

RELATÓRIO  
PROJETO  
APLICADO  
PÓS-GRADUAÇÃO

**XP Educação**  
**Relatório do Projeto Aplicado**

**Sistema de Gestão de *Tickets* de  
Atendimento**

Guilherme Fidelis da silva

Orientador(a): Reinaldo Galvão

Outubro de 2024



**GUILHERME FIDELIS DA SILVA**

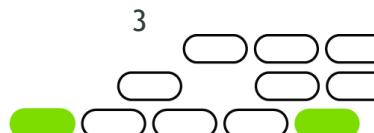
**XP EDUCAÇÃO**

**RELATÓRIO DO PROJETO APLICADO**

# **Sistema de Gestão de Tickets de Atendimento**

Relatório de Projeto Aplicado desenvolvido para fins de conclusão do curso Arquitetura de Software e Soluções.

Orientador (a): Reinaldo Galvão



## Sumário

1. CANVAS do Projeto Aplicado .....	6
1.1 Desafio .....	7
1.1.1 Análise de Contexto.....	7
1.1.2 Personas .....	10
1.1.3 Benefícios e Justificativas.....	14
1.2 Solução .....	18
1.2.1 Objetivo SMART.....	18
1.2.2 Premissas e Restrições.....	19
1.2.3 Backlog de Produto .....	21
2. Área de Experimentação.....	22
2.1 Sprint 1.....	22
2.1.1 Solução .....	22
• Evidência do planejamento: .....	22
• Evidência da execução de cada requisito: Coleta de requisitos .....	22
• Evidência dos resultados: Coleta de requisitos .....	23
• Evidência da execução de cada requisito: Visão geral do produto.....	35
• Evidência dos resultados: Visão geral do produto .....	35
2.1.2 Lições Aprendidas .....	45
2.2 Sprint 2.....	47
2.2.1 Solução .....	47
• Evidência do planejamento: .....	47
• Evidência da execução de cada requisito: Elaborar Casos de Uso.....	47
• Evidência dos resultados: Elaborar Casos de Uso .....	48
Casos de Uso .....	48
Diagramas de Caso de Uso .....	55
• Evidência da execução de cada requisito: Modelar Base de Dados .....	59
• Evidência dos resultados: Modelar Base de Dados .....	60
Tabela de Usuários .....	61
Tabela de Demandas.....	62
Tabela de Transferência de Demandas.....	64
Tabela de Relatórios .....	64
Evidências concretas.....	65
2.2.2 Lições Aprendidas .....	71
2.3 Sprint 3.....	73
2.3.1 Solução .....	73



● Evidência do planejamento: .....	73
● Evidência da execução de cada requisito: Elaborar plano de infraestrutura.....	73
● Evidência dos resultados: Elaborar plano de infraestrutura.....	73
● Evidência da execução de cada requisito: Elaborar DAS.....	77
● Evidência dos resultados: Elaborar DAS .....	77
2.3.2 Lições Aprendidas .....	79
<b>3. Considerações Finais.....</b>	<b>80</b>
3.1 Resultados .....	80
3.2 Contribuições.....	82
3.3 Próximos passos.....	83



## 1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado.

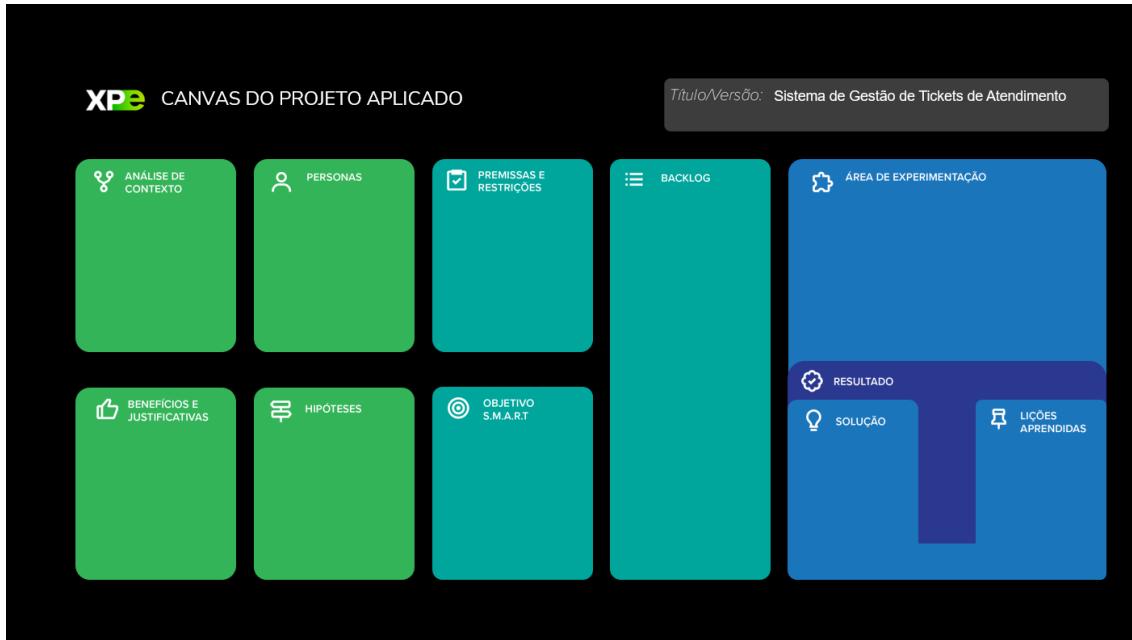
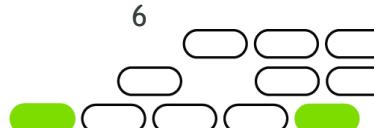


Figura. Canvas do Projeto Aplicado. Preenchido somente o título do trabalho. Serve como baseline para o desenvolvimento do Projeto Aplicado. Escolheu-se não preencher devido ao pouco espaço disponível para cada item.



## 1.1 Desafio

### 1.1.1 Análise de Contexto

A Papéis Miguel Escobar é uma pequena empresa que atua na distribuição de produtos de papel para diferentes fins. Atualmente possui dois sócios e um gerente na matriz. A equipe de vendas é composta por 5 funcionários, sendo um responsável para trabalho de campo, a contabilidade conta com 2 colaboradores, há 2 funcionários no RH e 5 responsáveis pelo armazém, almoxarifado, patrimônio e logística. O setor de TI conta com 2 colaboradores, e há alguns colaboradores que realizam serviços eventuais na modalidade *freelance*.

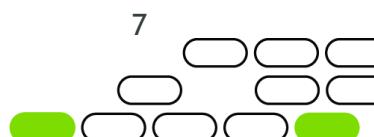
A diretoria da Papéis Miguel Escobar recebeu do setor de RH um pedido para avaliar as recorrentes reclamações dos funcionários referentes a falhas de comunicação e execução de algumas demandas na empresa. Muitas solicitações entre setores acabam por não serem atendidas, e há frequentes acusações de “perseguição” e “descaso” por conta de demandas ignoradas.

#### Quais são as verdadeiras causas do problema?

Por meio de diálogo com os funcionários dos setores, foi possível averiguar que os procedimentos efetuados pelos funcionários referentes às solicitações não possuem qualquer padronização. Atualmente, todas as solicitações são realizadas de modo caótico, seja diretamente pelo aplicativo de mensagens, seja por e-mail, seja por telefone para o ramal ou se deslocando diretamente ao setor em questão.

Isto acaba por atrasar muito o trabalho, pois devido à descentralização das solicitações, não é possível ter controle nem definir prioridades no cumprimento das tarefas e atendimento das demandas. Além disso, a falta de organização causada pelas diversas formas de se solicitar demandas resulta em várias delas sendo ignoradas, já que se perdem no meio de mensagens, e-mails e notas em papel que acabam se extraviando.

Visando melhorar a comunicação, organização, planejamento e controle, a Papéis Miguel Escobar optou por bancar a implementação de um sistema de gestão de *tickets* de atendimento. Ao invés dos funcionários utilizarem diversos meios para solicitarem serviços entre os setores, estas estariam centralizadas num sistema único de gestão de chamados. Um colaborador que necessite ajuste no ponto de frequência por conta de um atestado médico não mais enviará e-mail para o RH com o atestado digitalizado.

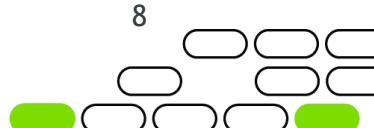


Ele acessaria o sistema de gestão de *tickets* de atendimento e abriria uma solicitação direcionada ao RH, anexando o atestado ao sistema. De forma semelhante, um outro funcionário que precisasse que um aplicativo específico fosse instalado no computador que utiliza para o trabalho utilizaria o mesmo sistema, porém direcionando a requisição ao setor de tecnologia da informação.

A expectativa é que com o uso de um sistema de gestão de *tickets* de atendimento os procedimentos dentro da empresa sejam melhorados, assegurando todos os passos de um atendimento eficaz, desde o cadastro da demanda, atendimento e avaliação da solução oferecida pelo atendente.

**Como as pesquisas foram conduzidas? Existem registros e evidências?**

Por se tratar de uma empresa de pequeno porte, foi possível obter informações a partir de diálogos informais com os funcionários de cada setor, juntamente com os registros de reclamações e sugestões do setor de Recursos Humanos. Com base nestas informações, foram elaborados alguns artefatos para auxiliar o desenvolvimento do projeto.



## MATRIZ CSD

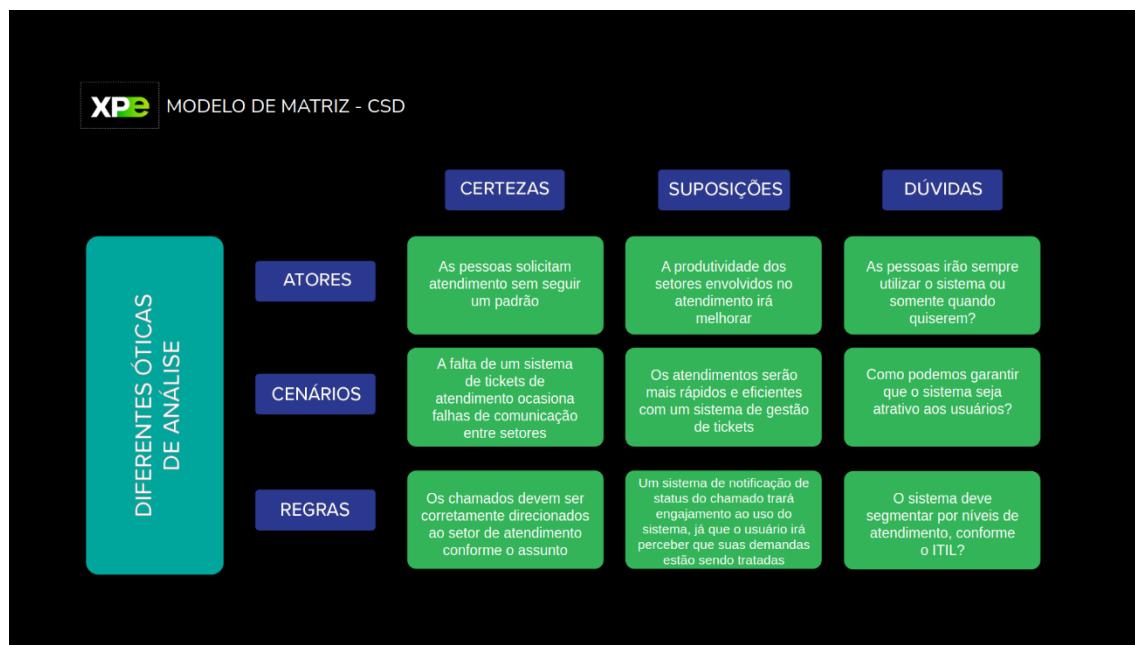
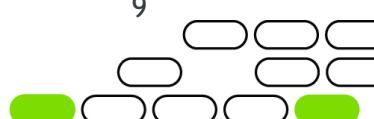


Figura. Matrix CSD

## OBSERVAÇÃO POEMS



Figura. Observações POEMS



## 1.1.2 Personas



**Duarte Chucrute**

**Idade:** 34 anos

**Cargo:** Vendedor

Duarte é um dos vendedores da Papéis Miguel Escobar. *Workaholic*, metódico, extremamente organizado e controlador. É muito ambicioso e almeja se tornar gerente de vendas. Gosta do ambiente competitivo que o setor de vendas proporciona, definindo como alvo sempre superar os colegas e a si mesmo.

Trabalha na Papéis Miguel Escobar há 10 anos, e é totalmente leal à empresa, mas tem plena noção de que suas habilidades de vendas seriam apreciadas em qualquer outra empresa do ramo.

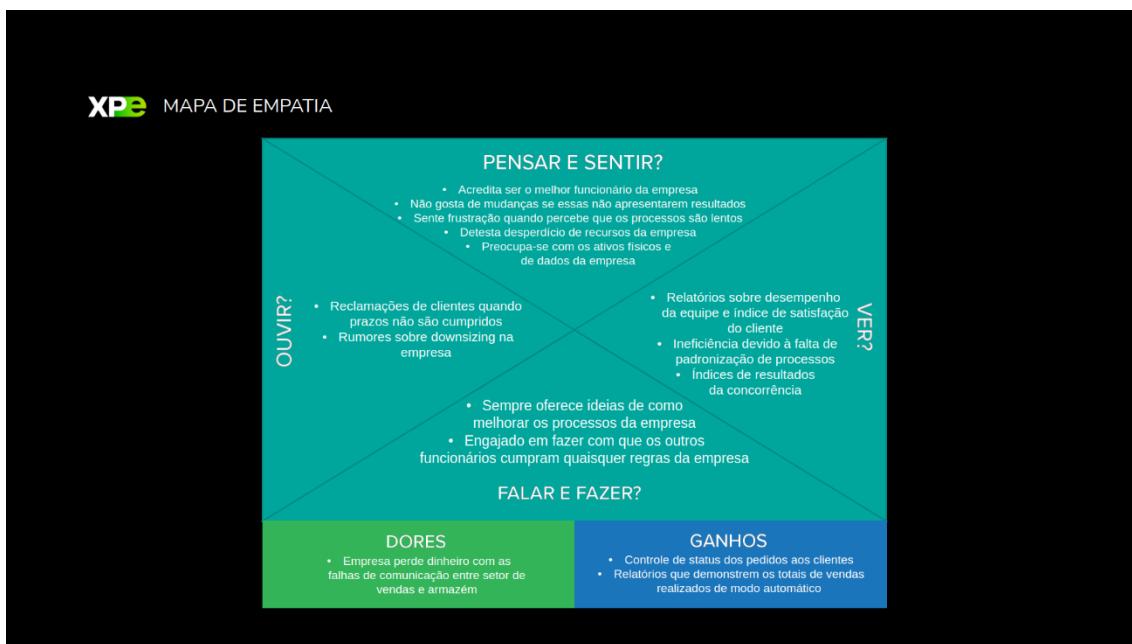


Figura. Mapa de Empatia de Duarte Chucrute.



**Oscar Martines****Idade:** 35 anos**Cargo:** Contador

Oscar trabalha no setor de contabilidade da empresa. É um sujeito tranquilo e preza por um trabalho realizado de modo preciso e extremamente detalhista. Odeia que utilizem atalhos para as tarefas, pois isso pode acarretar imprecisões nos dados contábeis. Cauteloso por natureza, conhece a proporção que pequenos erros podem acarretar.

Bastante inteligente e culto, possui um certo ceticismo quanto à adoção de novas tecnologias, mas sabe apreciar quando os resultados são positivos. Trabalha na Papéis Miguel Escobar há 8 anos e tem conhecimento total dos processos contábeis da empresa.

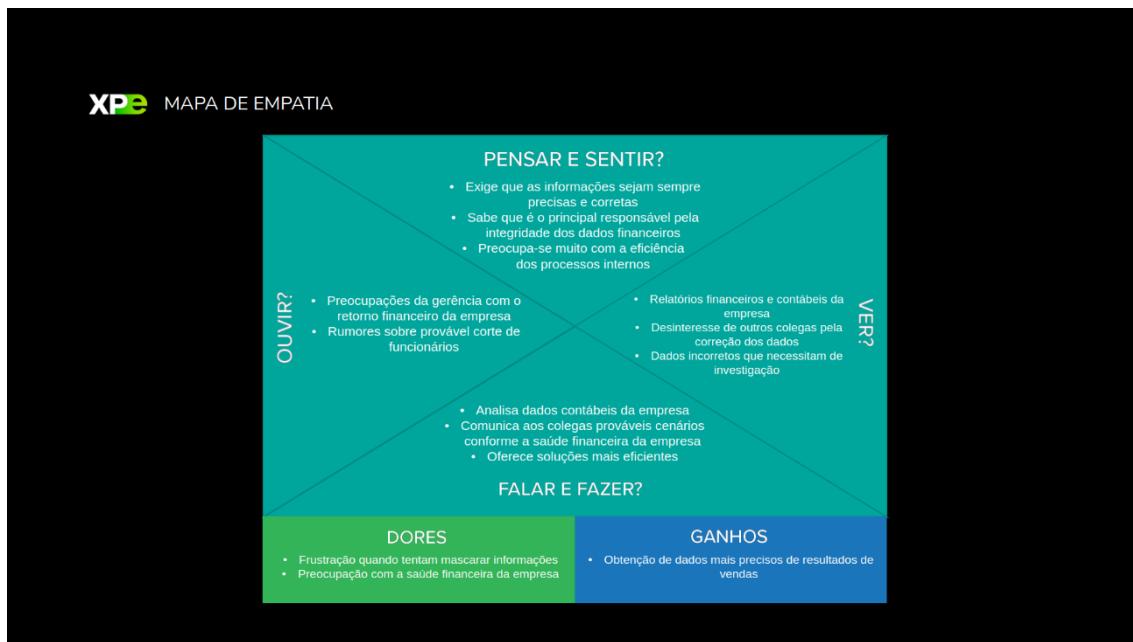


Figura. Mapa de empatia de Oscar Martines.





**Dário Felipe**

**Idade:** 40 anos

**Cargo:** Gerente de armazém e almoxarifado

Dario trabalha no almoxarifado, patrimônio, armazém e logística da empresa. É um funcionário competente, e prefere que as tarefas sejam realizadas de modo inteligente e simples. Não gosta de processos burocráticos, pois tomam muito tempo e aparecem ser ineficazes. Exerce uma liderança natural no setor, e todos os funcionários da empresa o respeitam bastante. Sabe ser direto sem parecer rude. Consegue se adaptar facilmente a novos métodos e sistemas, e almeja por maior reconhecimento do serviço realizado no armazém.

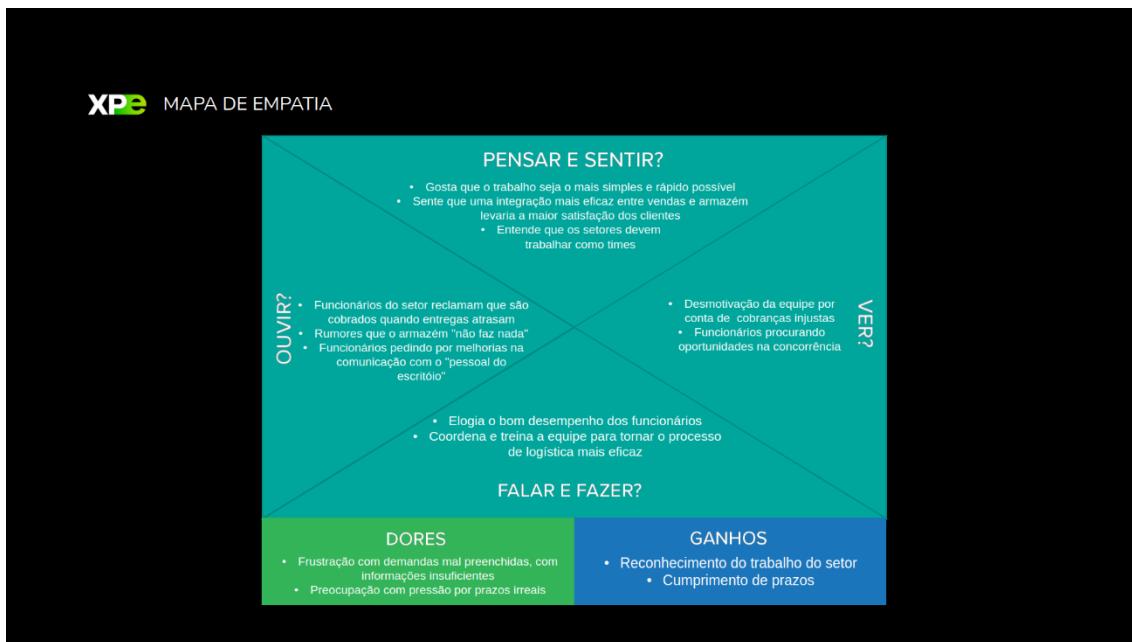


Figura. Mapa de empatia de Dário Felipe.





**Gabriel Luiz**

**Idade:** 28 anos

**Cargo:** Analista de Suporte Técnico

Gabriel trabalha no setor de Tecnologia da Informação da Papéis Miguel Escobar e, além de fazer parte da equipe de desenvolvimento do sistema, será usuário dele, pois uma de suas atribuições é agir no suporte técnico ao usuário na empresa de Papel Miguel Escobar.

É uma pessoa ambiciosa, bastante confiante de suas habilidades técnicas, porém inseguro quando fora de sua zona de conforto.

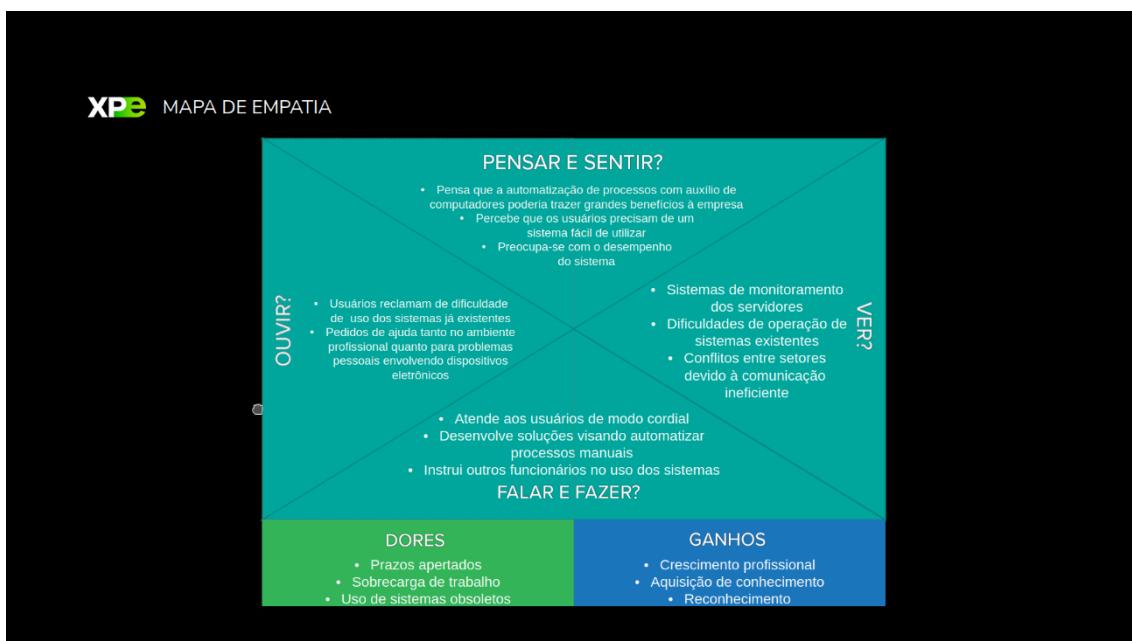
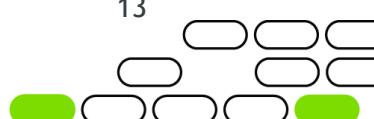


Figura. Mapa de Empatia de Gabriel Luiz.



### 1.1.3 Benefícios e Justificativas

A implantação de um sistema de gestão de *tickets* de atendimento na Papéis Miguel Escobar irá trazer algumas melhorias bastante sensíveis, otimizando e formalizando os processos envolvendo solicitações entre os setores e oferecendo métricas adicionais para avaliar o desempenho dos membros da equipe.

