples e ágil ao usuário.

Dentre as opções a que mais se destacou entre a equipe foi o Flask, que é um framework robusto o suficiente para atender o objetivo proposto e também conta com um grau de complexidade e aprendizado muito menor perante as outras opções do mercado.

3.5. APLICABILIDADE DO FRAMEWORK SCRUM

Dentro da metodologia ágil, há o framework SCRUM que traz consigo inúmeras propostas de procedimentos de atuação focados no trabalho em equipe, comunicação ágil, autogerenciamento e muitos outros aspectos para um desenvolvimento ágil, contudo, não seria viável aplicar tal framework em sua totalidade no desenvolvimento do software foco deste trabalho, sendo então filtrado quais aspectos poderíamos adotar em nossa jornada de planejamento e desenvolvimento que nos trariam resultados positivos.

Ficou definido então que seriam aplicados neste projeto o modelo de desenvolvimento por Sprints, sendo estas de 30 dias, a estruturação dos afazeres divididas em pequenas tarefas, dando origem então ao Backlog do projeto e ao roteiro de entregas, conforme exposto no cronograma nos capítulos anteriores.

Outro aspecto adotado foi a de reuniões de equipes, amplamente conhecidos como Daily Meeting, contudo devido a inconsistência da presença dos envolvidos logo foi passada para uma reunião semanal, ao invés de diária, para tratar dos avanços conquistados e dos impeditivos que estavam impedindo de progredir.

4. DESENVOLVIMENTO DO SOFTWARE

Pensando nisso, começamos a identificar problemas no nosso cotidiano que seriam interessantes serem solucionados, nos colocando como um cliente, identificamos que em muitos ambientes de trabalho não se tem uma maneira de se controlar o que foi solicitado a um departamento ou para algum funcionário (FERNANDES, 2003).

Com um problema a ser solucionado em mãos, iniciamos o nosso processo de levantamento das informações mais necessárias para dar início ao desenvolvimento.

Utilizando conceitos da engenharia de software começamos a pensar nas maneiras mais adequadas para que sejam solucionados e atendidos todos os problemas, com isso fizemos a seguinte lista: quais os produtos de software existentes no mercado que podem ser utilizados para solucionar os problemas e atender as necessidades do usuário; como devem ser esses produtos para atender bem aos requisitos do usuário; quais os problemas que podem surgir na aquisição e implementação dos produtos (FERNANDES, 2003).

É importante que os requisitos sejam bem compreendidos pelo cliente, pode ser adquirido um produto inadequado e, ao tentar implementá-lo, surgirão problemas que podem ser irremediáveis.

Após finalizar o planejamento, iniciamos a configuração do ambiente e realmente o início do desenvolvimento do projeto.

4.1. CONFIGURAÇÃO DO AMBIENTE

Começando a configurar o ambiente de desenvolvimento teríamos que escolher uma IDE ou editor de código-fonte para começar o desenvolvimento e execução do projeto.

IDE – *Integrated Development Environment*, ou em português Ambiente de Desenvolvimento Integrado, é um programa utilizado basicamente para que os desenvolvedores de software tenham um aumento na produtividade agilizando o processo durante o desenvolvimento, também utilizado para facilitar e auxiliar o desenvolvedor com o processo de digitação, gerando uma redução de erros de

digitação. Com a diminuição no tempo de produção do software, automaticamente gera uma diminuição nos custos do projeto (SANTOS, 2007)

Para o desenvolvimento do nosso projeto foi escolhido o Visual Studio Code que nada mais é que um editor de código-fonte desenvolvido pela Microsoft e, segundo a própria documentação, é compatível com os principais sistemas operacionais do mercado que são eles Windows, Linux e macOS, essa ferramenta tem uma ampla variedade de extensões que facilitam a digitação, diminuição de erros, tem a função de depuração que é muito útil para detectar bugs e erros, também tem compatibilidade com o controle de versionamento do Git, por esse e outros motivos resolvemos utilizar o mesmo (MICROSOFT, 2022).

Após escolher o nosso editor de texto, começamos a configurar realmente o nosso ambiente, para desenvolver nosso software com o Framework Flask utilizaremos um ambiente virtual.

Aplicações em Python na maioria das vezes, utilizam pacotes e módulos adicionais que na sua instalação padrão não vem instalado, em certos casos aplicações precisam de uma versão específica de uma biblioteca para resolver um problema que tenha sido solucionado utilizando uma versão antiga da biblioteca, isso significa que seja provável que a instalação padrão do Python não supra as necessidades de qualquer aplicação. Para solucionar problemas desse tipo é necessário criar um ambiente virtual, “uma árvore de diretórios que contém uma instalação Python para uma versão particular do Python, além de uma série de pacotes adicionais” (PYTHON SOFTWARE FOUNDATION, 2022).

O módulo utilizado para a criação e gerenciamento de ambientes virtuais em Python é chamado de venv. Como nosso sistema foi desenvolvido em Python3 por padrão já vem instalado o módulo venv. Para criar o ambiente virtual, deve ser escolhido o diretório onde deseja ser colocado, após isso deve ser executado o módulo venv como um script com o caminho do diretório onde desejamos que seja alocado para dar início ao desenvolvimento do projeto, porem antes de ser alocado o venv deve estar instalado o python3 para que seja reconhecido os comandos, todos os comandos a seguir são rodados dentro do prompt de comando, a figura 5 a seguir monstra a instalação do Python 3 e do venv (Flask, 2022).

```
C:\...\SSTI>python3

C:\...\SSTI>python3 -m venv venv
```

Figura 5 – Comando para inicialização do ambiente virtual.
Fonte: FLASK, 2022.