Além disso, com a obtenção de dados detalhados das principais demandas entre os setores, será possível direcionar melhor os recursos da empresa de modo a aliviar gargalos em setores sobrecarregados, bem como reavaliar a necessidade de ajustar o tamanho de setores menos demandados. Estes ajustes a longo prazo devem otimizar a lucratividade da empresa, reduzindo custos e incrementando a produtividade.

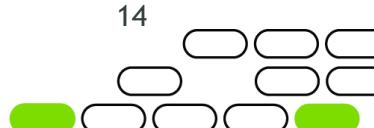
Como um dos principais problemas enfrentados pela Papéis Miguel Escobar são as diversas falhas de comunicação entre os setores, resultando em ineficiência e perda de prazos, o sistema proposto visa resolver gradualmente esta situação. Além disso, a análise dos relatórios pode oferecer *insights* valiosos, permitindo identificar alguns padrões e tendências e criar estratégias de prevenção de problemas recorrentes.

As vantagens não se limitam a processos internos. Melhor eficiência nos processos internos acarreta menores prazos de entrega de produtos a clientes, e redução de eventuais atrasos. Isto pode melhorar significativamente a imagem da empresa e inclusive atrair investidores interessados em expandir a área de atuação, visando um crescimento futuro.

Diante de tal cenário, foi possível elaborar alguns artefatos para auxiliar o desenvolvimento do sistema, que seguem abaixo.

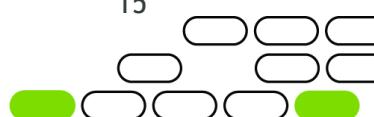
#### Blueprint:

ITENS	DETALHAMENTO
Objetivos	Emitir demandas a outros setores
Atividades	Funcionário entra em contato por ramal/telefone, ou envia e-mail, ou envia mensagens no aplicativo mensageiro, oficial e de uso pessoal, ou mesmo se dirige à mesa de alguém do setor do qual ele deseja a demanda para solicitar diretamente
Questões	Processo sem padronização, sem registros, sem qualquer controle, levando as demandas a não serem sempre atendidas
Barreiras	Resistência dos funcionários em aceitar o uso do sistema, gerenciamento de possíveis indisponibilidades
Ações do cliente	<b>Passo-a-passo do cliente para resolver o problema</b>
Funcionalidades	Registro de demandas, atribuição de atendentes, prazo de atendimento
Interação	Usuário acessa o sistema com o próprio login, escolhe a opção de cadastrar demanda, seleciona o setor responsável,



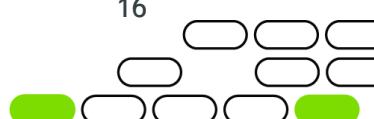
	classifica a demanda e oferece uma breve descrição da solicitação
Mensagem	“Demanda cadastrada com sucesso”, “Fulano iniciou o atendimento”, “Há uma pendência na demanda”, “Demanda atendida”, “Classifique o atendimento”, “Finalizar demanda”, “Iniciar atendimento”, “Encaminhar demanda a outro setor”, “Encaminhar demanda a outro funcionário”, “Reabrir demanda”
Onde Ocorre	Aplicativo
Tarefas aparentes	Criação de tickets, classificação das demandas, finalização de atendimento, avaliação do atendimento, verificação de status, solicitação de relatórios
Tarefas escondidas	Definição de níveis de permissão de acesso, monitoramento do sistema, atualizações do sistema
Processos de suporte	Backup de dados, logística de entrega de produtos, processamento dos dados para emissão de relatórios
Saída desejável	<b>Objetivo a ser alcançado</b>
Sistema intuitivo	Permite ao usuário cadastrar, consultar, atender e finalizar as demandas sem necessidade de amplo treinamento
Consistência dos dados	Objetiva manter registro das demandas de modo a permitir a consulta e resolver eventuais discordâncias (demanda atendida, demanda não atendida, demanda foi de fato solicitada) sem gerar dúvidas
Métricas de produtividade	Os relatórios elaborados oferecem métricas de avaliação da produtividade de cada funcionário, permitindo avaliar gargalos, sobrecarga de pessoas e redistribuição de tarefas

Tabela. Blueprint.



**Canvas da Proposta de Valor:**

Figura. Canvas da Proposta de Valor.



#### 1.1.4 Hipóteses

As informações obtidas durante os processos anteriores permitiram elaborar as seguintes Observações para cada Hipótese:

OBSERVAÇÃO	HIPÓTESES
As solicitações entre setores são realizadas de modo aleatório.	A implantação de um sistema de <i>tickets</i> de atendimento irá centralizar e padronizar o modo como as solicitações são feitas.
A falta de visibilidade sobre o status das demandas realizadas a outro setor gera frustração entre os funcionários. Eles não sabem se a solicitação está sendo tratada.	O sistema permitirá ao demandante verificar se a solicitação já teve o atendimento iniciado, se possui alguma observação, se está aguardando alguma informação complementar ou se já foi atendida.
Não há rastreabilidade de problemas recorrentes.	O sistema oferecerá a possibilidade de geração de relatórios das demandas, permitindo que seja realizada análise dos dados para identificar a recorrência de problemas
Não há registro de ações tomadas para cada demanda.	Cada ação tomada em cada demanda exigirá o registro do motivo daquela ação ser realizada, seja transferência para outro setor, solicitação de complementação de informações, anotações de suporte bem como fechamento da demanda.
As descrições das solicitações nem sempre permitem categorizá-las facilmente	É solicitado ao demandante que classifique o tipo de demanda no momento da abertura.
Os relatórios de demandas e atendimentos demoram muito tempo para serem elaborados, e possuem dados pouco precisos	O sistema irá emitir relatórios conforme os filtros definidos pela gestão da empresa, podendo ser ajustado conforme solicitações de melhorias no sistema - solicitações de melhorias que podem ser abertas no próprio sistema.
Os demandantes erram na classificação da solicitação	O sistema permite ao atendente a qualquer momento em que identificar incorreção na classificação do chamado que esta seja alterada de modo a refletir melhor a natureza da solicitação

Tabela. Observações e hipóteses.



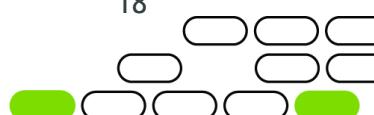
Funcionalidade	B	A	S	I	C	O	Pontuação Total	Prioridade
Centralização de solicitações	5	5	5	3	4	5	27	1
Visibilidade do status das solicitações	4	5	5	3	3	5	25	2
Histórico de solicitações e problemas recorrentes	4	5	3	3	3	4	22	5
Registro detalhado das ações	5	5	4	3	3	3	23	4
Categorização automática das solicitações	3	4	4	2	3	4	20	6
Geração de relatórios personalizados e em tempo real	5	5	5	2	3	3	23	3

Figura. Tabela de priorização.

## 1.2 Solução

### 1.2.1 Objetivo SMART

No período de 45 dias (T), implementar um sistema de *tickets* de atendimento que centralize todas as demandas entre setores da Papéis Miguel Escobar (S), reduzindo no mínimo 30% o tempo médio de atendimento (M), com provimento de uma interface amigável e simples, utilizando padrões modernos de UX/UI (A) e, ao mesmo tempo, melhorando a transparência das operações internas e garantindo a satisfação dos usuários (R).



## 1.2.2 Premissas e Restrições

### Premissas:

- A equipe envolvida possui conhecimento técnico para desenvolver e implantar o sistema;
- Ela conseguirá disponibilizar a versão inicial no prazo;
- Há infraestrutura de TI suficiente para suportar a execução do sistema;
- A inclusão de setores de atendimento será gradual, iniciando com setor de TI e Armazém, e expandindo para os demais gradualmente;
- Haverá regulamentação interna de que, salvo casos excepcionais (sistema fora do ar, por exemplo), todos os funcionários deverão adotar o sistema de *tickets* de atendimento para solicitar demandas aos setores que estiverem habilitados no sistema;
- O sistema estará em constante evolução conforme o feedback dos usuários;
- Não há legislação que impeça a implementação deste sistema.

### Restrições:

- O sistema deve possuir uma versão funcional no prazo de 3 meses, que suporte a criação de tickets de atendimento direcionados aos setores de TI de Armazém;
- O desenvolvimento do sistema deve atender ao orçamento estipulado;
- O sistema deve ser executado em ambiente web (Browser), de modo a ser compatível com um número elevado de dispositivos;
- Os dados devem ser protegidos no armazenamento, e os usuários somente devem ter acesso aos dados que lhes dizem respeito;
- O sistema deve ser capaz de ser expandido, conforme novas funcionalidades sejam sugeridas
- Deve seguir boas práticas de desenvolvimento, de modo a ser fácil de manter.

### Riscos:

1. Embora indesejável, sempre há risco de o projeto atrasar, frustrando as expectativas da diretoria e usuários;
2. O orçamento do projeto pode se mostrar insuficiente e isto pode pôr em xeque a relação custo x benefício;



3. Os usuários podem se recusar a utilizar o sistema, mesmo com determinação superior, e com isso os dados dos relatórios não refletirem a realidade da empresa;
4. O sistema pode apresentar desempenho insatisfatório, devido a inúmeros fatores, como erros de código, estimativa superestimada da infraestrutura existente, falhas de hardware imprevistas, etc.;
5. Pode haver falhas de segurança tanto no desenvolvimento quanto nas tecnologias utilizadas, expondo dados possivelmente sensíveis da empresa.

#### Matriz de Riscos:

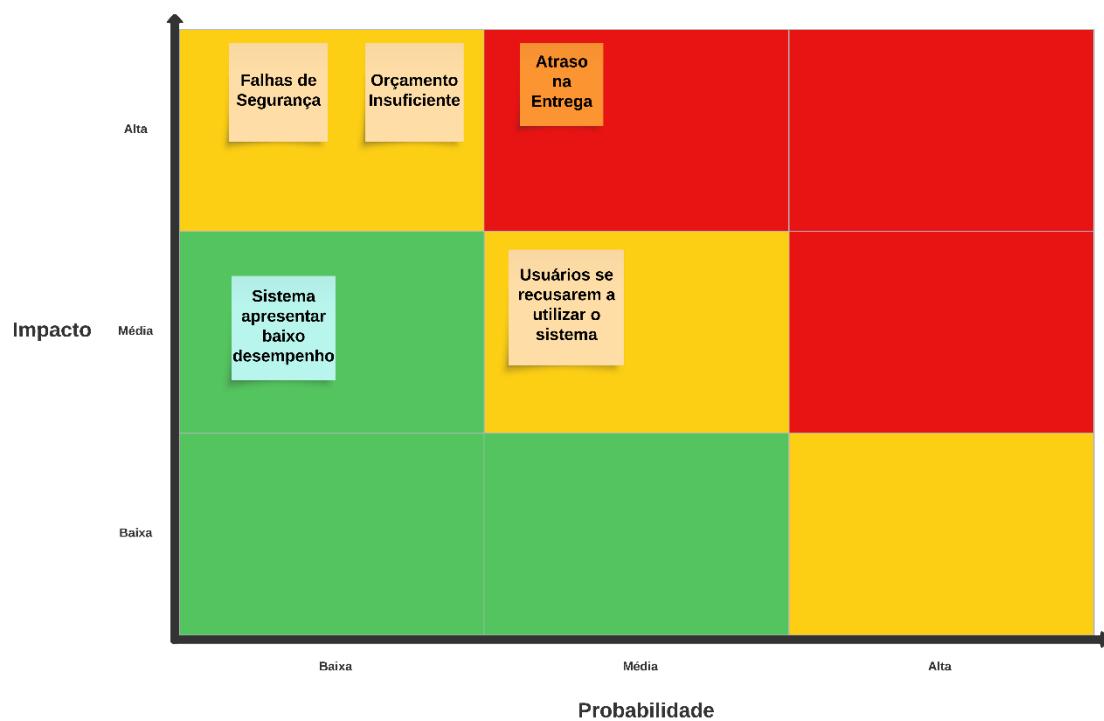
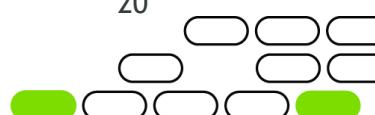


Figura. Matriz de Riscos.



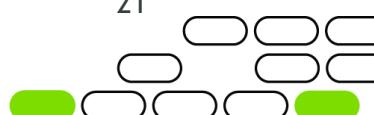
### 1.2.3 Backlog de Produto

The screenshot shows a digital product backlog interface. At the top, it says "Backlog do Produto". Below that, there are three columns representing sprints:

- Sprint 1:** Contains two cards: "Coletar requisitos" and "Criar visão geral do produto".
- Sprint 2:** Contains two cards: "Modelar base de dados" and "Elaborar Casos de Uso".
- Sprint 3:** Contains two cards: "Elaborar plano de infraestrutura" and "Elaborar DAS".

Each sprint section has a "Add a card" button at the bottom right. The interface has a dark blue header and a light blue footer.

Figura. Backlog do produto.



## 2. Área de Experimentação

### 2.1 Sprint 1

Para esta primeira Sprint, foi definido que o objetivo é obter os requisitos do sistema e após isto elaborar uma visão geral do produto.

#### 2.1.1 Solução

- Evidência do planejamento:

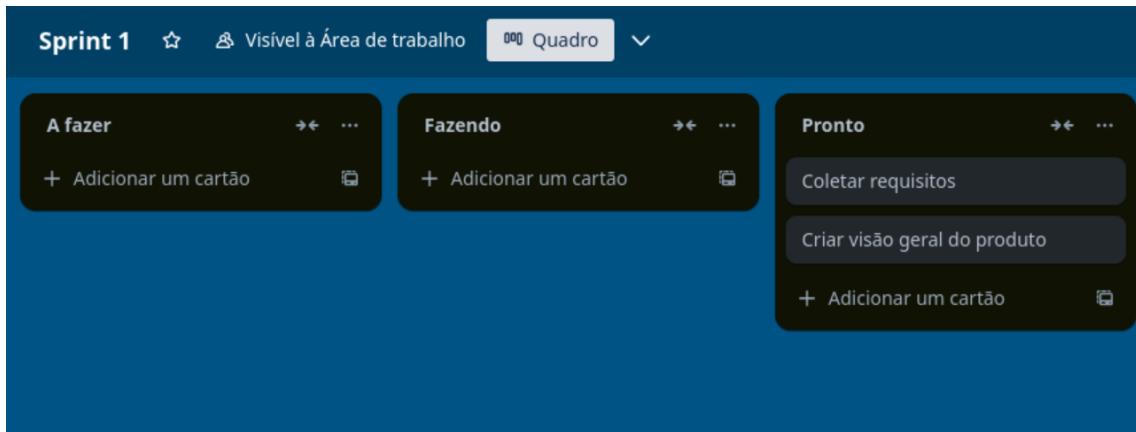


Figura. Planejamento da Sprint 1.

- Evidência da execução de cada requisito: Coleta de requisitos

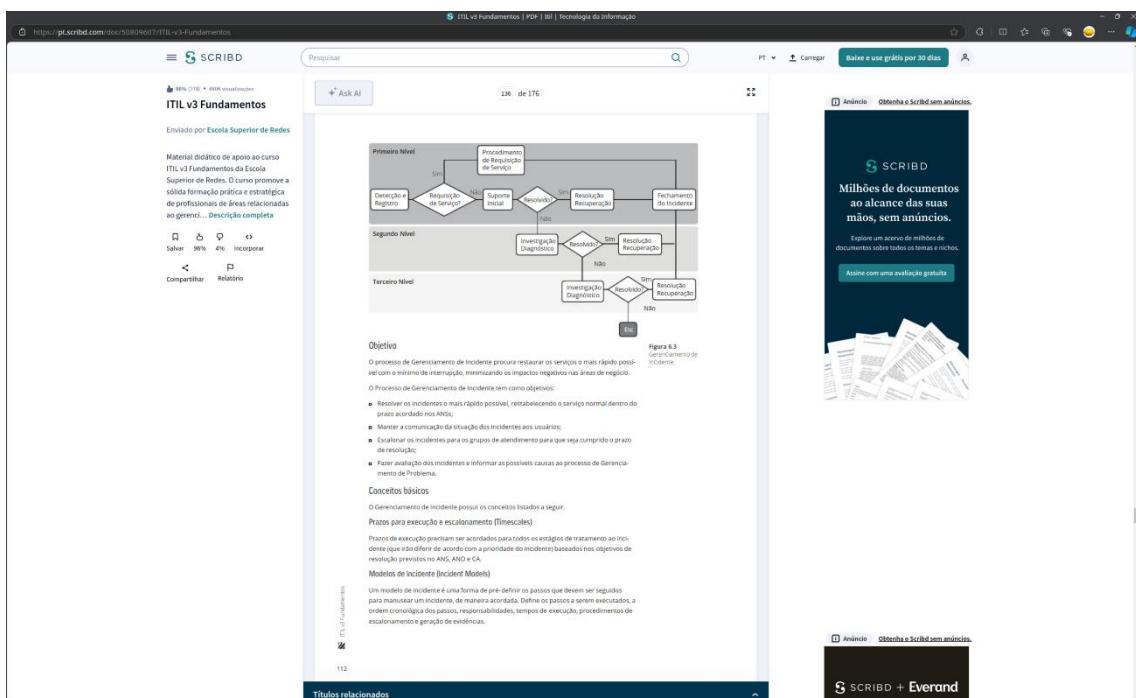
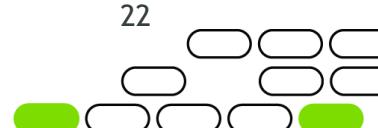


Figura. Tela de acesso ao documento ITIL V3 Fundamentos, fonte do diagrama de Gerenciamento de Incidentes utilizado para ilustrar a coleta de requisitos.



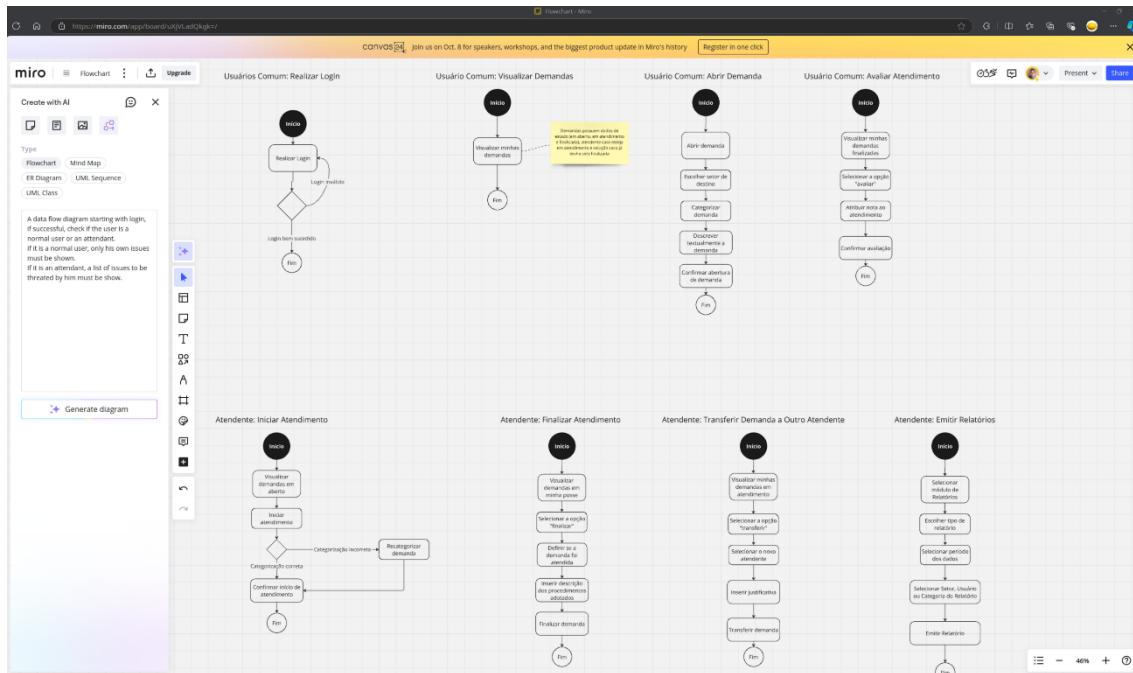


Figura. Criação de diagrama de atividades utilizando a ferramenta Miro. Nesta imagem temos o Diagrama de Atividades para usuários comuns e específicos para atendentes.

- **Evidência dos resultados: Coleta de requisitos**

Em um primeiro momento, foi necessário definir os requisitos necessários ao sistema de gestão de *tickets* de atendimentos, de acordo com as necessidades da Papéis Miguel Escobar. A definição de requisitos foi realizada em um documento informal, onde foi possível verificar as principais funcionalidades desejadas para que o software tivesse o comportamento adequado ao trabalho desenvolvido pela empresa.

Para a averiguação dos requisitos do sistema, foi possível se apoiar nos conceitos de gerenciamento de incidentes do ITIL (FILHO, 2012), que definem algumas regras para o processo de solução de problemas. Este define algumas atividades a serem realizadas desde o momento da detecção de um incidente até a sua solução.



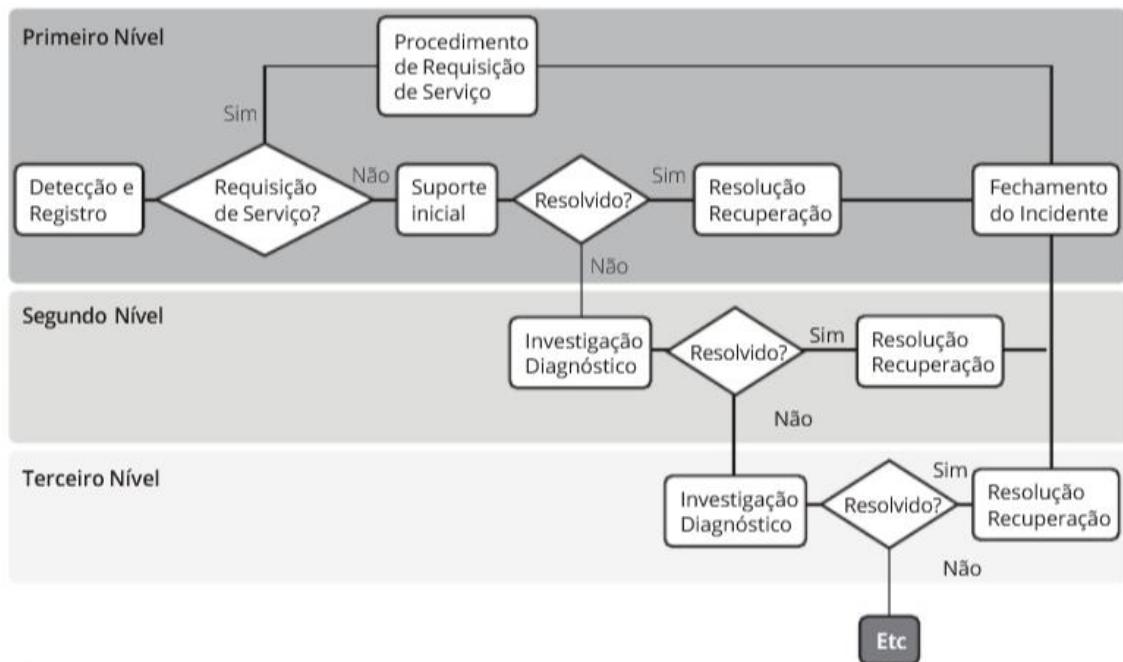
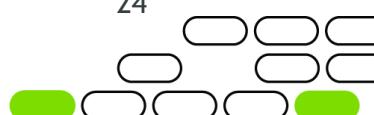


Figura - Gerenciamento de Incidentes - Fonte: (FILHO, 2012, pág. 112) <<https://pt.scribd.com/doc/50809607/ITIL-v3-Fundamentos>>. Acesso em 18/09/2024.

A figura acima apresenta um diagrama exemplificando o processo de resolução de problemas segundo o Gerenciamento de Incidentes do ITIL. Entretanto, somente algumas recomendações foram seguidas para definição de requisitos do sistema em um primeiro momento. Dado o porte da empresa, o trabalho será focado em oferecer um meio de solicitar demandas, oferecer meios de classificar o problema e registrar soluções na base de dados para posterior consulta. As funcionalidades envolvendo níveis de atendimento não estarão presentes nas primeiras versões, mas nada impede de serem escopo de Sprints futuras, que não fazem parte deste projeto.

Algo que é fundamental, dentro do escopo definido, é extrair os pontos cruciais de um sistema de chamados: o registro de solicitações e de soluções. A partir daí, cabe extraír os requisitos do sistema. Ainda dentro da metodologia do ITIL, é possível obter apoio para a extração dos requisitos a partir da sequência de Atividades de Gerenciamento de Incidente, demonstrado na figura abaixo.



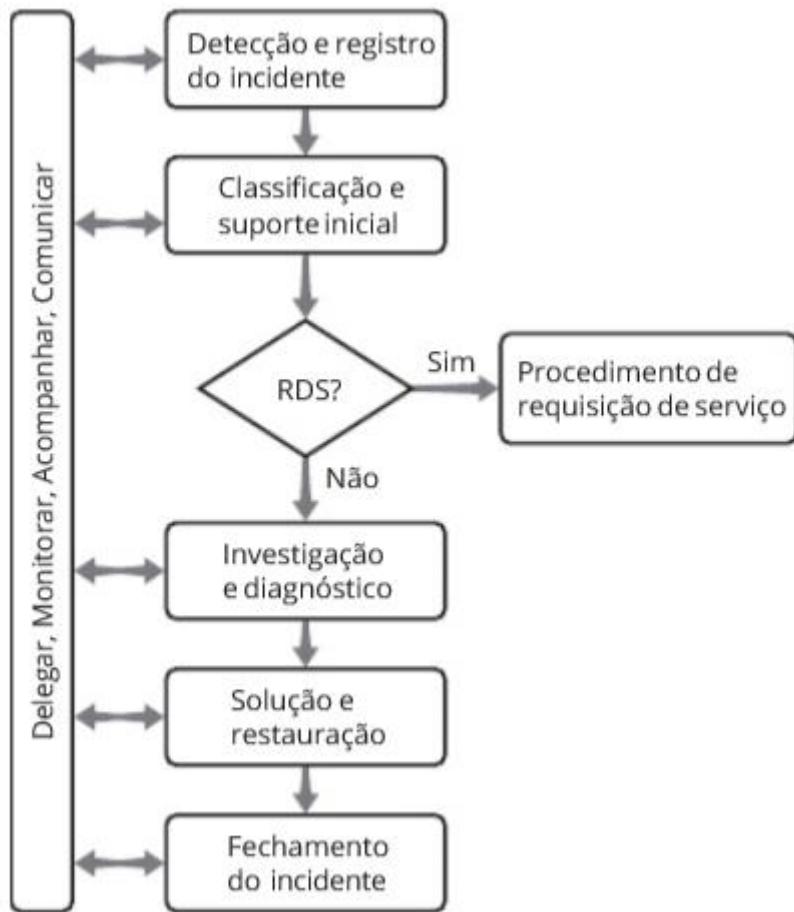
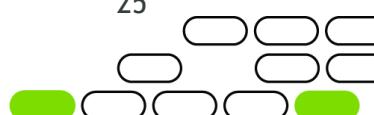


Figura - Gerenciamento de Incidentes - Fonte: (FILHO, 2012, pág. 113) <<https://pt.scribd.com/doc/50809607/ITIL-v3-Fundamentos>>. Acesso em 18/09/2024.

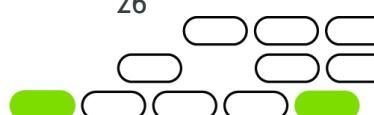
A base de funcionamento de um sistema de chamados está descrita neste diagrama. Desta ponto, é possível com algum esforço especificar os requisitos para o sistema e adaptá-los conforme a realidade e necessidades da Papéis Miguel Escobar.

Os requisitos foram definidos como segue, de acordo com as necessidades e restrições anteriormente apresentadas:

1. Os usuários do sistema serão cadastrados à parte em sistema próprio - inicialmente serão criados diretamente na base de dados.
2. O sistema deve permitir ao usuário acessar uma tela personalizada conforme o tipo de usuário - solicitações de clientes são diferentes de solicitações de funcionários.
3. O sistema deve permitir ao usuário abrir um *ticket* de atendimento.
4. O sistema deve permitir ao usuário categorizar a demanda, de acordo com a solicitação realizada.



5. O sistema deve permitir ao usuário cancelar um ticket aberto por ele mesmo, fornecendo um motivo.
6. O sistema deve apresentar uma lista de *tickets* abertos no sistema, permitindo filtrar a visualização por responsável (ou sem nenhum responsável atribuído), solicitante, categoria, *status* do atendimento (aberto, em atendimento, finalizado).
7. O sistema deve permitir ao atendente atribuir a si mesmo um *ticket*.
8. O sistema deve permitir ao atendente solicitar um *ticket* que já esteja sendo atendido por outro atendente.
9. O sistema deve permitir a um técnico transferir um *ticket* a outro técnico.
10. O sistema deve oferecer um relatório por período de tickets fechados por atendente.
11. O sistema deve permitir ao usuário avaliar o atendimento após finalizado, com nota e comentários.
12. O sistema deve permitir ao usuário reabrir um *ticket*, caso a solução não tenha ocorrido. Neste caso, um novo *ticket* é criado no sistema, referenciando o anterior.



Com base nestes requisitos, foi possível elaborar alguns diagramas de atividades. Seguem abaixo os diagramas de atividades para usuários comuns, que não são atendentes, com as atividades de Realizar Login, Visualizar Demandas, Abrir Demanda e Avaliar Demanda.

## Usuários Comum: Realizar Login

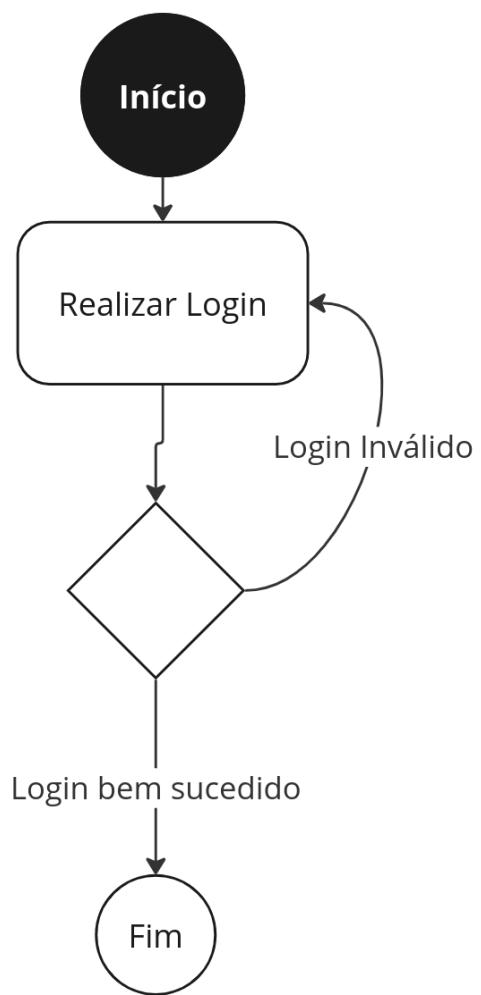


Figura. Diagrama de Atividades - Realizar Login



## Usuário Comum: Visualizar Demandas

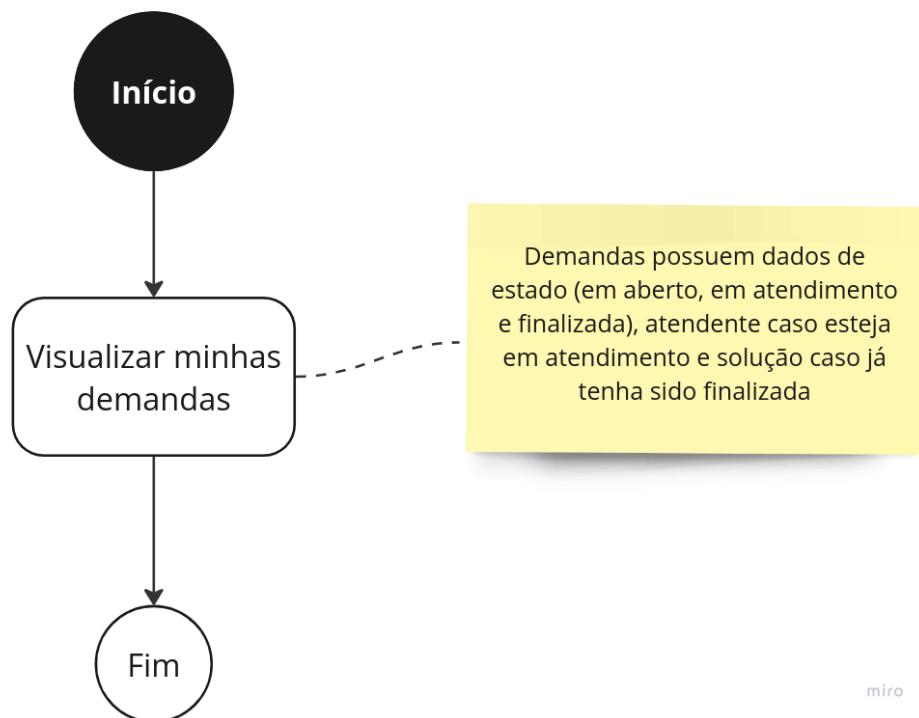
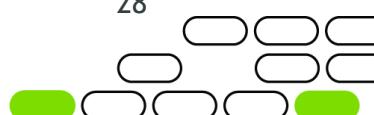


Figura. Diagrama de Atividades - Visualizar Demandas.



## Usuário Comum: Abrir Demanda

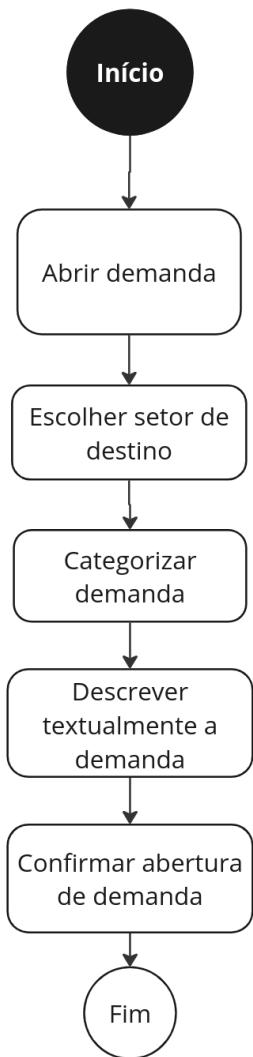
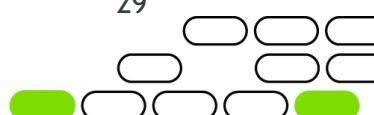


Figura. Diagrama de Atividades - Abrir Demanda.



## Usuário Comum: Avaliar Atendimento

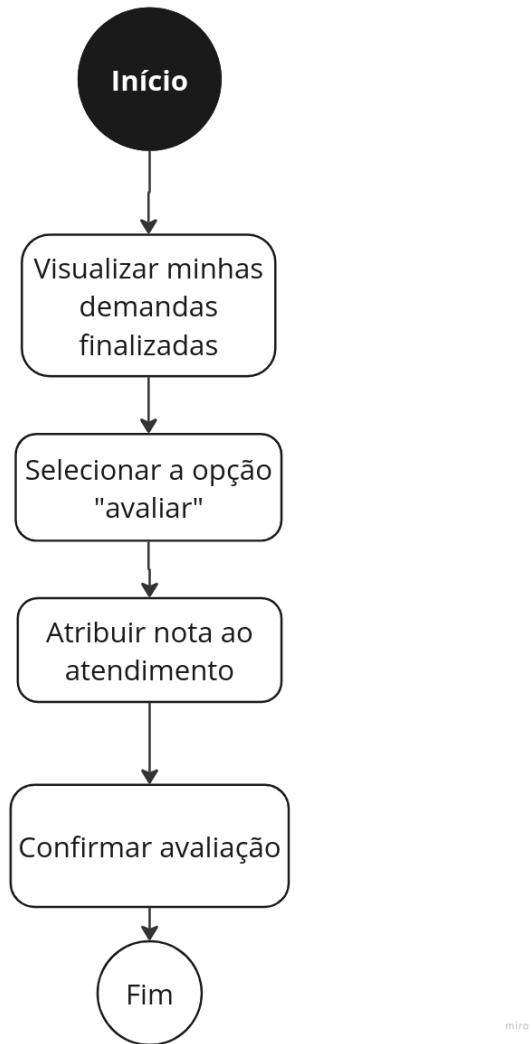
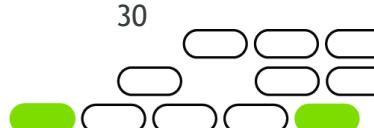


Figura. Diagrama de Atividades - Avaliar Atendimento

Para usuários atendentes, além das opções que usuários comuns possuem, estes também podem executar as atividades de Iniciar Atendimento, Finalizar Atendimento, Transferir Demanda a Outro Atendente e Emitir Relatórios.



## Atendente: Iniciar Atendimento

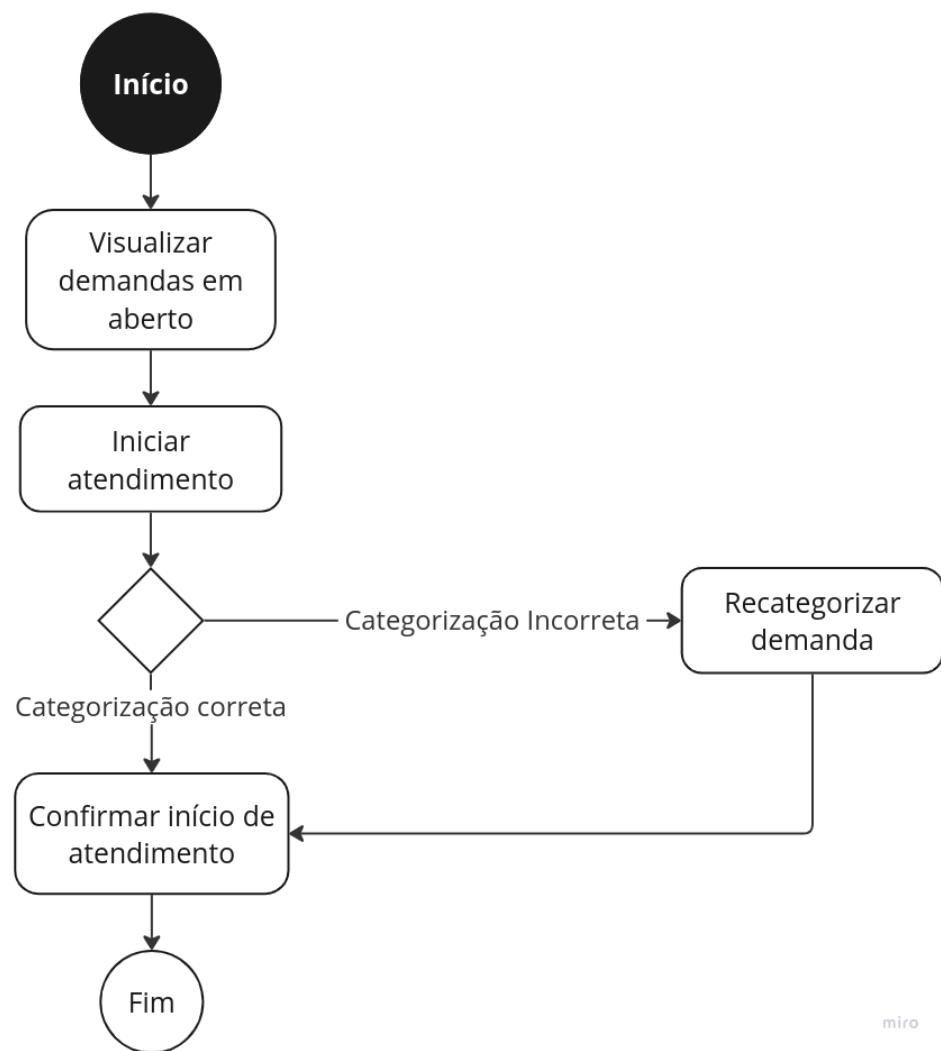


Figura. Diagrama de Atividades - Iniciar Atendimento.

miro

## Atendente: Finalizar Atendimento

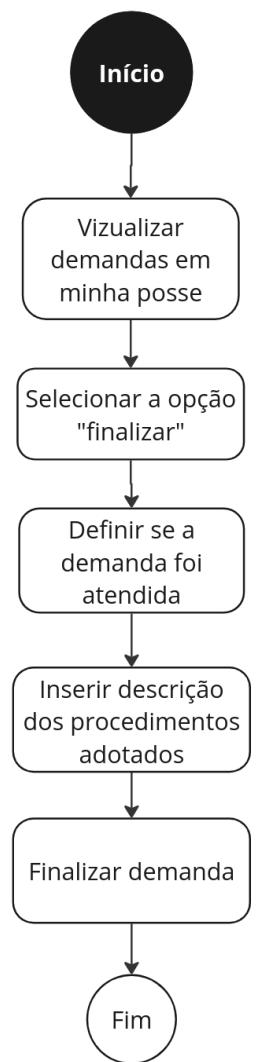
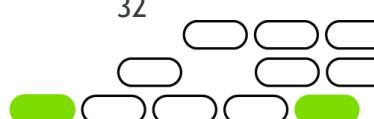
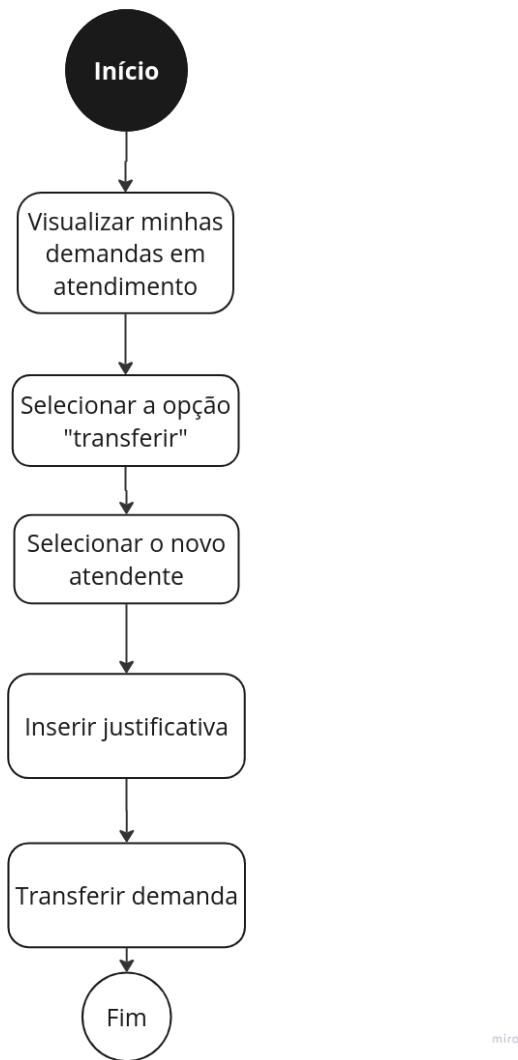


Figura. Diagrama de Atividades - Finalizar Atendimento.



## Atendente: Transferir Demanda a Outro Atendente



miro

Figura. Diagrama de Atividades - Transferir Demanda a Outro Atendente.



## Atendente: Emitir Relatórios

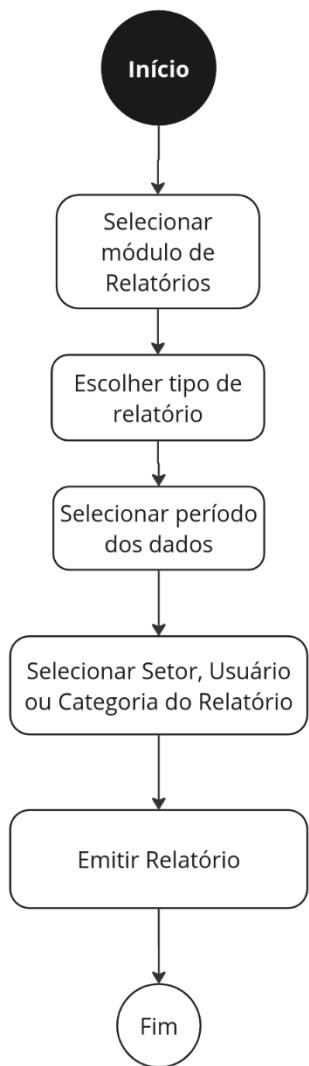
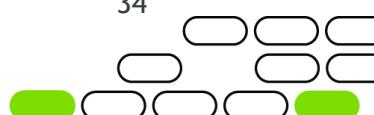


Figura. Diagrama de Atividades - Emitir Relatórios.



- Evidência da execução de cada requisito: Visão geral do produto

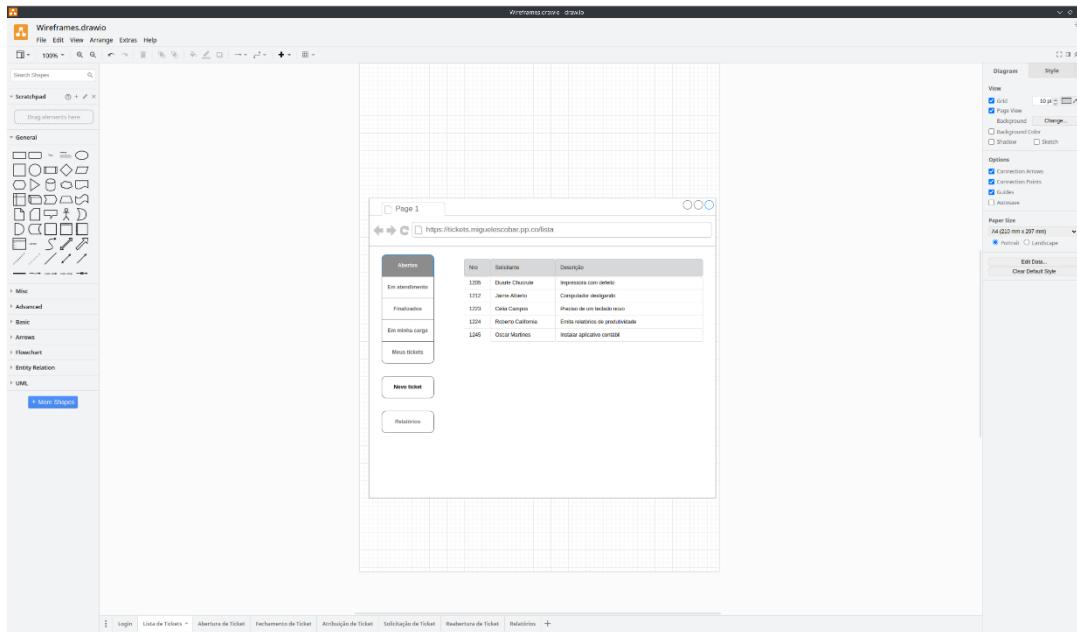


Figura. Criação de *wireframes* das telas utilizando a ferramenta Draw.io. Na imagem, a aba referente ao *wireframe* da lista de demandas.

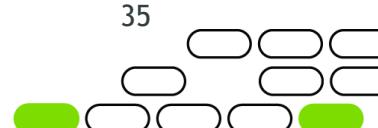
- Evidência dos resultados: Visão geral do produto

Nesta sprint se optou também por elaborar uma visão geral do produto, de modo a balizar as ideias já apresentadas durante a etapa do desafio. Para este item, dividimos a proposta em 5 tópicos: **Objetivo Principal, Público-alvo, Funcionalidades Essenciais, Benefícios Esperados, Wireframe das Telas**

## 1. Objetivo Principal:

O sistema de Gestão de *Tickets* de Atendimento da Papéis Miguel Escobar tem como objetivo principal centralizar, organizar, otimizar e oferecer transparência a todo o processo de atendimento de demandas entre setores da empresa, oferecendo um ambiente unificado que permita acompanhar as solicitações, consultar histórico, emitir relatórios, identificar padrões de dificuldades técnicas enfrentadas pelos funcionários e com isso, obter maior celeridade na execução dos processos internos da empresa, garantindo maior satisfação para os colaboradores e informações mais precisas para os gestores.