Com a instalação do venv concluída dentro do diretório que você escolheu terá uma estrutura similar a estrutura ilustrada na figura 6 a seguir:

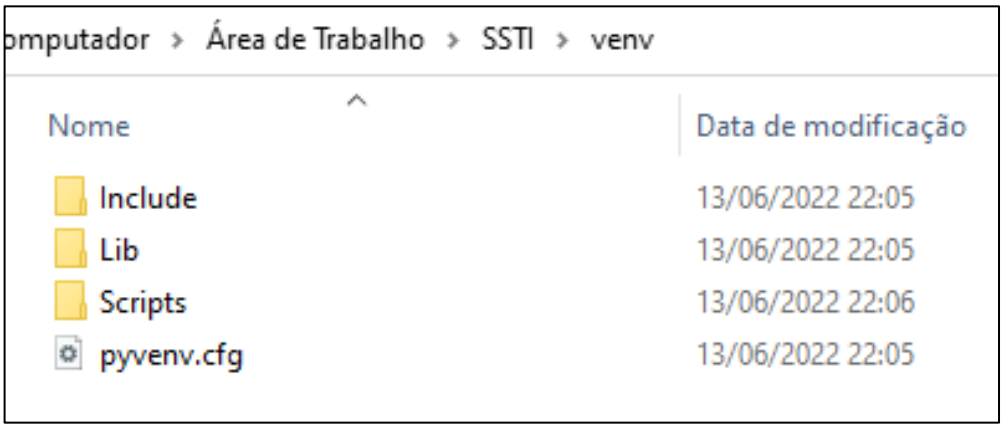

Figura 6 – Ambiente virtual configurado.
Fonte: AUTORES, 2022.

Com a criação do ambiente virtual concluída podemos iniciar a instalação do Framework Flask. O Flask será instalado dentro do nosso ambiente virtual venv, que para entrar nele você deve ativá-lo, utilizando o arquivo .bat que fica dentro da pasta Script (PYTHON SOFTWARE FOUNDATION, 2022).

C:\...\SSTI>venv\Scripts\activate

Após a ativação do ambiente virtual irá haver uma alteração no seu prompt de comando onde mostrará qual ambiente virtual está sendo utilizado (PYTHON SOFTWARE FOUNDATION, 2022).

(venv) C:\...\SSTI\venv\Scripts\>

Feito isso pode dar início a instalação do Flask, dentro do venv podemos utilizar um programa chamado pip que tem como função o gerenciamento de pacotes, com ele você pode instalar, remover e atualizar pacotes, este programa tem uma variedade de subcomandos por exemplo o install, que foi o que utilizamos para fazer a instalação do Flask. Voltando para a pasta raiz do nosso sistema iremos executar a instalação do Flask conforme ilustrado na figura 7, o script fica da seguinte forma:

(venv) C:\...\SSTI> pip install Flask

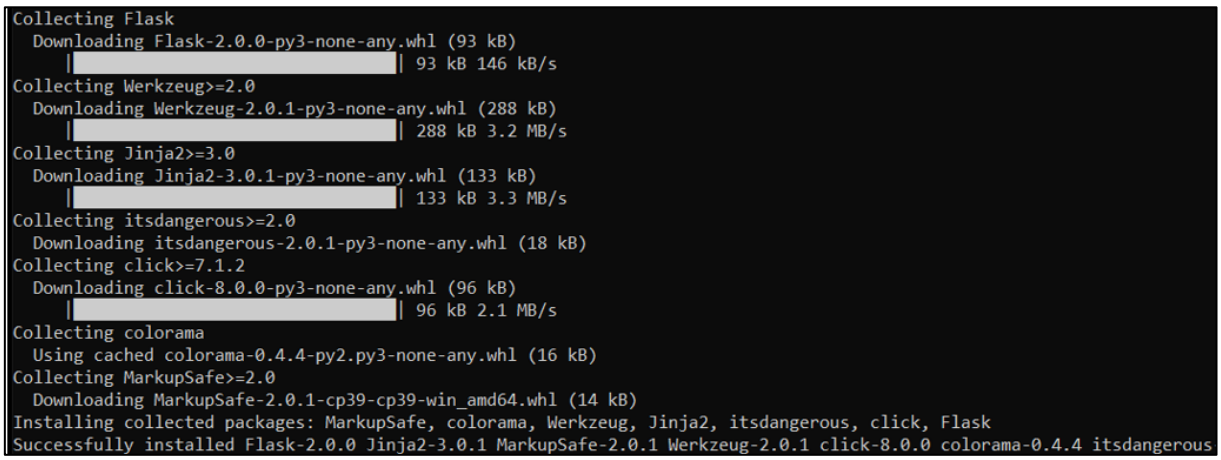

Figura 7 – Prompt de instalação correta do Flask.
Fonte: FLASK, 2022.

Dessa forma podemos considerar que toda a configuração do ambiente está finalizada, permitindo assim a continuação no desenvolvimento.