## 2. Público-alvo:



O público-alvo principal do produto consiste em 3 tipos de usuários:

- **Colaboradores:** funcionários da Papéis Miguel Escobar que utilizem o sistema para solicitar demandas a outros setores.
- **Atendentes:** funcionários da Papéis Miguel Escobar responsáveis por monitorar os *tickets* abertos que são de responsabilidade de seu setor. Todos os atendentes fazem parte do grupo de colaboradores também, pois podem solicitar demandas a outros setores dos quais não são atendentes. Em um primeiro momento, em que nem todos os setores prestarão atendimento utilizando o sistema, nem todos os colaboradores serão atendentes.
- **Gestores:** os gestores se utilizarão principalmente dos módulos de relatórios diversos do sistema. Estes podem ser parte da diretoria ou chefias setoriais.

### 3. Funcionalidades essenciais:

Em complemento à coleta de requisitos, é possível elencar as funcionalidades essenciais que o sistema deve apresentar já em sua versão inicial, de modo que possa ser considerado um produto mínimo viável:

- **Abertura de *tickets*:** todos os usuários poderão abrir *tickets* de atendimento, e deve ser oferecido ao menos um campo de texto para que ele ofereça uma descrição da demanda.
- **Classificação das demandas:** deve ser oferecida a possibilidade de categorizar o tipo de solicitação em pelo menos 1 categoria. A lista de categorias será definida durante as etapas de desenvolvimento do sistema.
- **Atribuição de responsabilidade:** o sistema deve permitir que os gestores atribuam *tickets* a atendentes específicos do próprio setor. Os próprios atendentes podem por iniciativa própria atribuir os *tickets* a si mesmos, caso julguem adequado executar a demanda envolvida.
- **Status dos *tickets*:** o sistema deve apresentar ao menos 3 tipos de status básicos automáticos aos *tickets*, conforme o andamento: aberto (quando foi criado e a nenhum atendente foi atribuída a responsabilidade), em atendimento (quando já existe um atendente tratando da demanda) e fechado (quando o atendente finaliza o *ticket*, seja cumprindo a demanda ou informando o motivo da impossibilidade de cumprir). O solicitante deve ser capaz de acompanhar o status em que o *ticket* se encontra
- **Histórico de demandas:** o sistema deve permitir consultar demandas existentes, em qualquer status, de modo a permitir obter informações sobre os atendimentos anteriores.
- **Fechamento de *tickets*:** o sistema deve exigir que, ao finalizar um *ticket*, o atendente informe qual foi o procedimento efetuado, de modo a garantir transparência sobre o processo.



- **Avaliação do atendimento:** o sistema deve permitir ao demandante classificar o atendimento realizado, inicialmente em 3 níveis: satisfatório, pouco satisfatório e não atendido.
- **Relatórios:** o sistema deve permitir aos gestores elaborarem relatórios sobre as demandas. Em um primeiro momento, relatórios de quantos tickets de atendimento cada atendente fechou em determinado período, por classificação dos tickets, por setor demandado e por nível de satisfação do demandante.

#### 4. Benefícios esperados:

Os principais benefícios que se espera colher com a adoção do sistema são elencados abaixo:

- **Qualidade do atendimento:** o atendimento tende a melhorar devido tanto à facilidade de se consultar as demandas quanto à possibilidade de o atendimento ser avaliado pelo demandante.
- **Melhor produtividade:** conforme o sistema for alimentado, será possível consultar demandas semelhantes e ganhar tempo de atendimento verificando os procedimentos adotados nestas situações, de modo rápido e simples. Estas consultas podem auxiliar tanto na questão escrita - ler as observações inseridas no fechamento do ticket - quanto interpessoais - é possível consultar diretamente o atendente que solucionou alguma demanda mais complexa anteriormente e questioná-lo dos procedimentos adotados.
- **Visibilidade:** a visibilidade das tarefas executadas será melhorada, visto que a emissão de relatórios permitirá que se demonstre o quanto cada colaborador é acionado.
- **Tomada de decisões:** decisões que se baseiem em dados oferecidos pelo sistema terão melhor suporte, permitindo identificar oportunidades de melhorias de processos.
- **Redução de custos operacionais:** com a avaliação de relatórios será possível obter uma redução de custos ao avaliar possíveis tarefas sendo executadas de modo redundante ou pouco eficiente.



## 5. Wireframe das Telas

Com o apoio das ideias anteriormente citadas, foi possível elaborar alguns *wireframes* de telas que entende-se que possam ser bastante úteis para o processo de desenvolvimento do sistema.

Embora não muito aprofundados, os *wireframes* criados oferecem uma visão gráfica dos requisitos e funcionalidades mínimos para que se tenha um sistema a ser utilizado pelos setores, a seguir: *logon* de usuário, listagem de *tickets* de atendimento, abertura de *tickets*, atribuição de *tickets* abertos para responsabilidade pelo atendente, encerramento de *tickets*, reabertura de *tickets* pelo solicitante, solicitação de responsabilidade de *tickets* de outro atendente e relatórios diversos.

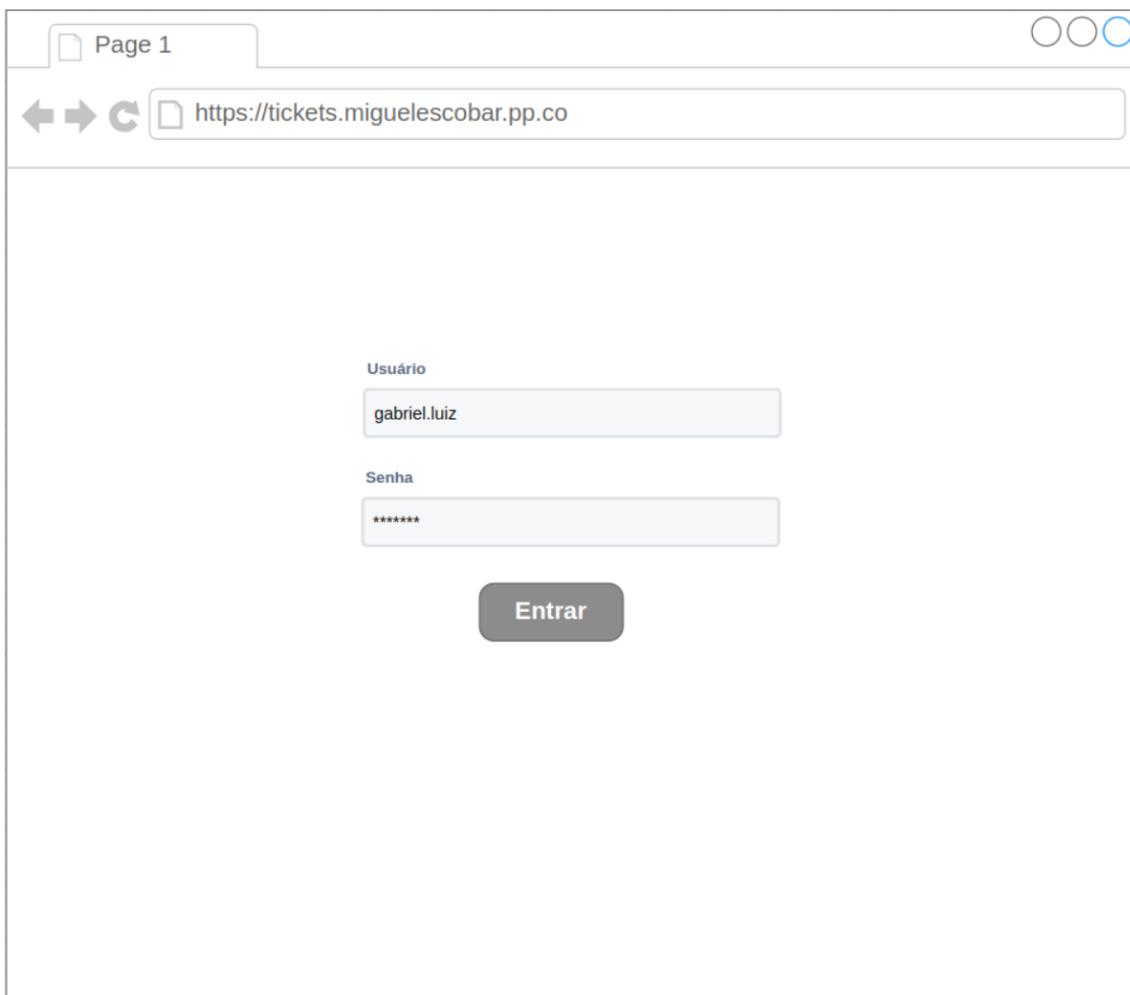


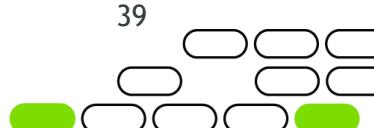
Figura. Tela de login.



The screenshot shows a web-based ticket management system. At the top, there's a header with a back/forward button, a search bar containing the URL <https://tickets.miguelescobar.pp.co/lista>, and three circular icons. On the left, a sidebar has a dark grey header "Abertos" and four buttons: "Em atendimento", "Finalizados", "Em minha carga", and "Meus tickets". Below the sidebar are two rounded rectangular buttons: "Novo ticket" and "Relatórios". The main area contains a table with five columns: "Nro", "Solicitante", and "Descrição" (all in grey headers), followed by five rows of ticket data.

Nro	Solicitante	Descrição
1205	Duarte Chucrute	Impressora com defeito
1212	Jaime Alberto	Computador desligando
1223	Célia Campos	Preciso de um teclado novo
1224	Roberto California	Emita relatórios de produtividade
1245	Oscar Martines	Instalar aplicativo contábil

Figura. Tela de listagem de Tickets.



×

Novo Ticket por DUARTE CHUCRUTE

Destino: Armazém

Classificação: Entrega de Produtos

Descrição:

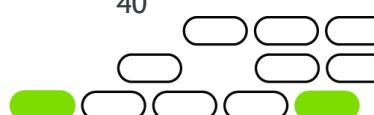
Cliente Marcelo Pasquale efetuou compra de 120 resmas de papel A4 gramatura 250g/m<sup>2</sup>.  
A entrega deve ser efetuada em até 2 dias.

Inserir anexos

Escolher arquivo Nota\_venda.pdf

Cancelar Abrir ticket

Figura. Diálogo de abertura de *ticket* de atendimento.



Fechamento de Ticket #1502 de DUARTE CHUCRUTE x

Solução

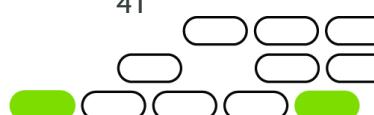
Atendimento Realizado  
 Atendimento Não Realizado

Descrição da solução:

Entrega realizada com sucesso. Comprovante de entrega anexado.

Inserir anexos

Figura. Diálogo de fechamento de *ticket*.



Ticket #1685 aberto por Célia Campos ×

Destino: TI

Classificação: Impressão ▼

Descrição:

Não tenho uma impressora no sistema.

Anexos: sem anexos

Cancelar Iniciar atendimento

Figura. Diálogo de atribuição de responsabilidade do *ticket* aberto para o atendente acessando o sistema.



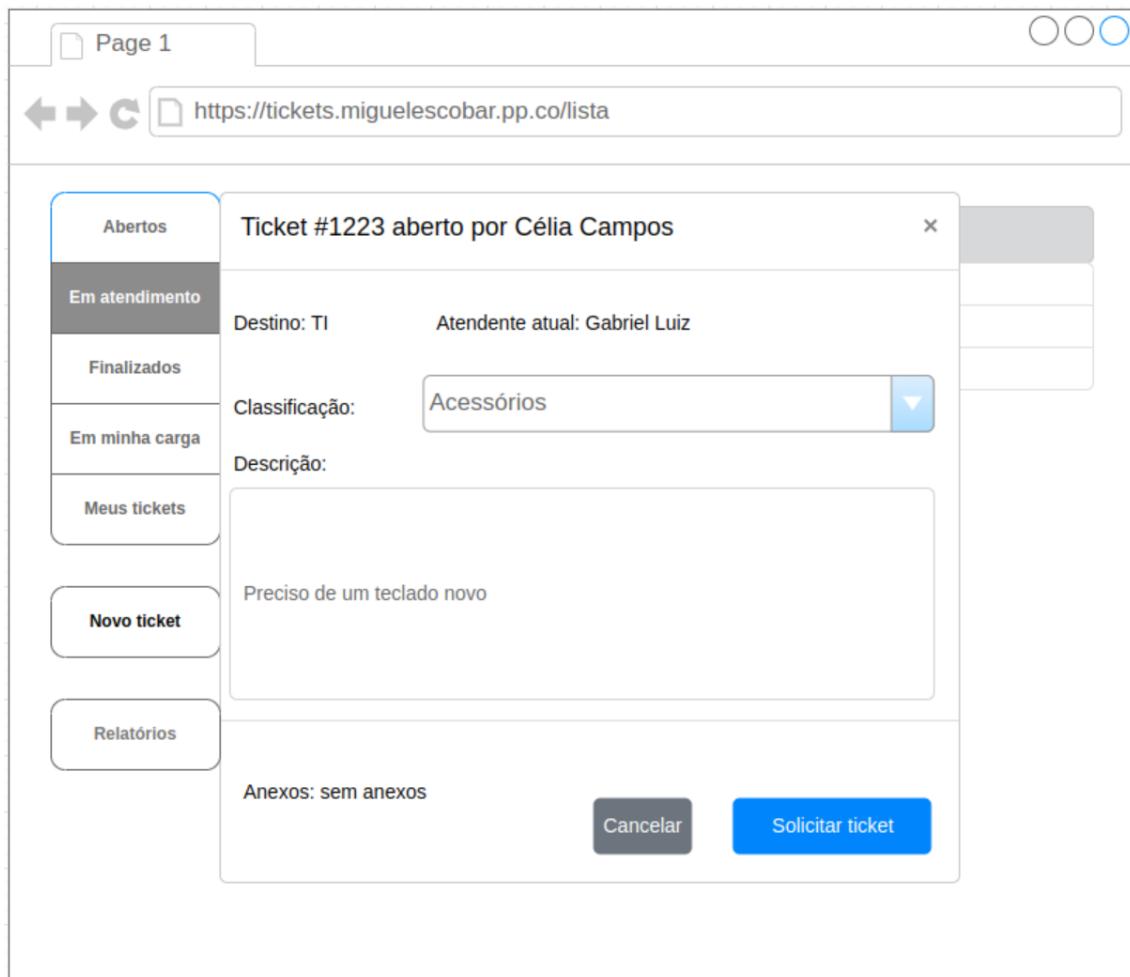


Figura. Tela com diálogo aberto solicitando *ticket* de atendimento que está sob responsabilidade de outro atendente.

Reabrindo Ticket #1258 por DUARTE CHUCRUTE x

Destino: Armazém

Classificação: Entrega de Produtos

Motivo da reabertura

Cliente informa que gramatura do papel não condiz com o solicitado, e encaminha fotos da embalagem

Descrição do ticket anterior

Cliente Marcelo Pasquale efetuou compra de 120 resmas de papel A4 gramatura 250g/m<sup>2</sup>.  
A entrega deve ser efetuada em até 2 dias.

Inserir anexos

Figura. Diálogo de reabertura de *ticket* por não ter sido atendido conforme esperado.



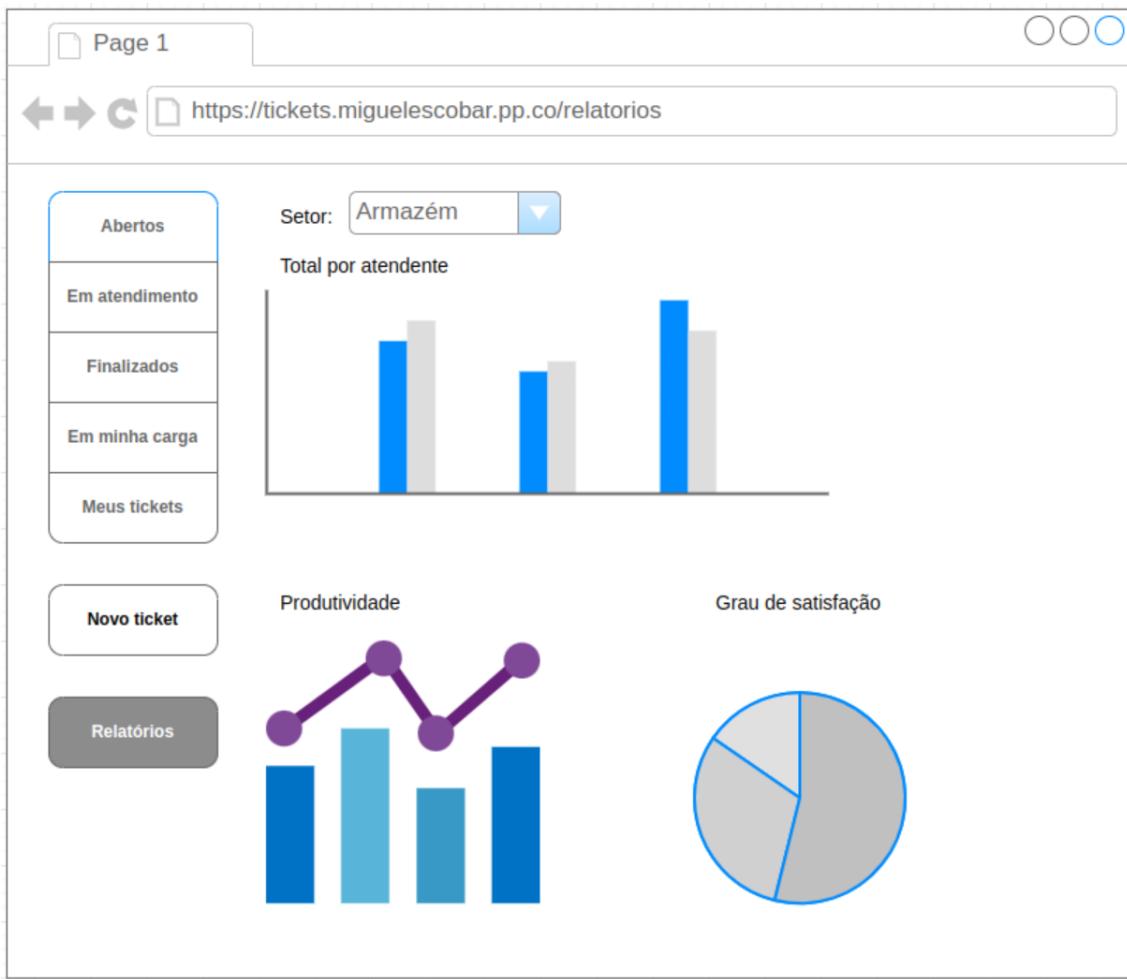


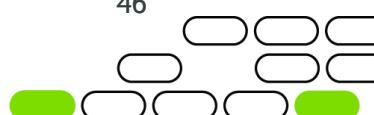
Figura. Tela exemplificando a ideia de uma lista de possíveis relatórios do sistema.

### 2.1.2 Lições Aprendidas

- A definição dos requisitos se provou uma etapa muito desafiadora. A necessidade de avaliar as necessidades de um sistema põe em prática muitas das lições aprendidas não somente durante o curso, mas também durante a vida profissional, e entender as necessidades dos usuários é sempre uma tarefa que demanda muito tempo, esforço e dedicação.
- O uso de ferramentas com as quais não sou tão familiarizado, como o Miro e O Draw.io para criação dos diagramas foi bastante importante para a execução das tarefas da Sprint. E deverão ser úteis nas Sprints seguintes.
- Para a criação dos wireframes foi necessário também ter alguma noção de UX/UI de modo a projetar um ambiente operacional que ao ser implementado seja simples, porém funcional.
- O apoio de literatura do ITIL foi importantíssimo para balizar a criação dos Diagramas de Atividades, ainda que não se utilize todas as regras deste framework.



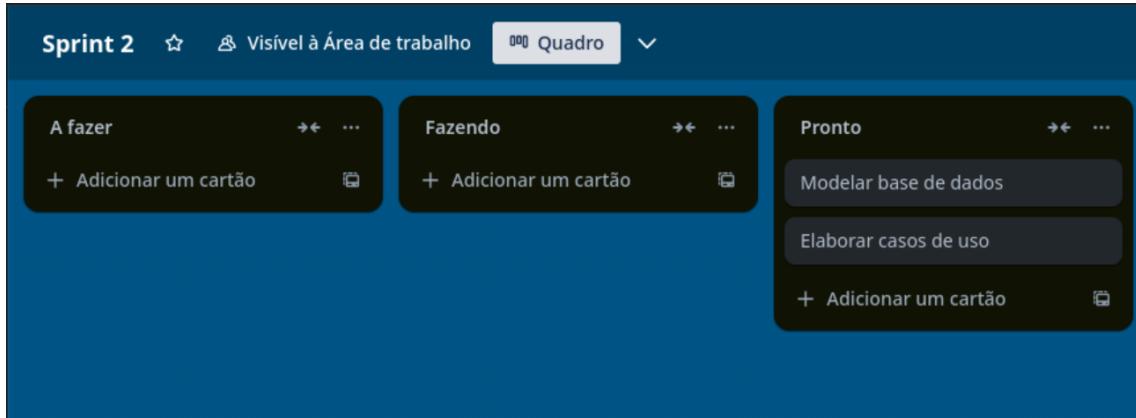
- A ferramenta Draw.io poderia ser utilizada nos dois cenários, tanto para criação dos *wireframes* quanto para o Diagrama de Atividades. A opção pelo Miro se deveu puramente por decisão de aprender a utilizar a ferramenta e por conta de que, para o diagrama em questão, a simplificação dos *shapes* disponíveis acabou por ser benéfica, pois a curva de aprendizado da ferramenta para esta tarefa foi extremamente rápida e a velocidade com que os diagramas foram criados foi compensatório.



## 2.2 Sprint 2

### 2.2.1 Solução

- Evidência do planejamento:



- Evidência da execução de cada requisito: Elaborar Casos de Uso

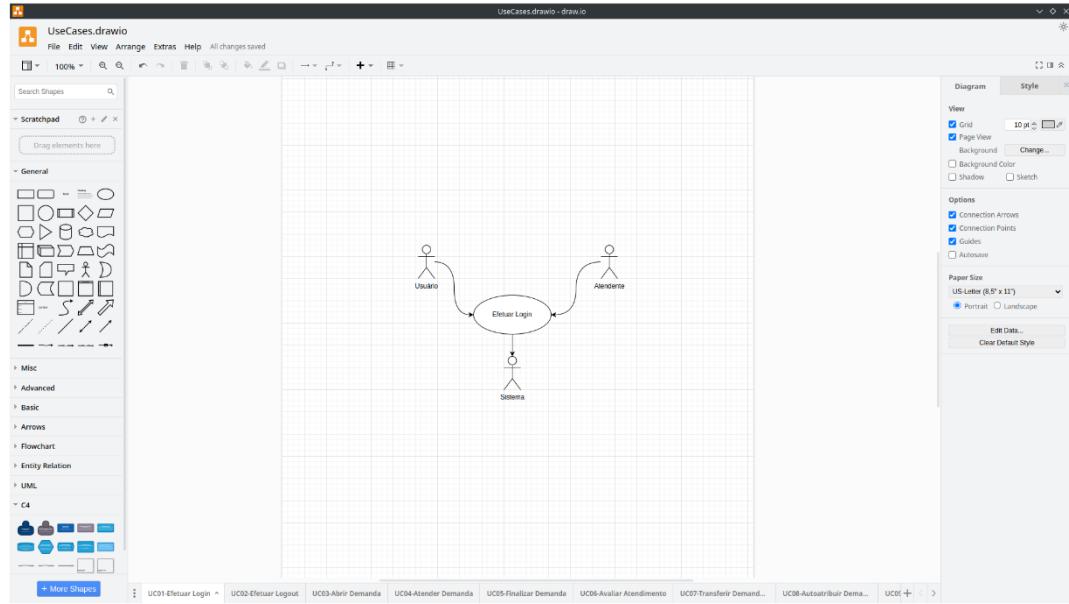


Figura. Criação de Diagrama de Casos de Uso. Na figura, print de tela do Draw.io, onde está selecionada a aba referente ao UC01 - Efetuar Login.

- **Evidência dos resultados: Elaborar Casos de Uso**

O desenvolvimento dos casos de uso do sistema foi realizado a partir das necessidades que são desejadas do sistema. Por se tratar de um sistema com poucos módulos envolvidos, há poucos atores por consequência. Os atores identificados foram:

**Usuário:** responsável por executar as solicitações junto ao sistema.

**Sistema:** responsável por efetuar as respostas às solicitações dos usuários.

**Administrador:** responsável pela emissão de relatórios de atendimento

Seguem abaixo os casos de uso definidos:

## Casos de Uso

### **UC01 – Efetuar login**

*Descrição:* o usuário efetua login no sistema

*Atores:* usuário, sistema

*Precondições:* o usuário deve estar deslogado. O usuário deve possuir cadastro ativo no sistema

*Pós-condições:*

Em caso de sucesso:

O usuário é enviado para a tela inicial do sistema.

Em caso de fracasso:

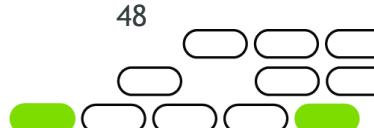
Uma mensagem é exibida na tela, informando o motivo da impossibilidade de se logar no sistema (cadastro, usuário ou senha inválida, falha de acesso ao banco de dados).

*Fluxo normal:*

1. Usuário acessa o sistema
2. Usuário fornece nome de usuário e senha
3. Sistema verifica a consistência dos dados informados junto à base de dados
4. O login é efetuado no sistema

*Fluxo alternativo:*

*Exceção:* o usuário ou a senha informada é inválida. O sistema informa que não foi possível se logar no sistema por este motivo.



## **UC02 – Efetuar logout**

*Descrição:* o usuário sai do ambiente do sistema de chamados

*Atores:* usuário, sistema

*Precondições:* o usuário deve estar logado.

*Pós-condições:*

Em caso de sucesso:

Os dados de sessão são eliminados e o sistema apresenta a tela inicial de login.

Em caso de fracasso:

O sistema informa qual falha ocorreu (impossibilidade de limpeza dos dados de sessão, falha ao acessar servidor) solicitando que o usuário efetue nova tentativa em alguns instantes.

*Fluxo normal:*

1. Usuário executa ação de logout
2. Sistema limpa as informações de sessão
3. Sistema retorna à tela inicial

## **UC03 – Abrir demanda**

*Descrição:* o usuário inicia a abertura de uma demanda no sistema.

*Atores:* usuário, sistema

*Precondições:* o usuário deve estar logado no sistema.

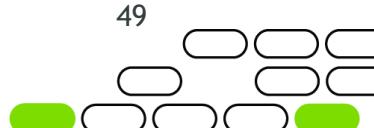
*Pós-condições:*

Em caso de sucesso:

O chamado é aberto no sistema e ganha automaticamente o estado “em aberto”.

Em caso de insucesso:

O sistema informa qual falha ocorreu durante o processamento da requisição (falha de acesso ao banco de dados, perda de comunicação com o servidor).



*Fluxo normal:*

1. Usuário seleciona opção de abrir nova demanda
2. Usuário seleciona as opções de classificação da demanda
3. Usuário fornece uma descrição do problema enfrentado ou da solicitação realizada
4. Usuário confirma os dados informados
5. A demanda é aberta no sistema

*Fluxo alternativo:*

Exceções: caso haja falha no acesso à base de dados para inserção, é informado ao usuário que este deve tentar novamente mais tarde.

**UC04 – Atender demanda**

*Descrição:* atendente inicia o atendimento de demanda no sistema que esteja aguardando atendimento.

*Atores:* atendente, usuário, sistema

*Precondições:* atendente deve estar logado no sistema.

*Pós-condições:*

Em caso de sucesso:

O chamado é atribuído ao atendente que o abriu

Em caso de fracasso:

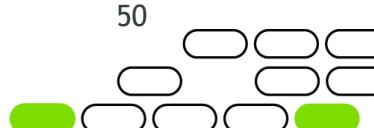
É informado o motivo da falha (problema de conexão com o sistema ou falha no acesso ao banco de dados)

*Fluxo normal:*

1. Atendente escolhe a demanda aberto a ser atendido
2. Verifica as informações disponíveis
3. Escolhe a opção “Iniciar atendimento”
4. Chamado é atribuído a si mesmo no sistema

*Fluxo alternativo:*

Exceção: Caso outro atendente já tenha atribuído a si mesmo o chamado durante o procedimento, é informado ao atendente que este já se encontra em atendimento por outro atendente.



## UC05 – Finalizar demanda

*Descrição:* atendente terminou o atendimento da demanda e deseja finalizá-la.

*Atores:* atendente, usuário, sistema

*Precondições:* atendente deve estar logado no sistema. A demanda tem que estar atribuída a este atendente

*Pós-condições:*

Em caso de sucesso:

A demanda é finalizada e sai da lista de demandas em atendimento

Em caso de fracasso:

Uma mensagem informando o motivo da falha (conexão com o servidor ou acesso ao banco de dados) é apresentada na tela

*Fluxo normal:*

1. Atendente seleciona demanda da lista em sua responsabilidade
2. Escolhe a opção “Finalizar Demanda”
3. Uma caixa de diálogo é apresentada, solicitando uma descrição da solução
4. Atendente submete uma descrição
5. Sistema atualiza a demanda para o status finalizado

## UC06 – Avaliar atendimento

*Descrição:* usuário avalia o atendimento de demanda finalizada

*Atores:* usuário, atendente, sistema

*Precondições:* usuário deve ter sido o responsável pela abertura da demanda no sistema. Demanda deve estar com status finalizado.

*Pós-condições:*

Em caso de sucesso:

A avaliação é salva no sistema

Em caso de fracasso:



É informado o motivo da falha, se de acesso ao servidor de aplicação ou de acesso ao banco de dados. É solicitado ao usuário tentar novamente mais tarde

*Fluxo normal:*

1. Usuário seleciona demanda aberta por si mesmo e já finalizada
2. Atribui uma nota no campo correspondente
3. Submete a avaliação e está é atualizada no sistema

### **UC07 – Transferir demanda a outro atendente**

*Descrição:* atendente deseja transferir demanda que está atendendo para outro atendente

*Atores:* atendente, sistema

*Precondições:* demanda deve estar atribuído ao atendente.

*Pós-condições:*

Em caso de sucesso:

Demandá é transferida para o novo atendente

Em caso de fracasso:

É informado o motivo da falha, se de acesso ao servidor de aplicação ou de acesso ao banco de dados. É solicitado ao usuário tentar novamente mais tarde

*Fluxo normal:*

1. Atendente seleciona chamado que deseja transferir
2. Escolhe o atendente para quem deseja transferir o chamado
3. Demanda muda de responsável

*Fluxos alternativos:*

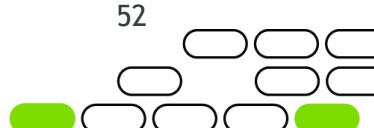
### **UC08 – Autoatribuir demanda de outro atendente**

*Descrição:* atendente atribui a si mesmo demanda atribuída a outro atendente

*Atores:* atendente, sistema

*Precondições:* Demanda precisa que estar em atendimento e de posse de outro atendente

*Pós-condições:*



Em caso de sucesso:

Chamado é transferido e passa a ter novo responsável

Em caso de fracasso:

É informado o motivo da falha, se de acesso ao servidor de aplicação ou de acesso ao banco de dados. É solicitado ao usuário tentar novamente mais tarde

*Fluxo normal:*

1. Atendente verifica demanda sendo atendida por outro atendente
2. Atendente transfere do chamado para si mesmo
3. Atendente é o novo responsável pela demanda

### **UC09 – Reabrir demanda**

*Descrição:* usuário deseja reabrir chamado que foi fechado pelo suporte, por insatisfação com a solução

*Atores:* usuário, sistema

*Precondições:* chamado deve ter sido finalizado. Chamado deve ter sido aberto pelo usuário que pretende reabri-lo. Chamado não deve ter sido avaliado ainda.

*Pós-condições:*

Em caso de sucesso:

Abre-se um novo chamado no sistema, referenciando o chamado anterior

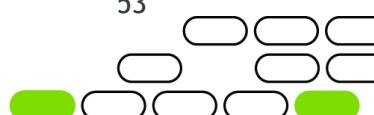
Em caso de fracasso:

É mostrada uma mensagem informando o motivo da falha, se no servidor de aplicações ou no servidor de banco de dados

*Fluxo normal:*

1. Usuário seleciona chamado que foi finalizado de sua lista e que não ficou satisfeito com o resultado
2. Seleciona opção reabrir chamado
3. Descreve o motivo de reabertura
4. Um novo chamado é aberto no sistema, referenciando o anterior

### **UC10 – Listar demandas**



*Descrição:* usuário logado recebe uma lista de chamados, conforme critérios de filtragem

*Atores:* usuário, sistema

*Precondições:* usuário deve estar conectado ao sistema

*Pós-condições:*

Em caso de sucesso:

É apresentada uma lista de chamados conforme os critérios de filtragem

Em caso de fracasso:

Sistema apresenta motivo da falha, se falha de acesso à base de dados ou falha no servidor de aplicação

*Fluxo normal:*

1. Usuário seleciona as opções padrão de filtragem: Abertos, Em Atendimento, Em minha posse (Somente usuário Atendente), Finalizados
2. Uma lista de chamados é apresentada na tela, conforme a filtragem de informações

## **UC11 – Emitir Relatório**

*Descrição:* sistema emite relatórios diversos, conforme filtros definidos pelo atendente, que atua como gestor

*Atores:* atendente, sistema

*Precondições:* atendente deve estar conectado ao sistema

*Pós-condições:*

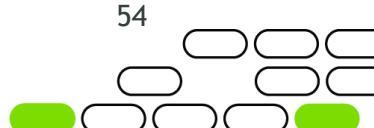
Em caso de sucesso:

É apresentado o relatório solicitado conforme os filtros estipulados

Em caso de fracasso:

Sistema apresenta o motivo da falha, se falha de acesso à base de dados ou falha no servidor de aplicação

*Fluxo normal:*



1. Atendente define o tipo de relatório
2. Atendente especifica os filtros para o relatório
3. O relatório solicitado é apresentado na tela, com opção de realizar o download em formato de documento

Após a elaboração dos casos de uso, para termos uma visualização gráfica de todos, foram elaborados os diagramas para cada um deles.

### Diagramas de Caso de Uso

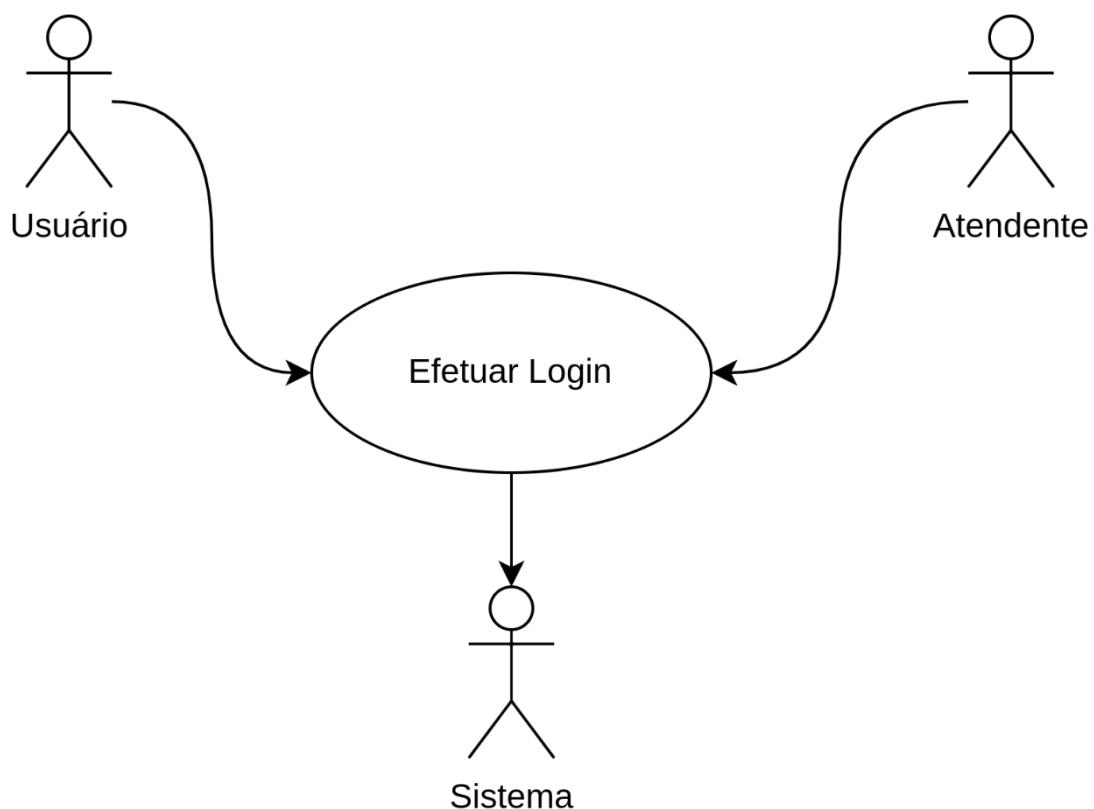
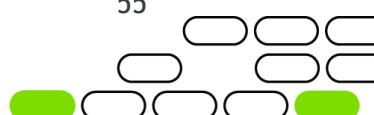


Figura. UC01 - Efetuar Login.



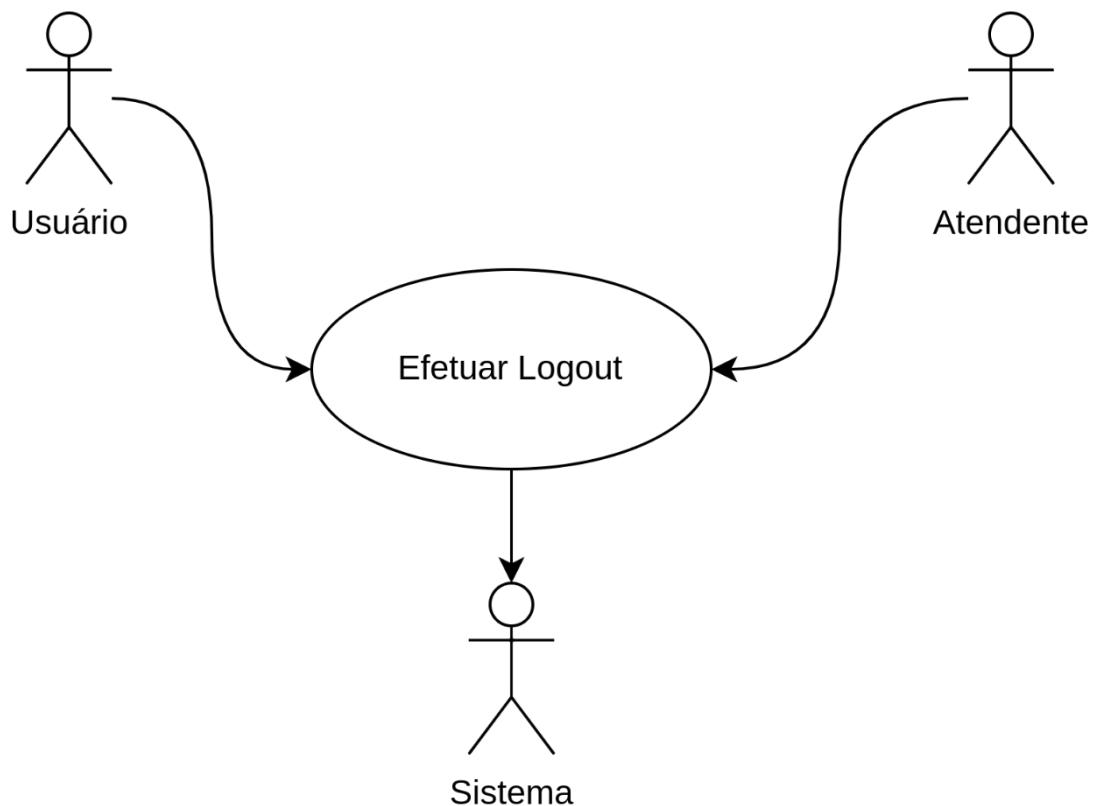
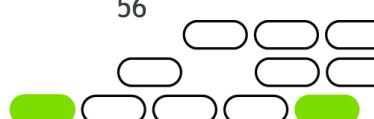


Figura. UC02 - Efetuar Logout.



Figura. UC03 - Abrir Demanda.



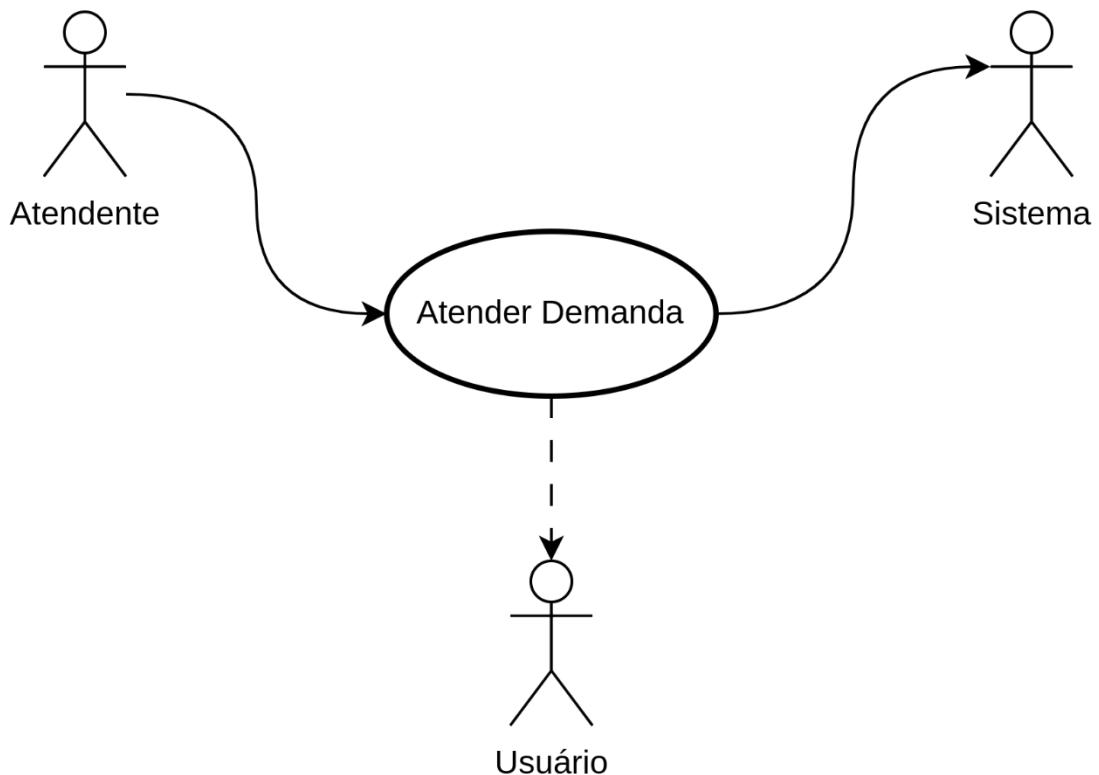


Figura. UC04 - Atender Demanda.

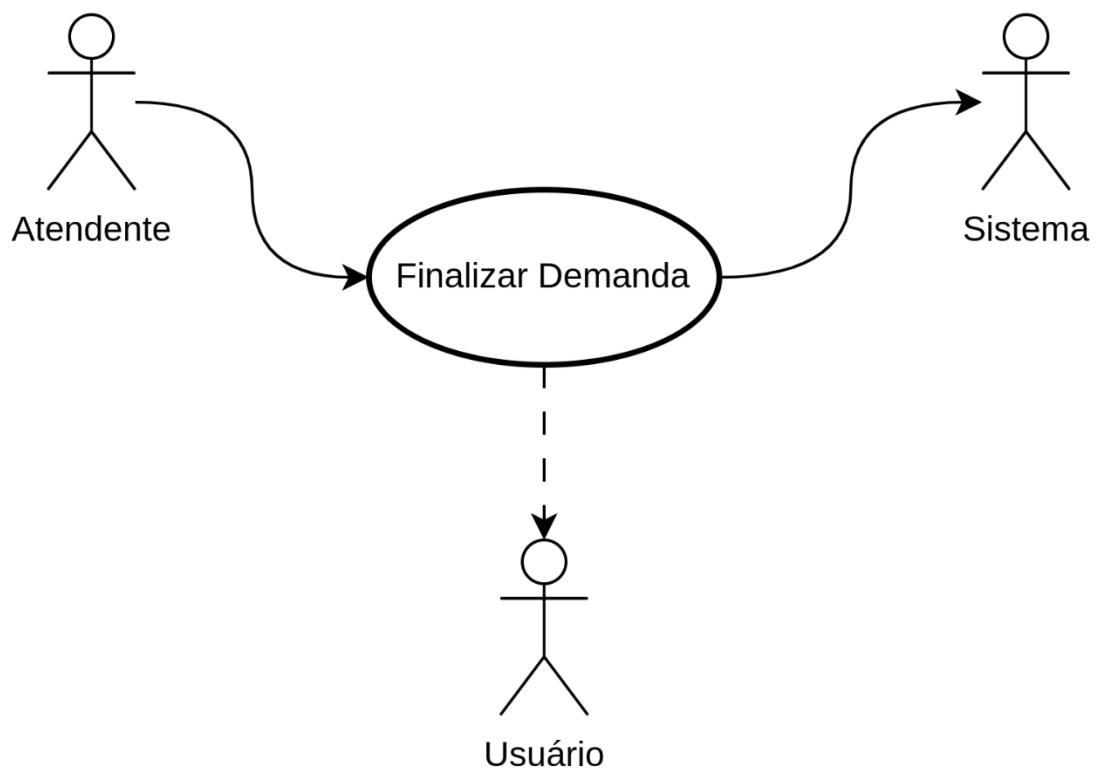


Figura. UC05 - Finalizar Demanda.

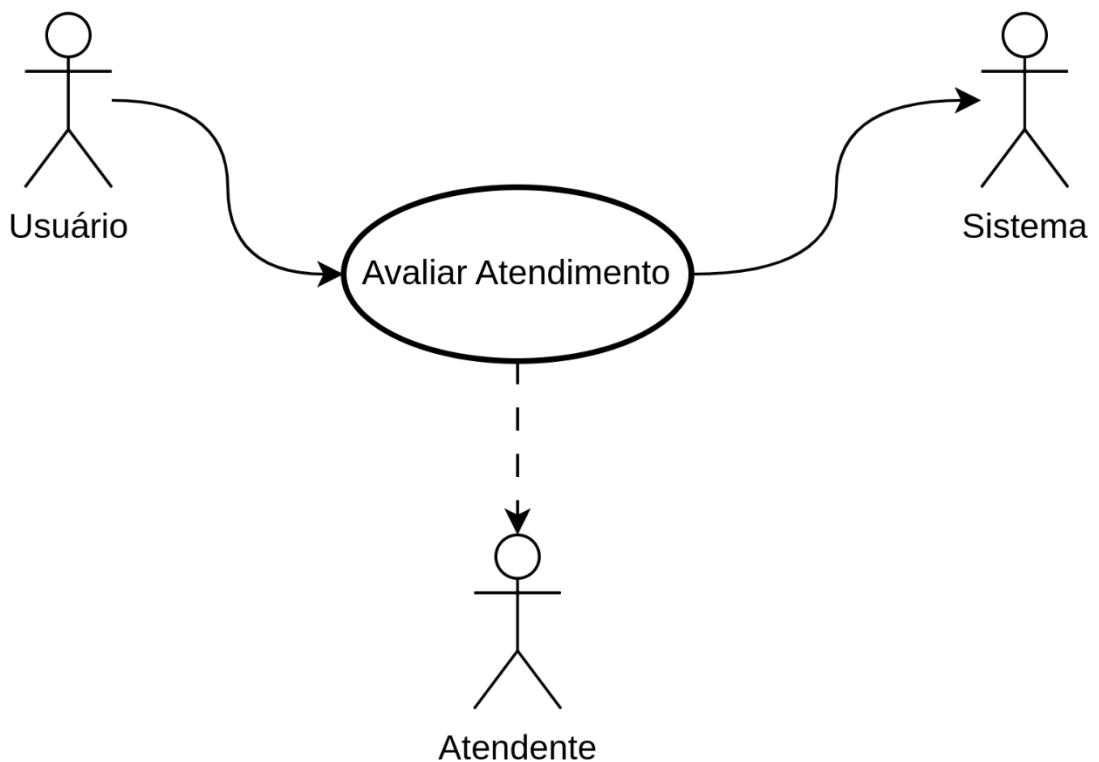


Figura. UC06 - Avaliar Atendimento.