4.2. DESENVOLVIMENTO DO SOFTWARE

No final o software desenvolvido terá o visual ilustrado na figura 8 e 9 a seguir:

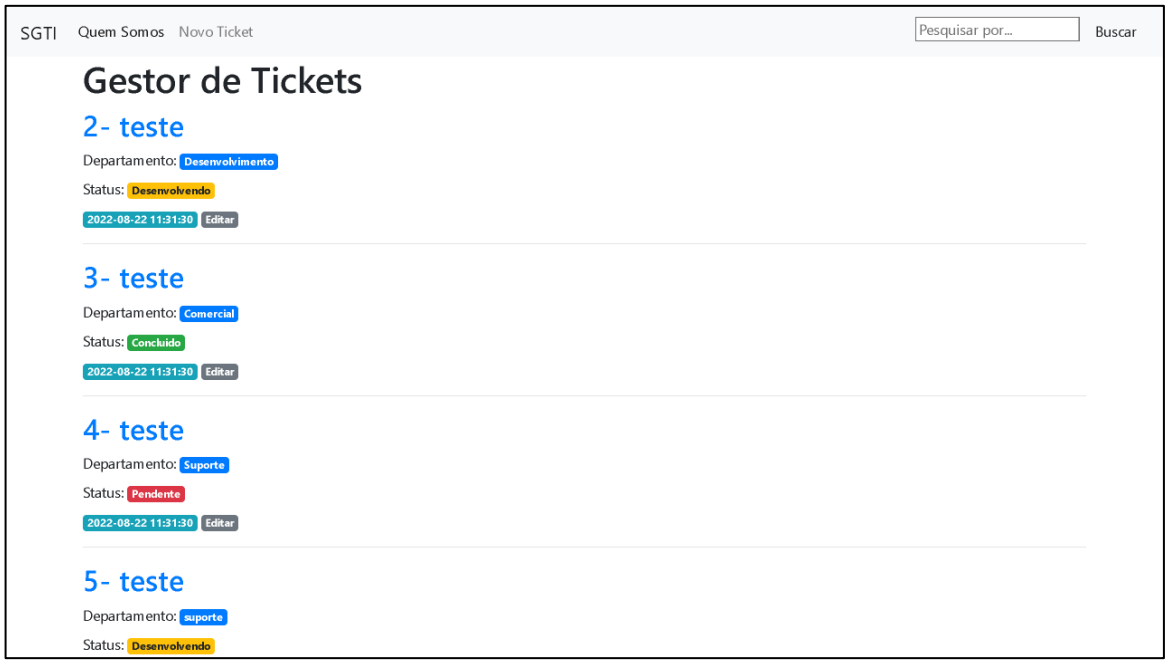

Figura 8 – Demonstração da interface da aplicação em desktop.
Fonte: AUTORES, 2022.

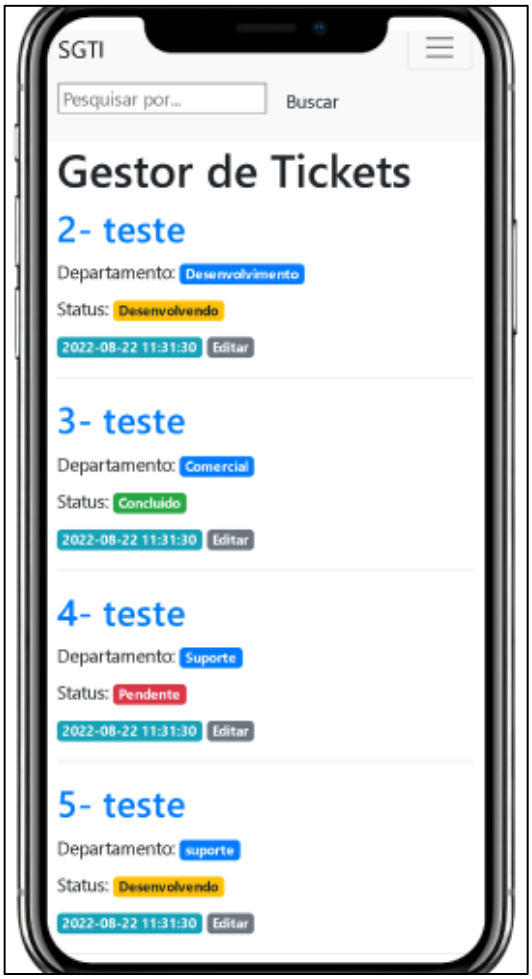

Figura 9 – Demonstração da interface da aplicação em celulares.
Fonte: AUTORES, 2022.

Com o ambiente de desenvolvimento configurado, podemos iniciar o desenvolvimento e o uso do Flask.

Para testar o nosso ambiente de desenvolvimento foi criado um arquivo app.py que através dele irá iniciar o servidor, que exibirá informações no navegador.

Este arquivo funcionará como um pequeno exemplo de como fazer uma solicitação HTTP. Dentro você importará o objeto e criará uma função que retornará uma resposta HTTP. Escreva o seguinte código em app.py conforme a figura 10 a seguir:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello():
    return 'Hello, World!'
```

Figura 10 – Exemplo de solicitação HTTP com Flask.
Fonte: FLASK, 2022.

No bloco de código anterior, importando o objeto Flask do pacote Flask. Use-o para criar seu aplicativo Flask chamado app. Passe uma variável '__name__' exclusiva que contém o nome do módulo Python atual. Ele é usado para descrever um exemplo onde ele existe.

Depois de criar um aplicativo de exemplo, use-o para processar a entrada do usuário e enviar comentários. O @app.route transforma funções padrão do Python em funções de visualização do Flask. Ele converte o retorno do serviço em uma resposta HTTP para um valor de navegador de um cliente HTTP, como um site HTTP, como um navegador. Passe o valor '/' para @app.route() para indicar que esta função responderá às solicitações da web para a URL /, que é a URL base.

A função de verificação hello() retorna a string "Hello, World!" como resposta. Salve e feche o arquivo.

Para executar seu aplicativo da web, primeiro informe ao Flask onde encontrar o aplicativo (o arquivo app.py) e a variável de ambiente FLASK_APP conforme a figura 11 a seguir:

```
(venv) C:\SSTI>set flask_ENV=development

(venv) C:\SSTI >set FLASK_APP=app

(venv) C:\SSTI >flask run
```

Figura 11 – Comando de inicialização do Flask.

Fonte: FLASK, 2022.

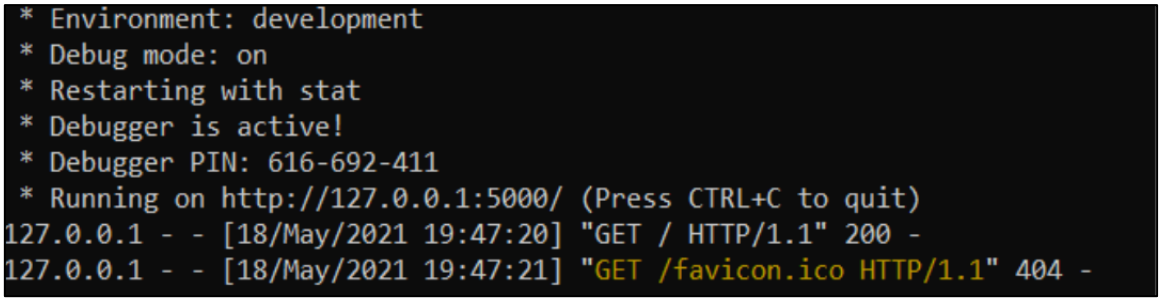

Figura 12 – Retorno de sucesso ao iniciar servidor de desenvolvimento rápido.

Fonte: FLASK, 2022.

Os resultados ilustrados na figura 12 anteriormente contêm informações diferentes, como:

- O nome do aplicativo que você inseriu.
- O ambiente em que o aplicativo funciona.
- Modo de depuração: ligado indica que o depurador do Flask está em execução. A solução de problemas é muito útil porque não é útil para solução de problemas.

- O aplicativo é executado localmente na URL http://127.0.0.1:5000/, 127.0.0.1 é o IP que representa o host local da sua máquina e :5000 é o número da porta.

Abra um navegador e digite a URL http://127.0.0.1:5000/. Você receberá a string ‘Hello, World!’ em resposta. Isso mostra que seu aplicativo está funcionando corretamente.

Fazendo isso será criado a base do software em WEB Flask.

4.3. INSERINDO HTML

Neste ponto, o aplicativo exibe uma mensagem simples sem HTML. A maioria dos aplicativos da Web usa HTML para exibir informações aos visitantes. Dessa forma, você pode trabalhar na vinculação de arquivos HTML ao seu aplicativo, que pode ser visualizado em seu navegador da web.

O Flask fornece um utilitário render_template() que permite o uso do mecanismo de modelagem do Jinja. Isso facilitará o gerenciamento de HTML quando você escrever seu código HTML em arquivos .html e quando usar lógica em seu código HTML. Você usa esses arquivos HTML, ( ) para criar todas as páginas do seu aplicativo, como a página inicial onde você exibe postagens, páginas onde o usuário pode adicionar formulários, etc.

Para dar continuidade podemos excluir o código anterior e adicionar o novo código importando também o render_template.

Nesse novo arquivo, você importará o objeto Flask para criar uma instância do aplicativo Flask, exatamente como fez antes. Você importará a função auxiliar render_template(), que permite gerar seus arquivos de modelo em HTML na pasta de modelos que deseja criar. O arquivo conterá uma única função de verificação, que será responsável por processar a solicitação do caminho/ponto. Adicione o conteúdo ilustrado na figura 13 a seguir:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')

def index():

return render_template('index.html')
```

Figura 13 – Exemplo de rotina para inserir html através do framework Flask.
Fonte: FLASK, 2022.

A função index() retorna o resultado de render_template() com index.html como argumento. Isso diz a render_template() para procurar um arquivo chamado index.html na pasta de templates.

Para dar continuidade, foi criado uma pasta chamada templates na pasta raiz do nosso software, após isso criamos um arquivo HTML com o nome index.html. Depois da criação do arquivo .html foi adicionado conteúdo ilustrado na figura 14 a seguir:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>FlaskBlog</title>