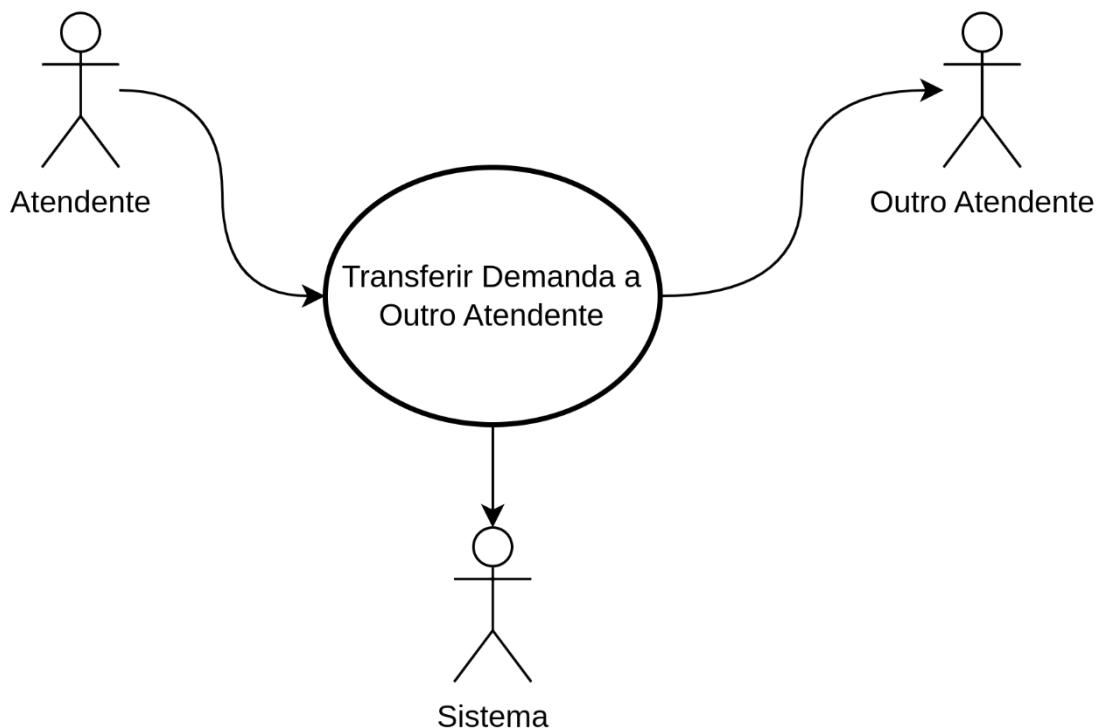
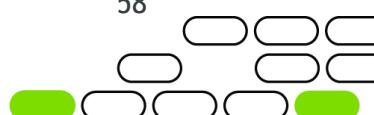


Figura. UC07 - Transferir Demanda a Outro Atendente.



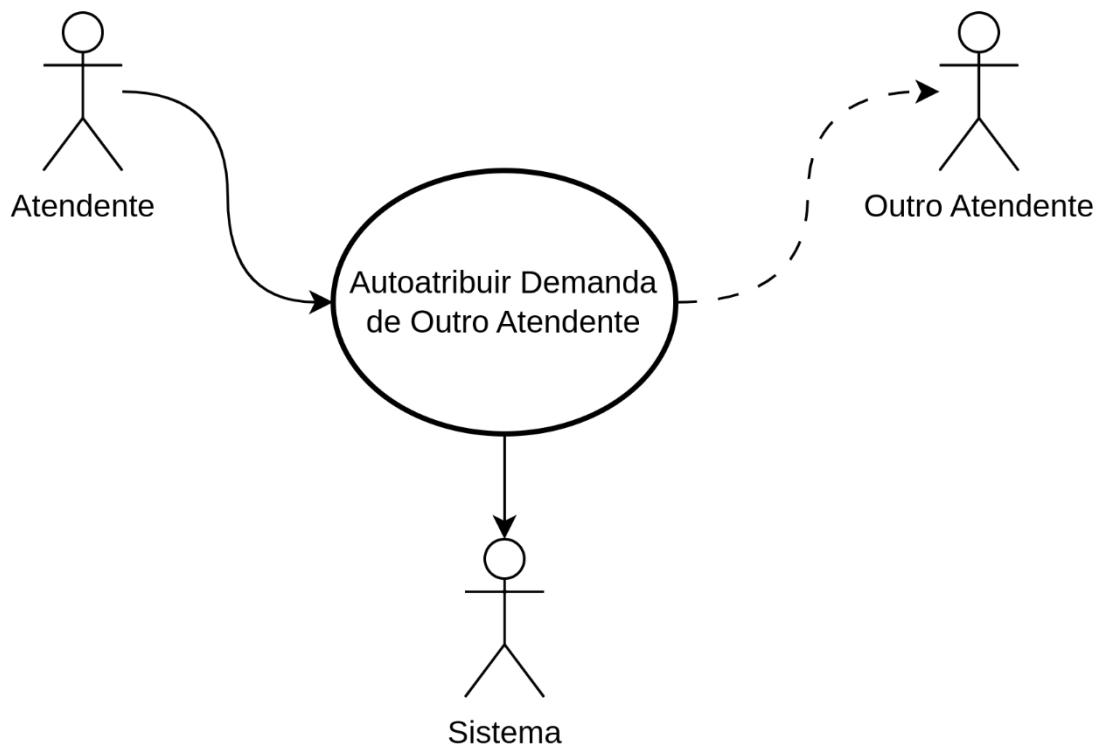


Figura. UC08 - Autoatribuir Demanda de Outro Atendente.

- Evidência da execução de cada requisito: Modelar Base de Dados

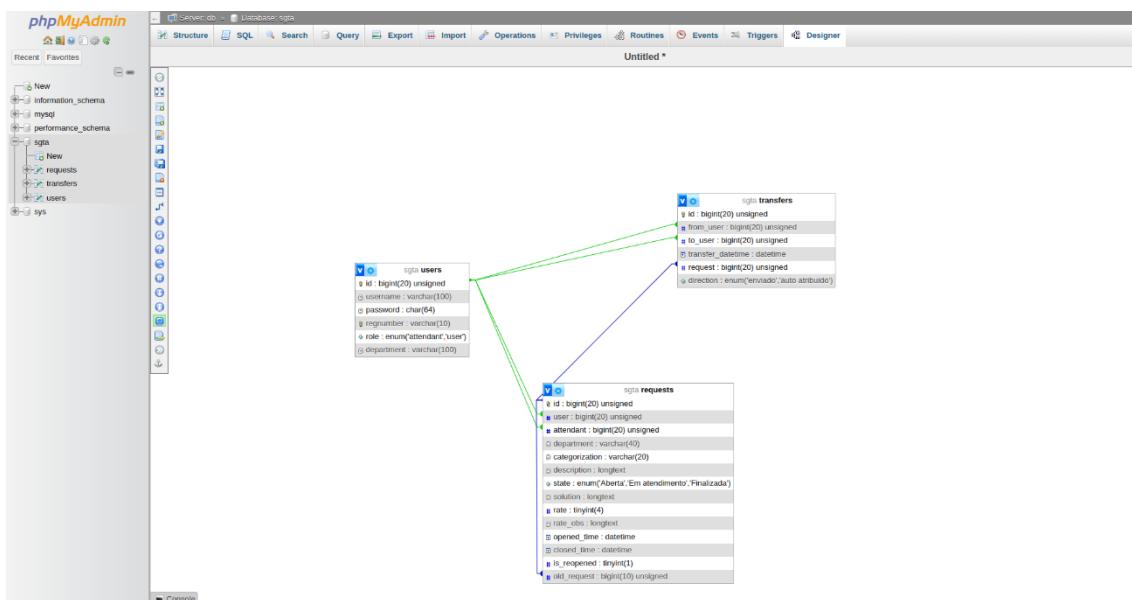


Figura. Tela do phpMyAdmin na Aba Designer, demonstrando os campos e relacionamentos entre as três tabelas no MariaDB.

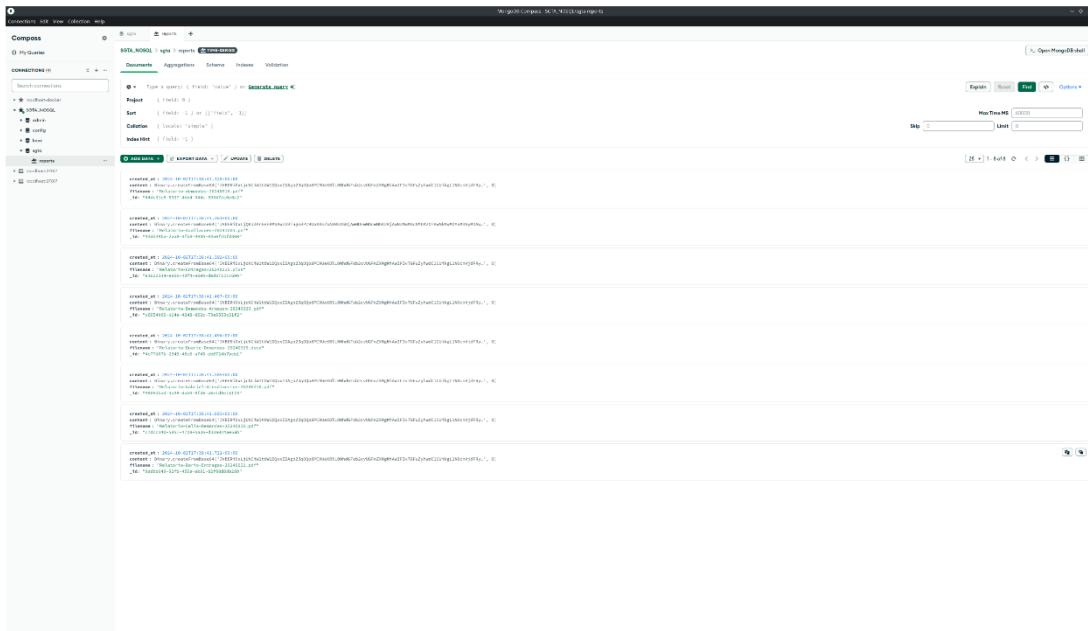


Figura. Tela do MongoDB Compass, com 8 relatórios armazenados.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
07faed9db777	sgta_pme-phpmyadmin-1	0.00%	29.45MiB / 31.12GiB	0.09%	74kB / 203kB	0B / 283kB	11
933950fda0dd	sgta_pme-db-1	0.01%	108.1MiB / 31.12GiB	0.34%	612kB / 2.49MB	12MB / 27.1MB	13
d0cbf2bfc52c	sgta_pme-mongodb-1	0.66%	85.86MiB / 31.12GiB	0.27%	200kB / 799B	242kB / 36.9MB	43

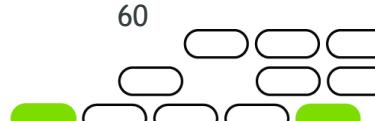
Figura. Contêineres Docker executando serviços de Banco de Dados SQL MariaDB, PHPMyAdmin para gerenciamento do MariaDB e Banco de Dados NOSQL MongoDB.

### • Evidência dos resultados: Modelar Base de Dados

Como esperado de qualquer sistema, um dos itens fundamentais, senão o mais importante, é a sua base de dados. Seja uma base estruturada, do tipo SQL ou uma base documental, NOSQL, trata-se do principal ativo para qualquer sistema, pois é o local onde as informações elaboradas pelos usuários são armazenadas.

Para o sistema de demandas da Papéis Miguel Escobar, pode-se definir duas tabelas como pedras fundamentais: a de usuários e a de demandas. Uma terceira tabela de controle da transferência das demandas entre usuários e setores foi criada, e uma quarta para manter os relatórios.

A tabela de dados de usuário foi criada com um conceito minimalista, somente o necessário para que se possa elaborar um sistema funcional. Os campos que se definiu como imprescindíveis foram o de nome de usuário, matrícula, senha de acesso, setor e papel no sistema de atendimentos - este utilizado para determinar o nível de acesso no sistema, se atendente ou usuário comum.



Já a tabela de dados das demandas necessita maior complexidade de forma a modelar corretamente o cenário desejado. Deve atender aos requisitos mínimos detalhados neste documento e ser de fácil manutenção e expansão, conforme surja a necessidade.

Além disso, temos uma tabela de controle de transferência de chamados entre atendentes e setores. Esta deve manter apontadores tanto para a demanda em questão, bem como para os setores e atendentes envolvidos - cedentes e receptores.

Para as tabelas anteriores, optou-se por utilizar uma abordagem estruturada para a base de dados. Logo, as tabelas de usuários, demandas e transferências se utiliza de uma SGBD do tipo SQL.

Entretanto, há mais um item do sistema que merece atenção e necessita ser guardado, que é a parte de relatórios. É importante que mesmo se apoiando em dados obtidos a partir da base de dados de demandas, seja possível manter em uma base à parte os relatórios emitidos, de forma que cada usuário possa acessar os relatórios que emitiu anteriormente.

Como os relatórios tratam primordialmente de documentos gerados pelo sistema, entende-se que um banco de dados NOSQL atenderia melhor esta situação, por trabalhar muito bem com documentos e dados não estruturados.

Na sequência, apresento as definições de cada tabela no sistema, bem como a criação delas nos respectivos SGBDs - foi utilizado MariaDB com o uso da ferramenta phpMyAdmin para as 3 primeiras tabelas e MongoDB com o uso do MongoDB Compass para a última.

## Tabela de Usuários

Como citado anteriormente, a tabela de usuários precisa somente de campos de nome de usuário, senha, matrícula, papel no sistema e setor do usuário, além da chave primária, obviamente. Foi modelada conforme a tabela abaixo:

users: tabela de usuários	
id	Chave primária da tabela
username	Nome de usuário
password	Senha de acesso do usuário



regnumber	Matrícula do usuário. Deve ser unique.
role	Papel no sistema: atendente ou usuário comum
department	Setor do usuário

Tabela. Tabela de Usuários.

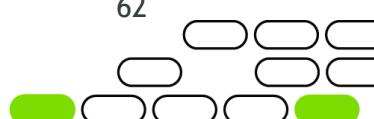
## Tabela de Demandas

Para a tabela que salva as demandas, determinou-se que os campos mínimos necessários para o funcionamento do sistema seriam o do usuário que solicitou a demanda, o atendente, o setor ao qual a demanda será encaminhado, a categorização da demanda, a descrição da solicitação, o status da demanda, a solução dada pelo atendente, a avaliação dada pelo demandante, um campo textual com alguma observação a ser adicionada pelo demandante na hora da avaliação, a data e hora de abertura da demanda, a data e a hora do início do atendimento, a data e hora de fechamento da demanda, um campo indicando se a demanda é uma reabertura de outra não solucionada e, caso positivo, a referência à demanda que está sendo reaberta.

Mesmo em sua forma mais reduzida ela já apresenta uma quantidade bem maior de campos, de modo a representar um conjunto de informações que se averiguou mínimos ao registro de demandas no sistema. Neste caso, há alguns campos que não são indispensáveis a uma abordagem minimalista - avaliação do atendimento e observação (Requisito 11; Caso de uso 6) bem como os campos referentes à reabertura de demandas (Requisito 12; Caso de uso 9), mas que se optou por manter para oferecer estas opções aos atendentes.

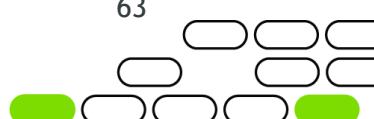
A tabela abaixo descreve cada um destes campos de forma estruturada.

Requests: tabela de demandas	
id	Chave primária da tabela.
User	Usuário que abriu a demanda - referencia a tabela ac_users.
Attendant	Usuário que atende a demanda - referencia a tabela ac_users. Este



	usuário obrigatoriamente deve ter papel de atendente do setor ao qual a demanda foi encaminhada.
Department	O setor ao qual a demanda deve ser encaminhada para atendimento.
Categorization	Categorização da demanda - A partir de lista drop-down apresentada ao abrir o chamado.
Description	Descrição do motivo da demanda, de forma textual.
State	Estado da demanda no sistema. Aberta, Em atendimento, Finalizada.
Solution	Solução oferecida pelo atendente, em modo textual.
Rate	Avaliação numérica dada pelo demandante à solução oferecida pelo Atendente. Esta avaliação é opcional, mas deve ser incentivado ao demandante avaliar cada demanda.
Rate_obs	Observação à avaliação dada pelo usuário demandante, fornecido junto da avaliação da solução. Campo opcional.
Opened_time	Data e horário de abertura da demanda.
Closed_time	Data e horário de finalização da demanda.
Is_reopened	Flag que indica se a demanda é reabertura de outra demanda.
Old_request	Chave estrangeira que aponta para a demanda que foi reaberta, caso seja reabertura de outra demanda.

Tabela. Tabela de Demandas.



## Tabela de Transferência de Demandas

Esta tabela foi desenvolvida devido à necessidade de se lidar com o processo de transferência de chamados entre atendentes. Como especificado nos Requisitos 8 e 9 e descrito nos Casos de uso 7 e 8, o sistema deve oferecer a funcionalidade de alterar o responsável por uma demanda, permitindo que tanto um atendente transfira a demanda a outro atendente (Requisito 8, UC07) quanto autoatribuir uma demanda de responsabilidade de outro atendente (Requisito 9, UC08).

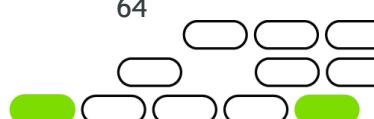
Para esta tabela, se definiram campos que indiquem o atendente atual, o novo atendente, a data e horário da transferência, a demanda em questão e a direção da transferência. A tabela abaixo descreve isto de modo mais detalhado:

transfers: tabela de transferência de demandas	
id	Chave primária da tabela.
user_from	Atendente atualmente responsável pelo chamado.
user_to	Atendente a receber o chamado.
transferred_at	Data e hora em que a solicitação de transferência ocorreu
request	Referência ao chamado a ser transferido.
direction	Indica de onde partiu a transferência. Um atendente pode tentar tanto atribuir a si um chamado quanto enviar o chamado a outro atendente.

Tabela. Tabela de Transferências de Demandas.

## Tabela de Relatórios

Esta tabela, diferente das demais, envolve o uso do SGBD MongoDB, devido à necessidade de armazenar documentos de diversos tipos, gerados pelos relatórios - pdf, xlsx, docx, json, csv e outros que possam ser utilizados na geração dos relatórios. Os campos padrão utilizados são o de nome do arquivo, conteúdo do arquivo e data em que o relatório foi gerado. Mas, por se utilizar um banco de dados NOSQL, a flexibilidade de adicionar ou remover campos é bastante alta, o que permite que, caso se deseje uma mudança nos campos a serem incluídos na hora de salvar um registro,



seja possível incluir novas informações sem necessidade de adequar os registros anteriormente criados.

O *schema* desta tabela foi definido como segue:

reports: tabela de relatórios	
_id	Chave primária da tabela.
filename	Nome do arquivo com a extensão.
content	Conteúdo do arquivo, codificado em base64.
created_at	Data e hora em que o relatório foi armazenado.

Tabela. Tabela de Relatórios.

## Evidências concretas

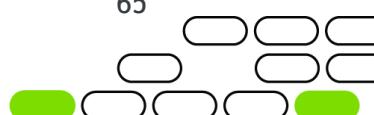
Para evidenciar concretamente a execução de cada etapa da modelagem de dados, trago aqui as imagens de alguns artefatos gerados durante o processo de criação das tabelas nos sistemas envolvidos.

Table	Action	Rows	Type	Collation	Size	Overhead
requests	Browse Structure Search Insert Empty Drop	0	InnoDB	---	64.0 Kib	-
transfers	Browse Structure Search Insert Empty Drop	0	InnoDB	---	64.0 Kib	-
users	Browse Structure Search Insert Empty Drop	0	InnoDB	---	32.0 Kib	-
3 tables	Sum	0	InnoDB		160.0 Kib	0 B

Filters: Containing the word: [ ]

Create new table: Table name [ ], Number of columns [4], Create button

Figura. phpMyAdmin com as 3 tabelas principais SQL.



Server: db > Database: sgta > Table: users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change  Drop  More
2	<b>username</b>	varchar(100)			No	None			Change  Drop  More
3	<b>password</b>	char(64)			No	None			Change  Drop  More
4	<b>regnumber</b>	varchar(10)			No	None			Change  Drop  More
5	<b>role</b>	enum('atendente', 'usuário')			No	usuário			Change  Drop  More
6	<b>department</b>	varchar(100)			No	None			Change  Drop  More

Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

Figura. Estrutura da tabela users.

Server: db > Database: sgta > Table: requests

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change  Drop  More
2	<b>user</b>	bigint(20)		UNSIGNED	No	None			Change  Drop  More
3	<b>attendant</b>	bigint(20)		UNSIGNED	Yes	NULL			Change  Drop  More
4	<b>department</b>	varchar(40)			No	None			Change  Drop  More
5	<b>categorization</b>	varchar(20)			No	None			Change  Drop  More
6	<b>description</b>	longtext			No	None			Change  Drop  More
7	<b>state</b>	enum('Aberta', 'Em atendimento', 'Finalizada')			No	None			Change  Drop  More
8	<b>solution</b>	longtext			Yes	NULL			Change  Drop  More
9	<b>rate</b>	tinyint(4)			Yes	NULL			Change  Drop  More
10	<b>rate_obs</b>	longtext			Yes	NULL			Change  Drop  More
11	<b>created_at</b>	datetime			No	current_timestamp()			Change  Drop  More
12	<b>opened_at</b>	datetime			Yes	NULL			Change  Drop  More
13	<b>closed_at</b>	datetime			Yes	NULL			Change  Drop  More
14	<b>is_reopened</b>	tinyint(1)			No	0			Change  Drop  More
15	<b>old_request</b>	bigint(10)		UNSIGNED	Yes	NULL			Change  Drop  More

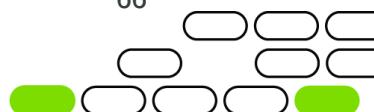
Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

Figura. Estrutura da tabela requests.

Server: db > Database: sgta > Table: requests

Foreign key constraints		Column	Foreign key constraint (INNODB)		
Actions	Constraint properties		Database	Table	Column
Drop	requests_attendant_users_id_	ON DELETE NO ACTION ON UPDATE NO ACTION	attendant	sgta	users id
Drop	requests_oldRequest_request	ON DELETE NO ACTION ON UPDATE NO ACTION	old_request	sgta	requests id
Drop	requests_user_users_id_FK	ON DELETE NO ACTION ON UPDATE NO ACTION	user	sgta	users id
Constraint name:		ON DELETE RESTRICT ON UPDATE RESTRICT			
<a href="#">+ Add column</a>					
<a href="#">+ Add constraint</a>					
<a href="#">Preview SQL</a> <a href="#">Save</a>					

Figura. Relações da tabela requests.