</head>

<body>

   <h1>Welcome to FlaskBlog</h1>

</body>

</html>
```

Figura 14 – Modelo de conteúdo HTML que pode ser inserido com Flask.
Fonte: FLASK, 2022.

Salve o arquivo e use seu navegador para acessar http://127.0.0.1:5000/ novamente ou atualize a página. Neste ponto, o navegador deve exibir o texto ‘Welcome to FlaskBlog’ na tag <h1>. Além da pasta de modelos, os aplicativos da Web geralmente têm pastas estáticas para hospedar arquivos como arquivos CSS, arquivos JavaScript e imagens que o aplicativo usa.

Para a estilização do nosso software utilizaremos o Bootstrap e alguns elementos de CSS puro.

Bootstrap é um framework web com código-fonte aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web. Além do HTML e CSS, o Bootstrap inclui suporte ao JavaScript, extensões de LESS e, opcionalmente, ao jQuery. Desenvolvido na Twitter e disponibilizado publicamente como software livre no GitHub, o Bootstrap é usado como base para mais de 2 milhões de sites (Bootstrap, 2022).

Iniciado pelo Twitter em 2010, o Bootstrap foi criado para aumentar a consistência nas ferramentas usadas internamente e para melhorar o processo de desenvolvimento. Como o Twitter cresceu, o Bootstrap cresceu com ele. Agora o Bootstrap, segundo sua documentação, é o projeto de interface de usuário mais popular do GitHub e foi o primeiro projeto de código aberto (Bootstrap, 2022).

Foi utilizado Bootstrap em toda a estilização do projeto, importado o estilo através de classes e com isso nosso sistema começou a tomar forma. O principal motivo para a escolha do uso desse framework foi dado a versatilidade e a responsividade entremeada no uso das classes, com o Bootstrap o sistema já fica responsivo automaticamente facilitando o desenvolvimento e mantendo um padrão de design.

Após a implementação do Bootstrap iniciamos a criação de novas telas, criação, edição e visualização dos Tickets, com isso foi pensado em utilizar a ferramenta de herança do Flask, mais uma maneira de economizar linhas e otimizar o desenvolvimento, para a implementação da herança no nosso sistema foi criado um arquivo .html com o nome base.html onde foi feito a base que seria utilizado em todas as telas do sistema.

{% block title %} {% endblock %} você o usará posteriormente em outros modelos para dar um nome personalizado para cada página em seu aplicativo, sem reescrever toda a seção todas as vezes.

{{ url_for ('index')}}: uma chamada de função que retornará uma URL para a função display index(). Isso é diferente da chamada url_for() anterior que você usou para vincular um arquivo CSS estático porque leva apenas um argumento, que é o nome da função de exibição eo link e o caminho associado a essa função, em vez de um arquivo estático.

{% block content %} {% endblock %} : outro bloco será substituído por conteúdo, dependendo de qual modelo filho (os padrões podem ser herdados do arquivo base.html) o substituirá.

4.4. CONFIGURANDO O BANCO DE DADOS

Para o banco de dados utilizamos o SQLite que foi o banco de dados que utilizamos em alguns projetos de Python durante o curso. SQLite é fácil de configurar e está disponível na biblioteca padrão do Python.

Para criar nosso banco de dados foi criado um arquivo chamado schema.sql onde fica o create (ilustrado na figura 15) das tabelas utilizado no projeto.

Para armazenar os tickets foi criado uma tabela posts onde ficará armazenado todas as informações contidas nos tickets:

```
DROP TABLE IF EXISTS posts;
CREATE TABLE posts (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  title TEXT NOT NULL,
  content TEXT NOT NULL,
  conclusao TEXT NOT NULL,
  departamentos TEXT NOT NULL,
  status TEXT NOT NULL,
  anexos TEXT NOT NULL);
```

Figura 15 – Rotina para criação das tabelas utilizadas na aplicação com SQLite.
Fonte: FLASK, 2022.

A primeira instrução SQL é ‘DROP TABLE IF EXISTS POSTS;’, que remove qualquer tabela existente com nomes de postagens para evitar comportamentos confusos.

Para criar o banco de dados usando um arquivo Python que irá gerar um arquivo de banco de dados .db. Abra um arquivo chamado init_db.py dentro do diretório e insira o código ilustrado na figura 16 a seguir:

```
import sqlite3

connection = sqlite3.connect('database.db')
with open('schema.sql') as f:
    connection.executescript(f.read())

cur = connection.cursor()

cur.execute("INSERT INTO posts (title, content, conclusao, departamentos, status,
anexos) VALUES (?, ?, ?, ?, ?, ?)",
            ('teste', 'teste', "21/25/4241",
             "administrativo", "pendente", "teste.jpg")
            )
connection.commit()
connection.close()
```

Figura 16 – Rotina para executar o script sql, criando então o banco de dados.
Fonte: SQLITE CONSORTIUM, 2022.

Assim que o arquivo for concluído, um novo arquivo chamado database.db aparecerá em seu diretório.

Com o banco criado e informações inseridas nele começamos a desenvolver as telas.

Criando o Index ilustrado na figura 17 irá exibir os títulos na página principal do nosso sistema.

```
{% extends 'base.html' %}


{% block content %}
<h1 id="titulo">{% block title %} Sistema de Solicitaçoes de Tickets Internos {%
endblock %}</h1>
{% for post in posts %}
<a href="{{ url_for('post', post_id=post['id']) }}">
    <h2>{{post['id']}}- {{ post['title'] }} </h2>
</a>
<label for="">Departamento: <span class="badge badge-
primary">{{post['departamentos']}}</span></label>
<br>
<label >Status: <span class="{{post['Status']}}">{{post['Status']}}</span></label>
<br>
<span class="badge badge-info">{{ post['created'] }}</span>
<a href="{{ url_for('edit', id=post['id']) }}">
    <span class="badge badge-secondary">Editar</span>
</a>
<hr>

{% endfor %}
{% endblock %}
</div>
```

Figura 17 – Html da tela inicial da aplicação.

Fonte: JINJA, 2022.

A sintaxe {% for post in posts %} conforme descrito na documentação do Jinja 2 é o loop for do de Jinja, que é semelhante ao loop for do Python, exceto que deve ser finalizado com a sintaxe {% endfor %} (Jinja, 2022).

Com a base do HTML pronta, foi implementado a rota (ilustrada na figura 18) para exibir as informações corretas no index.