Server: db > Database: sgta > Table: transfers

**Table structure**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
2	<b>from_user</b>	bigint(20)		UNSIGNED	No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
3	<b>to_user</b>	bigint(20)		UNSIGNED	No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
4	<b>transferred_at</b>	datetime			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
5	<b>request</b>	bigint(20)		UNSIGNED	No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
6	<b>direction</b>	enum('enviado','auto atribuido')			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>

[Check all](#) [With selected:](#) [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#) [Spatial](#) [Fulltext](#)

Figura. Estrutura da tabela transfers.

Server: db > Database: sgta > Table: transfers

**Foreign key constraints**

Actions	Constraint properties	Column	Foreign key constraint (INNODB)				
		Database	Table	Column			
<a href="#">Drop</a>	transfers_fromUser_users_id_FK	ON DELETE NO ACTION	ON UPDATE NO ACTION	from_user	sgta	users	<a href="#">id</a>
<a href="#">Drop</a>	transfers_request_requests_id	ON DELETE NO ACTION	ON UPDATE NO ACTION	request	sgta	requests	<a href="#">id</a>
<a href="#">Drop</a>	transfers_toUser_users_id_FK	ON DELETE NO ACTION	ON UPDATE NO ACTION	to_user	sgta	users	<a href="#">id</a>
Constraint name		ON DELETE RESTRICT	ON UPDATE RESTRICT		sgta		
<a href="#">+ Add constraint</a>							

Figura. Relações da tabela transfers.

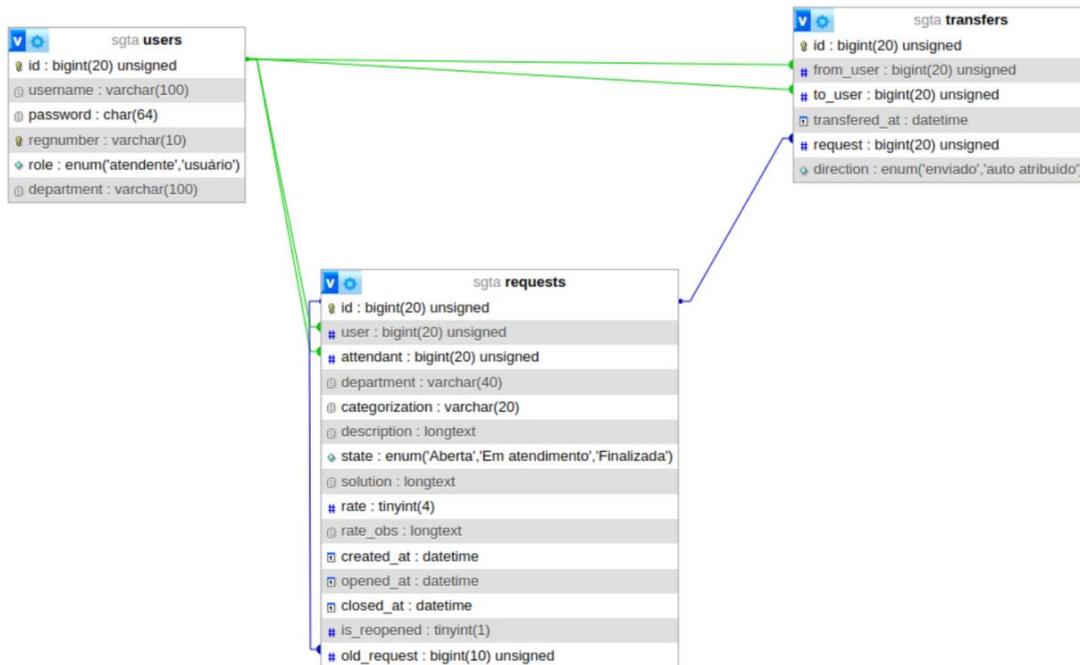


Figura. UML representando os campos e as relações entre as tabelas.

SOTA\_NOSQL > agile > reports SOME-SHAKE

[Documents](#) [Aggregations](#) [Scheme](#) [Indexes](#) [Validation](#)

Type a query: `{field: "value"}` or [generate query](#) \*

**Project** `{ field: 0 }`

**Sort** `{ field: -1 } or { field: 1 }`

**Collection** `{ locales: "simple" }`

**Index Hint** `{ field: -1 }`

[Reset](#) [Analyze](#) [Options](#)

Max Time MS: 400000  
Skip: 0 Unit: 0

This report is based on a sample of 8 documents. [Learn more](#)

_id	content	created_at	filename		
446c304-892f-46d4-94bc-3934fe0a6c2	665db000-3c38-4403-8f88-0edc0f673	af8c340-393-4728-bc0d-d30d240e5d0	0664b2c0-04c-4248-8d2-70e6553c9f2	8abbb49-92b-410c-09f1-82108660209	0362234-e953-4f49-a05b-dec872649
fc7f567b-234f-46c0-a796-0e472467b08	93a0fb9a-2a5f-4768-9f68-05a0fbfb682	Relatório-Gabriel-Atendimentos-20240220.pdf	Relatório-Duarte-Demandas-20240202.docx	Relatório-Celso-Demandas-20240204.pdf	Relatório-demandas-20240204.pdf
		Relatório-Dario-Entregas-20240201.pdf	Relatório-Avaliações-20240201.pdf	Relatório-Entregas-20240101.xlsx	Relatório-Demandas-Amazôn-20240223.pdf

Figura. Schema gerado automaticamente da tabela reports, após a inserção de 8 relatórios dummy no Banco de Dados.



```
import os
import uuid
from pymongo import MongoClient
from datetime import datetime

# Conecte ao MongoDB
client = MongoClient('mongodb://[REDACTED]:[REDACTED]@localhost:27017/')
print(client)
db = client['sgta']
print(db)
collection = db['reports']
print(collection)

# Caminho da pasta com os arquivos
pasta = '[REDACTED]/dummy reports/'

# Função para ler o conteúdo dos arquivos
def ler_arquivo(caminho):
    with open(caminho, 'rb') as file:
        return file.read()

nomes_arquivos = ['Relatorio-demandas-20240916.pdf',
                   'Relatorio-Avaliacoes-20241001.pdf',
                   'Relatorio-Entregas-20240110.xlsx',
                   'Relatorio-Demandas-Armazem-20240923.pdf',
                   'Relatorio-Duarte-Demandas-20240929.docx',
                   'Relatorio-Gabriel-Atendimentos-20240210.pdf',
                   'Relatorio-Celia-Demandas-20240930.pdf',
                   'Relatorio-Dario-Entregas-20240921.pdf']

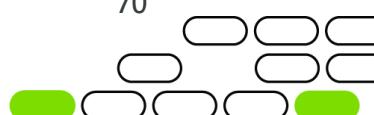
# Iterar sobre os arquivos na pasta
i = 0
for nome_arquivo in os.listdir(pasta):
    caminho_arquivo = os.path.join(pasta, nome_arquivo)
    if os.path.isfile(caminho_arquivo):
        # Gerar UUID
        documento_id = str(uuid.uuid4())
        # Ler conteúdo do arquivo
        conteudo = ler_arquivo(caminho_arquivo)

        # Criar documento JSON
        documento = {
            '_id': documento_id,
            'filename': nomes_arquivos[i],
            'content': conteudo,
            'created_at': datetime.utcnow()
        }
        # Inserir no MongoDB
        collection.insert_one(documento)
        print(f'Inserido: {nome_arquivo} com UUID: {documento_id}')
        i += 1
client.close()
```

Figura. Script utilizado para inserir os 8 relatórios *dummy* na tabela reports.

```
services:  
  db:  
    image: mymariadb  
    restart: unless-stopped  
    ports:  
      - 3306:3306  
    volumes:  
      - [REDACTED] /sgta_pme/db:/var/lib/mysql  
  
    environment:  
      TZ: "America/Sao_Paulo"  
      MYSQL_ROOT_PASSWORD: [REDACTED]  
      MYSQL_DATABASE: sgta  
      MYSQL_USER: [REDACTED]  
      MYSQL_PASSWORD: [REDACTED]  
  networks:  
    - pme  
  
  mongodb:  
    image: mymongo  
    restart: unless-stopped  
    ports:  
      - 27017:27017  
    volumes:  
      - [REDACTED] /sgta_pme/mongo:/data/db  
      - [REDACTED] /sgta_pme/mongods:/datasources  
    environment:  
      TZ: "America/Sao_Paulo"  
      MONGO_INITDB_ROOT_USERNAME: [REDACTED]  
      MONGO_INITDB_ROOT_PASSWORD: [REDACTED]  
  networks:  
    - pme  
  
  phpmyadmin:  
    image: phpmyadmin  
    restart: unless-stopped  
    ports:  
      - 8080:80  
    environment:  
      PMA_HOST: db  
  networks:  
    - pme  
  
networks:  
  pme:  
    driver: bridge
```

Figura. Arquivo docker-compose.yaml, utilizado pelo docker compose para iniciar um grupo de contêineres. No código, temos um contêiner executando o MariaDB, outro executando o MongoDB e um terceiro executando o phpMyAdmin, este último utilizado para facilitar as operações na base de dados MariaDB.



```

/*140101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*140101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*140101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
SET NAMES utf8mb4;
/*140014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*140101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*140111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

CREATE TABLE `requests` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user` bigint(20) unsigned NOT NULL,
  `attendant` bigint(20) unsigned DEFAULT NULL,
  `department` varchar(40) NOT NULL,
  `categorization` varchar(20) NOT NULL,
  `description` longtext NOT NULL,
  `state` enum('Aberta', 'Em atendimento', 'Finalizada') NOT NULL,
  `solution` longtext DEFAULT NULL,
  `rate` tinyint(4) DEFAULT NULL,
  `rate_obs` longtext DEFAULT NULL,
  `created_at` datetime NOT NULL DEFAULT current_timestamp(),
  `opened_at` datetime DEFAULT NULL,
  `closed_at` datetime DEFAULT NULL,
  `is_reopened` tinyint(1) NOT NULL DEFAULT 0,
  `old_request` bigint(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `requests_user_users_id_FK` (`user`),
  KEY `requests_attendant_users_id_FK` (`attendant`),
  KEY `requests_oldRequest_requests_id_FK` (`old_request`),
  CONSTRAINT `requests_attendant_users_id_FK` FOREIGN KEY (`attendant`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `requests_oldRequest_requests_id_FK` FOREIGN KEY (`old_request`) REFERENCES `requests` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `requests_user_users_id_FK` FOREIGN KEY (`user`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;

CREATE TABLE `transfers` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `from_user` bigint(20) unsigned NOT NULL,
  `to_user` bigint(20) unsigned NOT NULL,
  `transferred_at` datetime NOT NULL,
  `request` bigint(20) unsigned NOT NULL,
  `direction` enum('enviado', 'auto atribuido') NOT NULL,
  PRIMARY KEY (`id`),
  KEY `transfers_fromUser_users_id_FK` (`from_user`),
  KEY `transfers_toUser_users_id_FK` (`to_user`),
  KEY `transfers_request_requests_id_FK` (`request`),
  CONSTRAINT `transfers_fromUser_users_id_FK` FOREIGN KEY (`from_user`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `transfers_request_requests_id_FK` FOREIGN KEY (`request`) REFERENCES `requests` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `transfers_toUser_users_id_FK` FOREIGN KEY (`to_user`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;

CREATE TABLE `users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL,
  `password` char(64) NOT NULL,
  `regnumber` varchar(10) NOT NULL,
  `role` enum('atendente', 'usuário') NOT NULL DEFAULT 'usuário',
  `department` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_unique` (`regnumber`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;

```

Figura. Código SQL para criação das Tabelas no MariaDB, definindo campos, chaves e relações.

## 2.2.2 Lições Aprendidas

- Seguindo com os desafios desta etapa, verifiquei que ter os requisitos definidos e enumerados serviu como grande balizador na hora de desenvolver os casos de uso do sistema. Foi um apoio importantíssimo, embora o esforço para a elaboração deles tenha sido bastante elevado.
- O aprendizado de uso do Draw.io foi fundamental para a criação dos Diagramas de Casos de Uso. Esta etapa foi bastante facilitada com os conhecimentos anteriores de uso da ferramenta.
- O uso de containerização de aplicações, com o apoio de docker e docker compose se mostrou uma decisão que facilitou bastante para executar os serviços de Banco de Dados necessários à criação das tabelas do sistema.



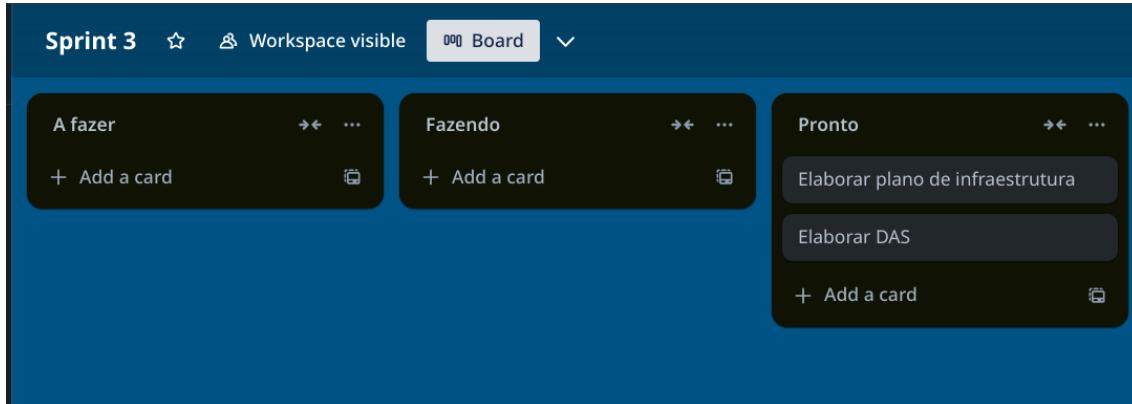
- Ferramentas já antigas no mercado como o phpMyAdmin se mostraram bastante eficazes para a geração das tabelas no MariaDB. Além desta ferramenta, também foi utilizado o AntaresSQL, uma ferramenta Desktop para trabalhar com Banco de Dados que facilita a definição de tabelas e geração de código SQL para clonagem de um banco de dados dentro do MariaDB.
- A decisão de se utilizar um script Python para criar dados fictícios de relatórios dentro do MongoDB se deu dada à facilidade de se gerar códigos para automatizar tarefas e à existência de bibliotecas que simplificam a inserção de dados no MongoDB. Sendo o MongoDB um Banco de Dados NOSQL, não é possível definir um *schema* fixo de campos a serem utilizados e com isto, para demonstrar os tipos de dados esperados a serem armazenados na tabela de relatórios, entendi ser interessante criar dados fictícios no sistema, até para apoiar o desenvolvimento de uma aplicação real oferecendo informações importantes para a equipe de desenvolvimento.



## 2.3 Sprint 3

### 2.3.1 Solução

- Evidência do planejamento:



- Evidência da execução de cada requisito: Elaborar plano de infraestrutura

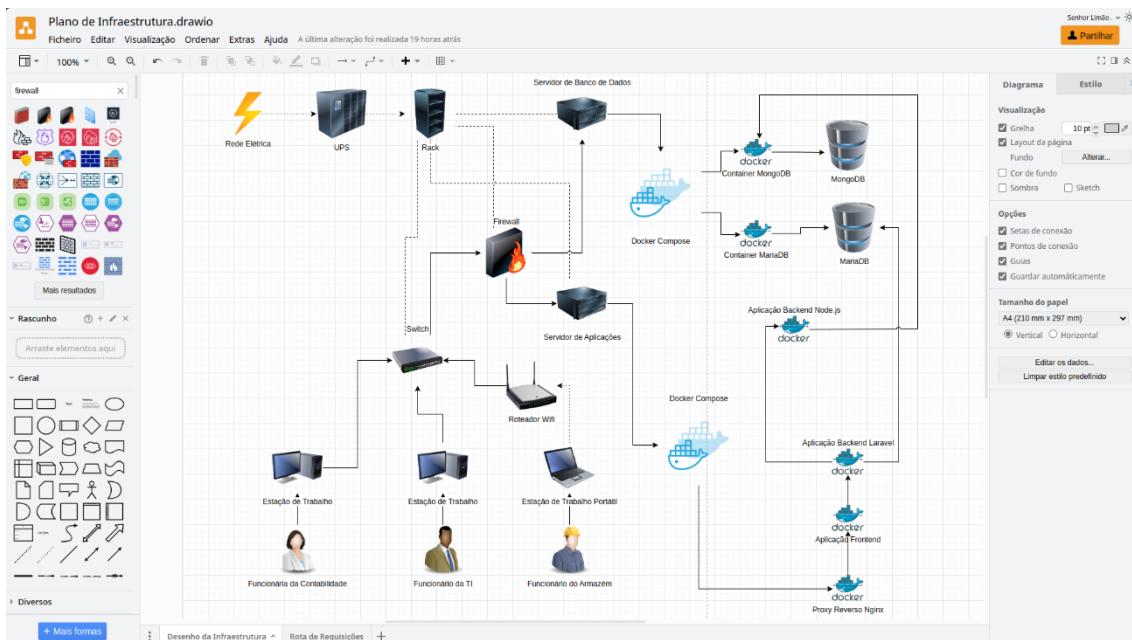
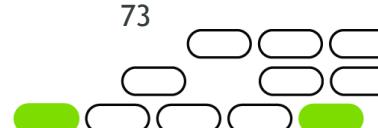


Figura. Tela do Draw.io, com a elaboração de figura representando graficamente a infraestrutura do sistema.

- Evidência dos resultados: Elaborar plano de infraestrutura

Aqui elaboramos um plano de infraestrutura para suportar a execução da aplicação, unindo tecnologias modernas, porém voltadas para um ambiente on-



premises. Foi definido que a infraestrutura deve ter custo baixo, fácil manutenibilidade, deve se apoiar em tecnologias com mão de obra farta no mercado e com desempenho adequado.

Com base nestas premissas, foi determinado que alguns elementos devem fazer parte da infraestrutura, de modo a se adequar a estas premissas:

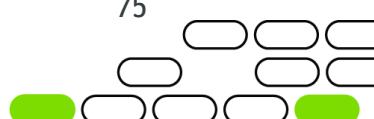
- Devem ser utilizados ao menos 2 equipamentos atuando como servidores de aplicações - um para a execução da aplicação e outros para os bancos de dados - SQL e NOSQL.
  - Neste caso, entende-se que a melhor abordagem seria ter uma separação entre os bancos de dados, porém há limitação de custos, o que em um primeiro momento impede a aquisição de um terceiro equipamento dedicados a isto.
  - A estratégia de virtualização poderia ser adotada, porém o sistema atual de subscrições oferecido pela Broadcom para o Vmware tem se mostrado muito onerosa. Outra opção seria virtualização utilizando Xen Project, que é um Hypervisor grátis e de código aberto, porém as experiências que tive previamente com este sistema acarretaram alguns insucessos e perda de dados. Em ambiente de produção não é possível lidar com a incerteza de estabilidade do sistema, logo, foi descartado.
  - Avalia-se a possibilidade de futuramente migrar os bancos de dados para ambientes em nuvem, primeiramente o NOSQL e posteriormente o SQL. Porém, não é o objetivo neste momento - dado que a empresa ainda tem um porte pequeno, a tendência de crescimento destas bases não é muito elevada, o que permite que se mantenha somente um equipamento servindo ambas sem prejuízo significativo no desempenho.
- Cada servidor de aplicações irá executar um sistema operacional linux para servidor. Os principais candidatos são Ubuntu Server ou Alma Linux, tendendo mais ao primeiro devido à maior base ativa do sistema e à possibilidade de contratação de suporte especializado, caso seja necessário.
- As aplicações - frontend, backend Laravel, backend Node.js, SGBD MariaDB e SGBD MongoDB serão executadas utilizando containerização, com o uso do Docker. Dada a simplicidade do sistema em seu estágio inicial, optou-se por realizar a integração entre os contêineres utilizando o Docker Composer ao invés do Kubernetes, porém esta decisão pode ser revista futuramente.
- Como informado, o backend da aplicação se apoiará tanto na tecnologia PHP com Laravel como framework quanto Javascript com Node.js. São tecnologias que possuem bastante mão de obra disponível no mercado e oferecem segurança, estabilidade e confiabilidade para a execução das aplicações.
  - O uso de Laravel como framework para a parte da aplicação que lida com o banco de dados SQL se dá devido a este possuir nativamente uma série de validações e verificações de segurança para acesso às bases de dados.
  - O uso de Node.js para as operações envolvendo a integração com o NOSQL se dá devido a 2 fatores principais: a integração com MongoDB é bastante madura e a integração do framework Laravel com o MongoDB



é deficitária. Assim sendo, adicionou-se esta camada de modo a agilizar o desenvolvimento da aplicação

- o O acesso à camada de aplicação Node.js se dará unicamente a partir do backend Laravel. A aplicação Node.js irá recusar qualquer tentativa de acesso proveniente de outro lugar que não seja o servidor de aplicação Laravel. O servidor Laravel recebe todas as requisições de APIs e redireciona as que sejam relacionadas a relatórios.
- Para o frontend da aplicação, será utilizado o Next.js como framework. Este se apoia em outro framework Javascript, o React. A opção pelo uso do Next.js se dá devido a este implementar diversas melhorias em relação ao React, como roteamento automático de endereços sem configurações adicionais, por exemplo. As funcionalidades de backend oferecidas pelo Next.js não serão utilizadas.
- Será utilizado um proxy reverso para acesso ao sistema pelo usuário
  - o Dada a popularidade e reconhecido desempenho, o NGINX será utilizado para tal.
  - o Além de funcionar como proxy reverso, o NGINX é um excelente平衡ador de carga, o que permitirá que ele atue também deste modo conforme a aplicação precise escalar, bastante ajustar o arquivo de configuração e reiniciar o serviço.

Foi possível elaborar os seguintes diagramas, utilizando Draw.io, para ilustrar melhor as ideias a serem implementadas para a infraestrutura do sistema.



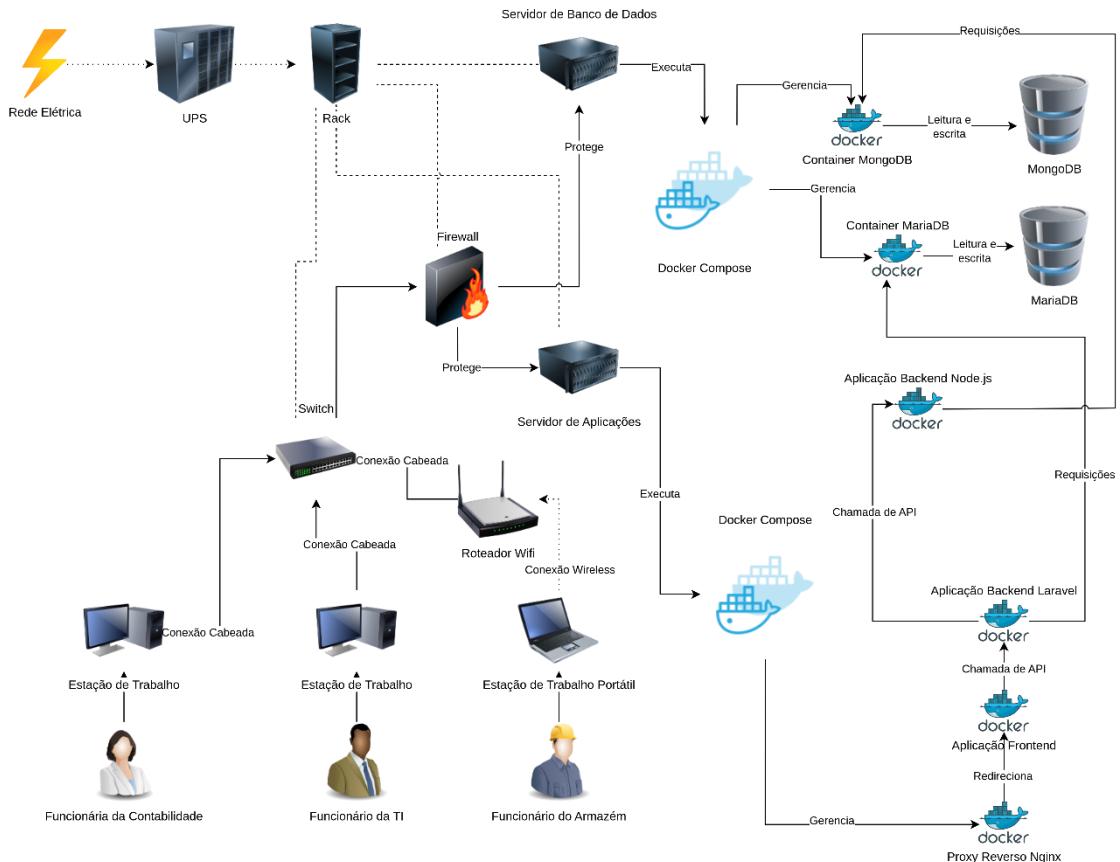


Figura. Diagrama representando a infraestrutura operacional para o sistema da Papéis Miguel Escobar. A parte referente aos funcionários está subdimensionada, servindo somente como exemplo dos tipos de usuários que se pretende ter no MVP - um usuário comum, representado pela Funcionária da Contabilidade, um atendente do setor de TI e um atendente do setor de Armazém.

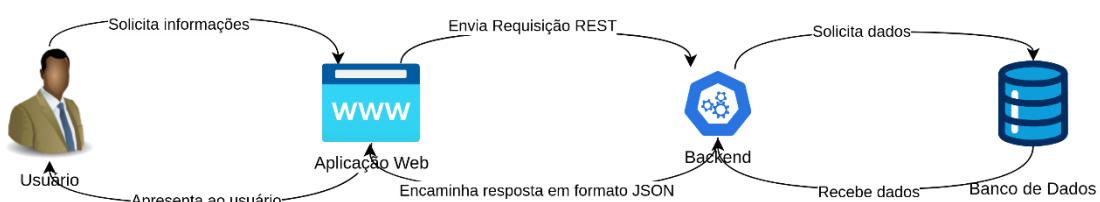


Figura. Rota das requisições no aplicativo, com os principais papéis envolvidos, o usuário, a aplicação que o usuário utiliza, o backend da aplicação e o banco de dados.

- Evidência da execução de cada requisito: Elaborar DAS

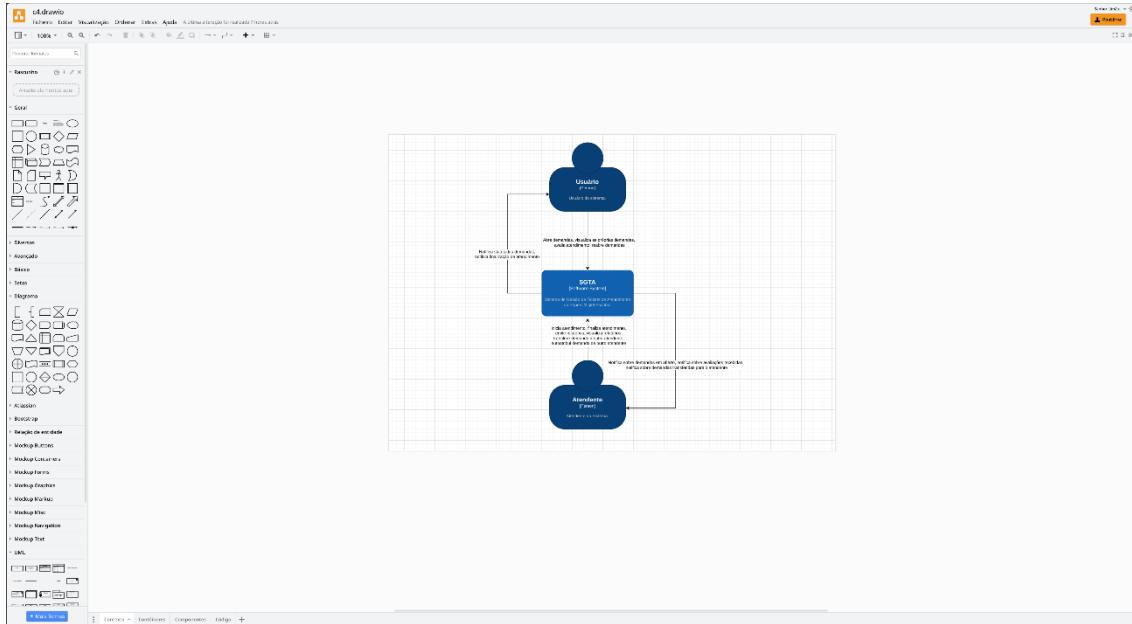


Figura. Elaboração dos Diagramas do C4 Model utilizando o Software Draw.io. Na figura, a aba referente ao Diagrama de Contexto.

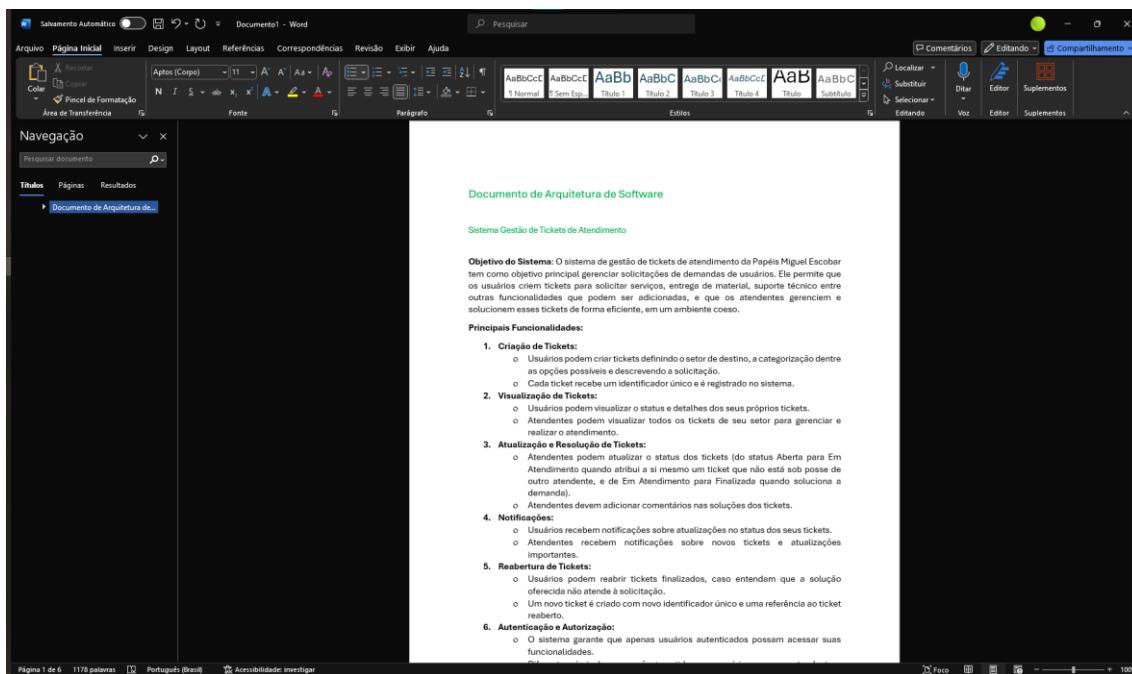


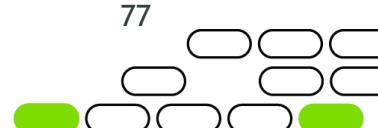
Figura. Elaboração do Documento de Arquitetura de Software.

- Evidência dos resultados: Elaborar DAS

O Documento de Arquitetura de Software elaborado pode ser encontrado no repositório público do Github a seguir:

<https://github.com/gfidelisdev/projetoaplicadoxpe>

Mais evidências seguem abaixo.



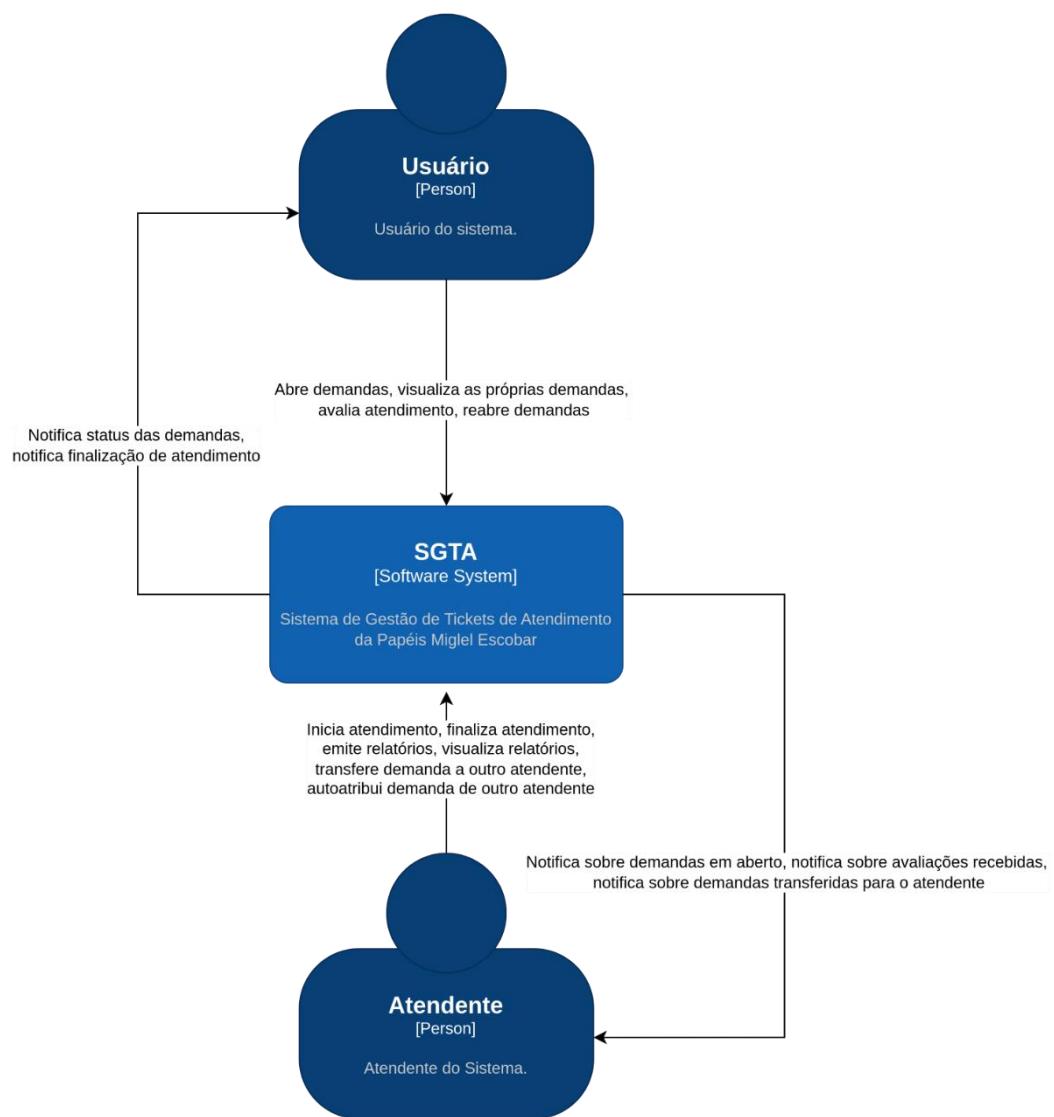


Figura. Diagrama de Contexto do DAS.

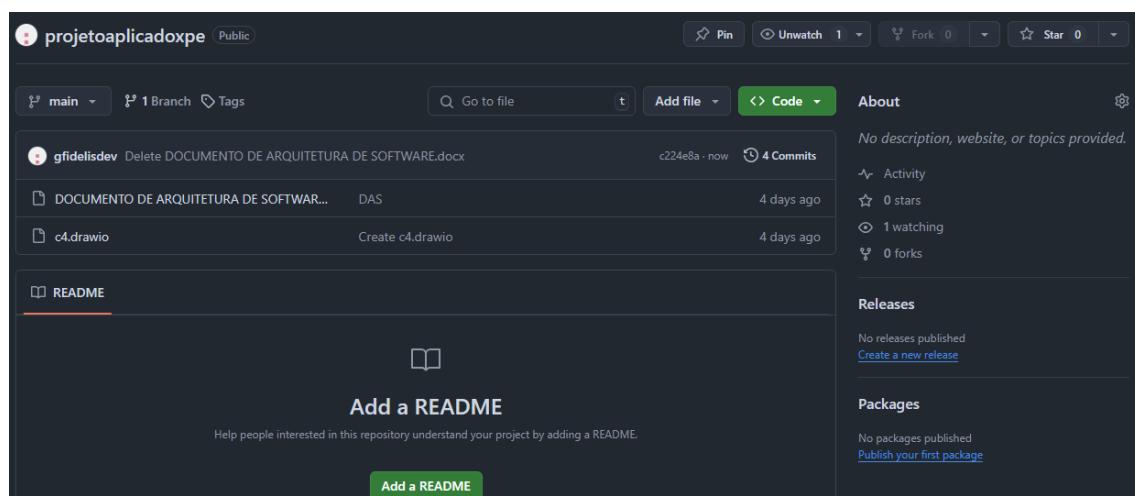
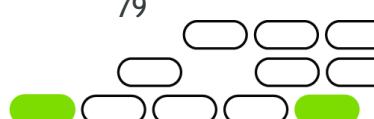


Figura. Repositório no Github com o DAS disponível em formato .pdf, bem como os diagramas C4 produzidos no Draw.io.

### 2.3.2 Lições Aprendidas

Esta *Sprint* se mostrou a mais desafiadora de todas, por se utilizar de ferramentas às quais não estou acostumado a utilizar em projetos. Dentre os principais desafios enfrentados, cito:

- Utilização do C4 Model como modelo de abstração, que embora tenha sido apresentado durante os *bootcamps*, tive que estudar mais a fundo para elaborar os diagramas e documentação necessária.
- A Elaboração de um Documento de Arquitetura de Software foi uma tarefa que exigiu bastante pesquisa, tempo e dedicação para que o resultado se mostrasse satisfatório.
- Também a elaboração do Plano de Infraestrutura se mostrou bastante complexo, sendo necessário avaliar os principais riscos envolvidos e todos os componentes necessários ao perfeito funcionamento do sistema mesmo nas fases iniciais de desenvolvimento, que já exigem grande parte do que foi estabelecido mesmo que com alguns níveis de abstração.
- Foi necessário reconstruir algumas vezes os documentos elaborados de cada item da *Sprint*, pois em vários pontos havia similaridade entre o que era abordado. Embora a redundância não tenha sido totalmente eliminada, por ser necessária para contextualizar as tarefas, esta foi bastante reduzida após refatoração dos documentos.
- Isso vale para tarefas das sprints anteriores, onde foi necessário buscar alguns elementos já mencionados - especialmente na definição de requisitos - para que o DAS elaborado fizesse sentido.



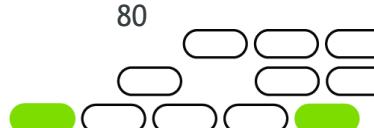
### 3. Considerações Finais

#### 3.1 Resultados

Posso elencar os principais pontos positivos como os que seguem:

- Foi possível exercitar os conhecimentos que eu já possuía sobre desenvolvimento de projetos, e esta prática é sempre importando para um aperfeiçoamento contínuo das habilidades profissionais.
  - Mesmo em relação aos conhecimentos prévios, havia tarefas que precisei realizar que não praticava há bastante tempo, servindo como um segundo aprendizado.
- Referente à parte de Desafio e Solução, grande parte dos elementos utilizados, como *personas*, *blueprints*, objetivo SMART são elementos aos quais eu tinha pouca familiaridade - até o momento não havia utilizado profissionalmente.
  - O aprendizado do uso destas ferramentas sempre pode auxiliar, mesmo que não façam parte de algum projeto. É possível abstrair ideias a partir dos conceitos envolvidos e com isto acelerar o processo de definição de arquitetura de sistemas.
- A execução de tarefas utilizando metodologia ágil era algo ainda desconhecido para mim. O uso do ferramental de *backlogs* e *sprints* é outro conhecimento que adicionei à minha lista de habilidades.
- A atenção e dedicação exigidas para o desenvolvimento dos diagramas e documentos ajudou a melhorar o meu gerenciamento de tempo, em que tive que reorganizar os tempos de estudo e criação de artefatos de modo a conseguir realizar as entregas.
- Os documentos gerados servirão de grande ajuda para as próximas etapas do desenvolvimento do sistema, oferecendo uma documentação concisa, clara e objetiva para que o projeto siga o seu fluxo de desenvolvimento e implantação, incluindo todas as etapas do processo, envolvendo desde a infraestrutura necessária até as tecnologias utilizadas para o software.

Como pontos negativos, posso citar os seguintes:



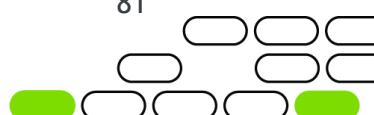
- Dificuldade inicial com a elaboração de artefatos aos quais eu tinha pouquíssima intimidade - como a elaboração de personas, por exemplo. Por não se tratar de uma metodologia de trabalho que aplico no dia a dia, foi bastante exaustivo o esforço em criar cada uma das personas e elaborar as características de cada uma.
- Pouca familiaridade com as metodologias ágeis, o que me fizeram criar tarefas de sprints bastante extensas. Em uma análise mais profunda, me parece que as tarefas que determinei poderiam ser quebradas em subtarefas menores.
- Outro ponto negativo que eu posso citar é da própria ferramenta que estou utilizando para elaborar o relatório - Word Online. Como utilizo sistema operacional Linux, não há versão desktop disponível, e a versão online do Word tem muitas limitações que pude constatar.
  - Um exemplo de limitação forte é que a atualização do sumário na versão online não é nem um pouco satisfatória.
  - O uso do Libreoffice poderia ser uma solução, porém já tive muitas experiências em que a formatação do documento era totalmente quebrada ao abrir os documentos nesse editor de textos. As limitações do Word online são menos prejudiciais do que o retrabalho que seria necessário utilizando o Libreoffice.
- As limitações de porte da empresa de Papéis Miguel Escobar não permitem que se tenha uma equipe de implantação grande, o que deve acarretar dificuldades técnicas elevadas durante as próximas etapas do projeto.

Dentre as experiências vivenciadas, posso elucidar os seguintes itens:

- Aplicações de padrões de desenvolvimento alinhados com regras de UX/UI.
- Elaboração de documentação técnica visando clareza e direcionamento à equipe de desenvolvimento.
- Aplicação de princípios de metodologia ágil na definição de tarefas e prazos.
- Foco na execução das sprints obedecendo aos prazos definidos.

Por fim, obtivemos alguns bons resultados:

- A coleta de requisitos permitiu obter insights valorosos de como o sistema deve funcionar.
- Temos um banco de dados modelado - e implementado - com as tabelas necessárias à elaboração de um MVP.



- Os casos de uso servirão de fundamental guia para o desenvolvimento das funcionalidades do sistema.
- A visão geral do produto complementa de modo detalhado os casos de uso, e oferece à equipe de UX/UI uma visualização gráfica de como são esperadas as telas do sistema.
- O plano de infraestrutura define as tecnologias a serem utilizadas, que são de ampla utilização no mercado, facilitando encontrar profissionais para o desenvolvimento e manutenção. São tecnologias bastante testadas e que se apoiam nos princípios de boas práticas, garantindo manutenibilidade, segurança e confiabilidade, dentre outros.
- O documento de arquitetura de software oferece um resumo de tudo o que foi visto, servindo como um guia de fácil consulta para as partes envolvidas no desenvolvimento do sistema.

### 3.2 Contribuições

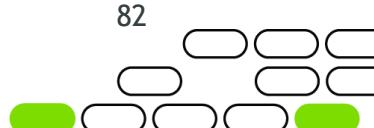
O cenário encontrado na Papéis Miguel Escobar ao início do projeto era de desorganização total nos métodos utilizados para a solicitação de quaisquer demandas entre setores.

A avaliação realizada era de que os meios utilizados eram muito ineficientes - as solicitações eram realizadas de diversas maneiras, e não havia qualquer registro formal das demandas realizadas. Isto fazia com que diversas demandas simplesmente não fossem atendidas imediatamente, necessitando novas solicitações até que houvesse uma solução.

Como não havia registro nem por parte dos demandantes nem por parte dos atendentes, não era possível garantir que a solicitação era realizada, nem havia meios de se obter dados sobre as demandas existentes entre setores.

A oportunidade de tratar desta situação foi o que motivou o desenvolvimento do sistema de Gestão de *Tickets* de Atendimento. Com o objetivo de padronizar as solicitações entre setores e com isto melhorar a comunicação e eficiência na solução de demandas identificadas, o sistema solucionará grande parte das falhas nos atendimentos da empresa.

O alcance deste objetivo é fundamental para que a empresa possa seguir crescendo. Com uma implantação gradual das funcionalidades, será possível sanar eventuais dificuldades encontradas e tornar o sistema um ponto de controle operacional completo, servindo tanto para controle de solicitações de suporte técnico quanto controle de vendas e entregas - inclusive o armazém que é responsável pela entrega



de produtos aos clientes será, junto com o departamento de TI, um dos dois setores a utilizar o MVP no papel de atendente.

Claro, os resultados obtidos dependem dos próximos passos. O documento que aqui foi produzido serve como referência arquitetural para o sistema a ser desenvolvido, e há ainda um caminho bastante difícil a ser trilhado para que seja possível oferecer um MVP para a empresa. O restante da rota necessária para atingir o objetivo está listado na seção seguinte, “Próximos passos”.

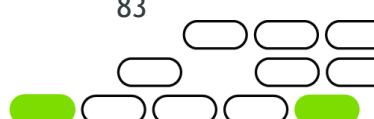
Mas o interesse demonstrado tanto pela diretoria quanto pelos colaboradores da Papéis Miguel Escobar em obter de modo operacional o sistema é um grande incentivo a se dar sequência ao desenvolvimento do produto. Houve bastante colaboração da equipe da empresa durante os processos de coleta de requisitos, situação que facilitou o trabalho nestas etapas iniciais. Por isto, há grande expectativa de que os primeiros protótipos a serem testados pelos usuários se tornem de fundamental relevância ao processo de construção do MVP e das melhorias contínuas do sistema, dado que é esperado o mesmo nível de comprometimento da equipe conforme as etapas forem avançando.

Assim, tendo os artefatos gerados neste documento como guias e pedra fundamental para a construção do sistema, resta determinar a sequência de tarefas para seguir o desenvolvimento.

### 3.3 Próximos passos

Descreva quais são os próximos passos que poderão contribuir com o aprimoramento da solução apresentada pelo seu Projeto Aplicado.

- Implantação de MVP do sistema.
  - Utilizando os artefatos desenvolvidos até o momento, será criada uma primeira versão do sistema que permita aos usuários realizarem demandas e aos atendentes oferecerem soluções.
  - Por se tratar de um MVP, alguns dos requisitos não estarão ainda disponíveis no dia 1.
    - Somente abertura e fechamento de demandas estarão disponíveis. Transferência de demandas entre atendentes, geração de relatórios, reabertura de demandas e avaliação do atendimento serão incluídas conforme estejam funcionais.
    - O MVP será pouco mais que um protótipo do sistema, pois os dados inseridos serão incluídos de modo permanente na base de dados de produção.
- Coleta de *feedbacks* junto ao cliente, visando inclusão de melhorias.



- Conforme os usuários e atendentes começarem a utilizar o MVP, será solicitado a eles que informem à equipe de desenvolvimento quais aspectos do sistema devem melhorar e quais funcionalidades são mais desejáveis.
- Com estes dados em mãos, será possível priorizar a inclusão de funcionalidades.
- Requisitos que ainda não estejam implementados terão maior priorização, mas boas ideias que não foram incluídas nos requisitos podem “furar a fila” caso se julgue que agregam mais ao sistema.
- Melhoria contínua do sistema, com inclusão gradual de outros setores como atendentes.
  - Com base na filosofia DEVOPS, o sistema será continuamente melhorado.
  - Os feedbacks coletados a partir do uso do sistema, após priorizados, serão incluídos no backlog do produto.
  - A inclusão gradual de outros setores permitirá que o impacto de uso do sistema seja suave. Dificuldades enfrentadas pelos usuários e aspectos de funcionamento dos setores que não foram previstos podem ser contornados de modo mais fácil desta maneira.