```
@app.route('/', methods=('GET', 'POST'))
def index():



    conn = get_db_connection()
    posts = conn.execute('SELECT * FROM posts').fetchall()
    conn.close()

    return render_template('index.html', posts=posts)
```

Figura 18 – Rotina para implementação de rotas adotada na aplicação.

Fonte: FLASK, 2022.

4.5. TELA DE VISUALIZAÇÃO DE TICKET

Com a tela principal feita criamos uma tela onde mostra detalhadamente as informações contidas nos tickets, você tem a possibilidade de visualizar as imagens vinculadas, data de conclusão e outras informações.

A exibição da imagem é feita através de uma tag <img> que pega o caminho da imagem que você salvou e é renderizado na tela.

Nessa tela foi implementado além do label para mostrar as informações, um Modal para facilitar a visualização da imagem para este modal foi utilizado JavaScript e o próprio modal disponível na documentação do Bootstrap, com leves adaptações para melhor atender nosso sistema.

A figura 19 a seguir ilustra a forma que foi desenvolvido a tela de visualização:

```
<div id="imagens">
    <div>
        <label><b> Imagem:</b></label>
        <a href="#" parametro="imagem1.jpg" class="abrirModal"><img
src="/static/images/{{ post['anexos'] }}"></a>
        <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myLargeModalLabel">
          <div class="modal-dialog modal-lg" role="document">
            <div class="modal-content">
              <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
              </div>
              <div class="modal-body text-center">
                <img src="" style="width: 100%;" />
              </div>
            </div>
          </div>
        </div>
    </div>
</div>
<hr>
<label for="title"><b>Descrição:</b></label>
<p>{{ post['content'] }}</p>
<hr />
<script>$(".abrirModal").click(function() {
    var url = $(this).find("img").attr("src");
    $("#myModal img").attr("src", url);
    $("#myModal").modal("show");
  });</script>
```

Figura 19 – Rotina para criação do modal com as classes do Bootstrap.

Fonte: BOOTSTRAP, 2022.

Com o front-end finalizado foi implementado uma rota para exibir as informações vinculadas a cada ticket ilustrado na figura 20 a seguir:

```
@app.route('/<int:post_id>')
def post(post_id):
    post = get_post(post_id)
    return render_template('post.html', post=post)
```

Figura 20 – Rotina para criação da rota de cada ticket.

Fonte: FLASK, 2022.

Com a rota pronta a nossa função de visualização dos Tickets está pronta e funcional.

4.6. TELA DE INCLUSÃO DE TICKETS

Essa tela é um formulário onde é inserido as informações em input e temos o botão enviar que faz o envio do formulário em forma de requisição para o banco e salvando as informações. Nessa parte do projeto foi utilizado label e input para inserir os dados, porém no campo data de conclusão foi inserido JavaScript para inserir as "/" da data automaticamente ilustrado na figura 21 a seguir:

```
        <input class="form-control" type="text" name="conclusao"
placeholder="dd/mm/yyyy" onkeyup= "
            var v = this.value;
        if (v.match(/^\d{2}$/) !== null) {
            this.value = v + '/';
        } else if (v.match(/^\d{2}\/\d{2}$/) !== null) {
            this.value = v + '/';
        }" maxlength="10" value="{{ request.form['conclusao'] }}"></input>
```

Figura 21 – Utilização de REGEX com JavaScript para inclusão do caractere "/" a cada 2 dígitos.

Fonte: MOZILLA FOUNDATION, 2022.

Com o front-end finalizado foi feito a rota, responsável pela criação do ticket ilustrado na figura 22 a seguir:

```
@app.route('/create', methods=('GET', 'POST'))
def create():
    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']
        conclusao = request.form['conclusao']
        departamentos = request.form['departamentos']
        status = request.form['status']
        anexos = request.form['anexos']

        if not title:
            flash('É necessário ter um título!')
        else:
            conn = get_db_connection()
            conn.execute('INSERT INTO posts (title, content, conclusao, departamentos,
status, anexos) VALUES (?, ?, ?, ?, ?, ?)',
                         (title, content, conclusao, departamentos, status, anexos))
            conn.commit()
            conn.close()
            return redirect(url_for('index'))

    return render_template('create.html')
```

Figura 22 – Rotina para criação da rota de criação de tickets.

Fonte: PYTHON SOFTWARE FOUNDATION, 2022.

Nessa rota temos um tratamento onde é obrigatório que o ticket tenha um título, caso não tenha ele irá exibir uma mensagem através do objeto Flask, caso esteja tudo correto ele irá inserir as informações no banco.

Com isso nossa tela de criação de tickets está pronta.

4.7. TELA DE EDIÇÃO DE TICKETS

Pensando que os tickets podem mudar de status, sofrer correções entre outras coisas pensamos em adiciona uma tela de edição do ticket, nela também terá a opção de deletar o ticket.

Para fazer essa tela do programa, primeiramente foi desenvolvido o front-end essa parte foi aproveitado basicamente tudo da tela de criação porem com um diferencial o input de deletar o ticket e a rota de editar e deletar o ticket.

Para o front-end o botão é um input do tipo submit(ilustrado na figura 23) onde ao clicar aparece uma imagem perguntando se deseja realmente deletar o ticket.

```
<form action="{{ url_for('delete', id=post['id']) }}" method="POST">
   <input type="submit" value="Deletar"
          class="btn btn-danger btn-sm"
          onclick="return confirm('Você tem certeza que deseja deletar seu Ticket?')">
</form>
```

Figura 23 – Rotina para criação de botão de deleção com confirmação de certeza.
Fonte: MOZILLA FOUNDATION, 2022.

Conforme descrito na documentação do JavaScript o método confirm() serve para exibir uma mensagem antes de enviar a solicitação para back-end.

Com o nosso botão de deletar feito, podemos dar início nas rotas de editar e deletar o Ticket.

Para a função de deletar o ticket foi implementado a rota delete onde ele deleta através do ID do Ticket, o código da rota é ilustrado na figura 24 a seguir:

```
@app.route('/<int:id>/delete', methods=('POST',))
def delete(id):
    post = get_post(id)
    conn = get_db_connection()
    conn.execute('DELETE FROM posts WHERE id = ?', (id,))
    conn.commit()
    conn.close()
    flash('"{}" deletado com sucesso'.format(post['title']))
    return redirect(url_for('index'))
```

Figura 24 – Rotina para deleção de um ticket por ID.

Fonte: PYTHON SOFTWARE FOUNDATION, 2022.

Nossa rota de edit é muito similar a rota de criação com algumas mudanças como o script para atualizar as informações, o código da rota é ilustrado na figura 25 a seguir:

```
@app.route('/<int:id>/edit', methods=('GET', 'POST'))
def edit(id):
    post = get_post(id)
    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']
        conclusao = request.form['conclusao']
        departamentos = request.form['departamentos']
        status = request.form['status']
        anexos = request.form['anexos']
        if not title:
            flash('É necessário ter um título!')
        else:
            conn = get_db_connection()
            conn.execute('UPDATE posts SET title = ?, content = ?, conclusao = ?, departamentos = ?, status = ?, anexos = ?'
                         ' WHERE id = ?',
                         (title, content, conclusao, departamentos, status, anexos, id))
            conn.commit()
            conn.close()
            return redirect(url_for('index'))
    return render_template('edit.html', post=post)
```

Figura 25 – Rotina para Edição do ticket.

Fonte: Fonte: PYTHON SOFTWARE FOUNDATION, 2022.

4.8. IMPLEMENTAÇÕES ADICIONAIS

Após a base do sistema feito, fizemos duas implementações, a cor do status do ticket e dos departamentos vinculados ao ticket, também foi implementado uma barra de busca na tela inicial para auxiliar a achar os tickets desejados.

A barra de pesquisa foi utilizada a barra de busca do Bootstrap conforme ilustrado na figura 26, com certas adaptações para funcionar junto ao Flask.

```
<div>
    <form class="search-box" method="POST">
      <input class="search-box" type="busca" value="{{request.form['buscar']}}"
placeholder="Pesquisar por..."
        name="buscar">
      <button class="btn btn-outline-sucess my-2 my-sm-0"
type="submit">Buscar</button>

   </div>
```

Figura 26 – Implementação da barra de busca com Bootstrap.

Fonte: BOOTSTRAP, 2022.

A função responsável pela busca no banco e exibição na tela implementada dentro da roda do index como uma condicional, ilustrado na figura 27 a seguir:

```
if request.method == 'POST':
      buscar = request.form['buscar']
      conn = get_db_connection()
      print(buscar)
      posts = conn.execute(
         "SELECT * FROM posts WHERE status=? or id=? or departamentos=? or
title=?", (buscar, buscar, buscar, buscar,)).fetchall()
      conn.close()
      return render_template('index.html', posts=posts)
```

Figura 27 – Função de realização de busca de tickets.

Fonte: PYTHON SOFTWARE FOUNDATION, 2022.

A mudança de cor é feita dinamicamente caso mude a situação do ticket, para fazer essas mudanças foi criado uma classe com o nome de cada situação e a estilização para cada classe cada uma com sua cor e dentro do html onde é denominado a classe é passado o nome salvo no banco ilustrado na figura 28, dessa forma é muda a cor da situação.

```
<label >Status: <span class="{{post['Status']}}">{{post['Status']}}</span></label>
```

Figura 28 – Rotina para criação de label de demonstração de status.

Fonte: JINJA, 2022.

Dessa forma foi finalizado o desenvolvimento do nosso sistema, agora, totalmente funcional, com os principais campos e funcionalidades.

5. CONCLUSÃO

O software produzido cumpre seu objetivo de trazer a solução para problemas de gestão de tickets de suporte dentro de uma empresa, facilitando o dia a dia do usuário final, sempre mantendo sua simplicidade e aparência de entendimento fácil, que é requisito básico para que seja uma experiência tranquila a sua utilização e simples de entender pelo perfil de persona previsto como nosso usuário.

6. MELHORIAS FUTURAS

O software produzido cumpre seu objetivo de trazer a solução para problemas de gestão de tickets de suporte dentro de uma empresa, facilitando o dia a dia do usuário final, sempre mantendo sua simplicidade e aparência de entendimento fácil, que é

Após todas as etapas de planejamento e desenvolvimento do software, podemos perceber que há muito mais envolvido no desenvolvimento de um produto de tecnologia do que se percebe a primeira vista, atividades que não estão diretamente relacionadas ao código em si ou a programação, mas que demandam boa parte do trabalho dos envolvidos para desenvolver uma boa aplicação WEB é de planejamento de requisitos, mapeamento de necessidade, trabalho em equipe e principalmente testes.

No decorrer do desenvolvimento do projeto pensamos em certas melhorias que podem ser feitas futuramente:

- Cadastro dinâmico de departamentos onde terá uma tela que pode ser cadastrados novos departamentos.
- Cadastro de usuários e denominar permissões, onde somente responsável pelo ticket pode alterar status e certas informações.
- Tela de login ao sistema.
- Filtros para facilitar a pesquisa
- Relatórios com gráficos para o gerenciamento.

REFERÊNCIAS

ALVES, R. A.; MARTINS, R. C.; PAULISTA, P. H. **OPERAÇÃO E MANUTENÇÃO DE SOFTWARE:** UMA ABORDAGEM TEÓRICA. XX Encontro Latino Americano de Iniciação Científica, XVI Encontro Latino Americano de Pós-Graduação e VI Encontro de Iniciação à Docência. São José dos Campos: Universidade do Vale do Paraíba. 2016. p. 4.

BOOTSTRAP. **Bootstrap**, 2022. Disponivel em: <https://getbootstrap.com>. Acesso em: 1 Novembro 2022.

DIGITÉ. What Is Scrum Methodology? & Scrum Project Management. **digité**, 2022. Disponivel em: <https://www.digite.com/agile/scrum-methodology/>. Acesso em: 20 Setembro 2022.

DILKIN, D. A importância de fazer o levantamento de requisitos para um sistema. **VVerner**, 2020. Disponivel em: <https://vverner.com/a-importancia-de-fazer-o-levantamento-de-requisitos-para-um-sistema/>. Acesso em: 22 Outubro 2022.

FADEL, A. C.; SILVEIRA, H. D. M. **Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean**. UNICAMP – Universidade Estadual de Campinas. Limeira, p. 26. 2010.

FERNANDES, J. H. C. Qual a prática do desenvolvimento de software? **Cienc. Cult.**, Abril 2003. ISSN 2317-6660. Disponivel em: <http://cienciaecultura.bvs.br/scielo.php?pid=S0009-67252003000200021&script=sci_arttext>. Acesso em: 12 Novembro 2022.

FLASK. **Flask**, 2022. Disponivel em: <https://flask.palletsprojects.com/en/1.1.x/installation/>. Acesso em: 23 Outubro 2022.

GONÇALVES, J. E. L. Processo, que processo? **RAE - Revista de Administração de Empresas**, São Paulo, 40, n. 4, Outubro 2000. 8-19. Disponivel em: <http://www.producao.ufrgs.br/arquivos/disciplinas/493_goncalves_2000b.pdf>. Acesso em: 29 Outubro 2022.

GOOGLE. Requisitos do sistema do navegador Chrome. **Google**, 2022. Disponivel em: <https://support.google.com/chrome/a/answer/7100626?hl=pt-BR>. Acesso em: 03 Novembro 2022.

HARRINGTON, H. J. **Aperfeiçoando Processos Empresariais**. São Paulo: Makron Books, 1993. 368 p.

IGNACZUK, C. Vantagens, funcionalidades e como escolher um sistema de tickets de suporte. **MOVIDESK**, 1 Setembro 2020. Disponivel em: <https://conteudo.movidesk.com/sistema-de-tickets-suporte/#dos-beneficios-aos-recursos-tudo-o-que-voce-queria-saber-sobre-sistemas-de-atendimento>. Acesso em: 1 Novembro 2022.

JINJA. **Jinja**, 2022. Disponivel em: <https://jinja.palletsprojects.com/en/3.1.x/>. Acesso em: 01 Novembro 2022.

LEMOS, M. F. D. et al. **APLICABILIDADE DA ARQUITETURA MVC EM UMA APLICAÇÃO WEB (WebApps)**. UNIFENAS. Alfenas, p. 17. 2013.

LOPES, E. H. T. **ESTUDO SOBRE A IMPORTÂNCIA DA MANUTENÇÃO DE SOFTWARE**. FACULDADES INTEGRADAS DE CARATINGA. Caratinga, p. 73. 2017.

LUCIANO, J.; ALVES, W. J. B. **PADRÃO DE ARQUITETURA MVC: MODEL-VIEW-CONTROLLER**. Centro Universitário UNIFAFIBE. Bebedouro, p. 6. 2011.

MICROSOFT. Getting Started. **Visual Studio Code**, 2022. Disponivel em: <https://code.visualstudio.com/docs>. Acesso em: 27 Outubro 2022.

MIRANDA, G. H.; ALVES, E. J. **MELHORES PRÁTICAS EM MÉTODOS ÁGEIS APLICADOS NO SUPORTE E MANUTENÇÃO DE SOFTWARE:** UMA REVISÃO SISTEMÁTICA DA LITERATURA. VII Simpósio Internacional de Gestão de Projetos, Inovação e Sustentabilidade (VII SINGEP). São Paulo: [s.n.]. 2018. p. 17.

MOZILLA FOUNDATION. JavaScript. **mdn web docs**, 2022. Disponivel em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 25 Outubro 2022.

PADUELLI, M. M. **Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica**. USP - Universidade de São Paulo. São Carlos, p. 155. 2007.

PEÇANHA, V. Descubra o que é buyer persona e quais os 5 passos essenciais para criar a sua. **Rock Content**, 4 Junho 2020. Disponivel em: <https://rockcontent.com/br/blog/personas/>. Acesso em: 10 Novembro 2022.

PYTHON SOFTWARE FOUNDATION. O tutorial de Python. **Python**, 2022. Disponivel em: <https://docs.python.org/pt-br/3/tutorial/venv.html>. Acesso em: 27 Novembro 2022.

SANTOS, A. K. D. **Os IDE's (Ambientes de Desenvolvimento Integrado) como ferramentas de trabalho em informática**. Universidade Federal de Santa Maria. São Carlos, p. 10. 2007.

SQLITE CONSORTIUM. Documentation. **SQLite**, 2022. Disponivel em: <https://www.sqlite.org/docs.html>. Acesso em: 25 Outubro 2022.

SVIRCA, Z. Everything you need to know about MVC architecture. **Towards Data Science**, 29 Maio 2020. Disponivel em: <https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1>. Acesso em: 16 Outubro 2022.

UFPE. Questões de Concursos. **QConcursos**, 2010. Disponivel em: <https://www.qconcursos.com/questoes-de-concursos/questoes/aef9dfc8-1d>. Acesso em: 29 Outubro 2022.

VARGAS, A. A. F.; PEREIRA, J. V. D. S. **GERENCIAMENTO DA MANUTENÇÃO DE SOFTWARE**. [S.l.], p. 14. 2019. (2358-646x).

VARVAKIS, G. et al. **Gerenciamento de Processos**. Universidade Federal de Santa Catarina. Florianópolis, p. 103. 2018.