



igti

RELATÓRIO

PROJETO APLICADO

Instituto de Gestão e Tecnologia da Informação
Relatório do Projeto Aplicado

Proposta de um sistema de
campanhas de *phishing* baseado em
uma política de base conceitual
behaviorista

Guilherme da Franca Batista

Orientador: Professor Maximiliano Jacomo

2022



GUILHERME DA FRANCA BATISTA

INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO

RELATÓRIO DO PROJETO APLICADO

PROPOSTA DE UM SISTEMA DE CAMPANHAS DE *PHISHING* BASEADO EM UMA POLÍTICA DE BASE CONCEITUAL *BEHAVIORISTA*

Relatório de Projeto Aplicado
desenvolvido para fins de conclusão do
curso de MBA em Segurança Cibernética.

Orientador: Professor Maximiliano
Jacomio

Guarulhos

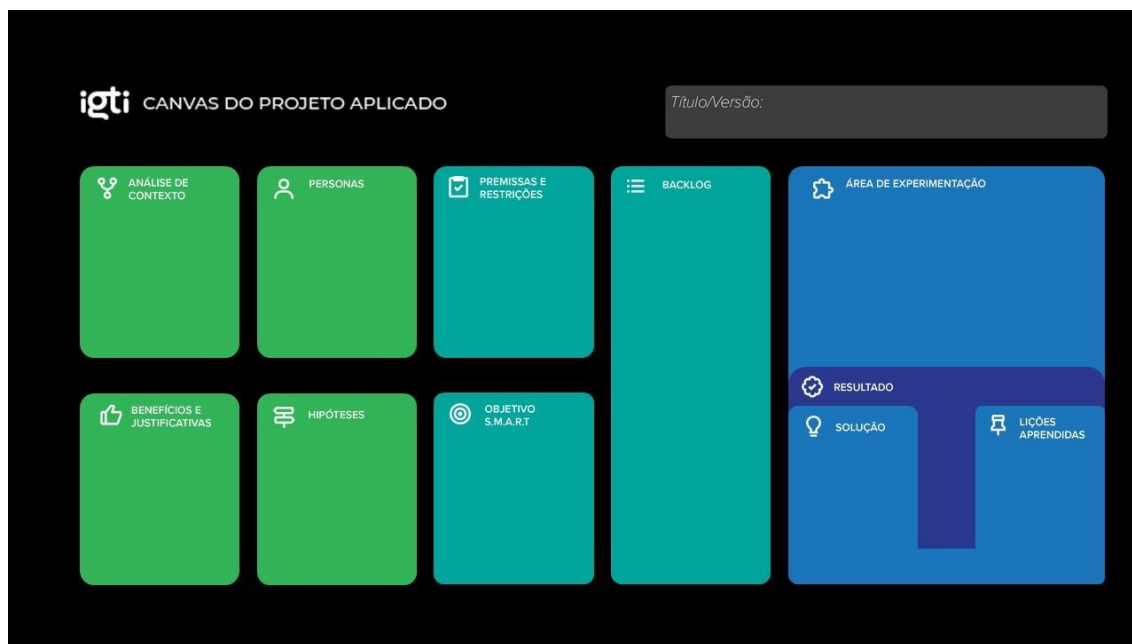
2022

Sumário

1. CANVAS do Projeto Aplicado	4
1.1 Desafio	5
1.1.1 Análise de Contexto	5
1.1.2 Personas	9
1.1.3 Benefícios e Justificativas	12
1.1.4 Hipóteses	16
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	23
2.1 Sprint 1	23
2.1.1 Solução	23
• Evidência do planejamento:	23
• Evidência da execução de cada requisito:	25
• Evidência dos resultados:	28
2.1.2 Experiências vivenciadas	37
2.2 Sprint 2	38
2.2.1 Solução	38
• Evidência do planejamento:	38
• Evidência da execução de cada requisito:	40
• Evidência dos resultados:	45
2.2.2 Experiências vivenciadas	55
2.3 Sprint 3	56
2.3.1 Solução	56
• Evidência do planejamento:	56
• Evidência da execução de cada requisito:	60
• Evidência dos resultados:	62
2.3.2 Experiências vivenciadas	69
3. Considerações Finais	70
3.1 Resultados	70
3.2 Contribuições	70
3.3 Próximos passos	71

1. CANVAS do Projeto Aplicado

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



1.1 Desafio

1.1.1 Análise de Contexto

Há cerca de setenta e um anos atrás, Presper Eckert e John Mauchly, engenheiros da Universidade da Pensilvânia, entregaram ao governo americano o *Universal Automatic Computer I (Univac-I)* para que o Departamento de Censo dos Estados Unidos da América pudesse realizar o monitoramento do *Baby Boom*¹. Nesta época, apesar de estas máquinas estarem sendo usadas em larga escala pelos setores civil e militar do governo americano e por outras grandes corporações, as pessoas ainda não poderiam vislumbrar o que haveria de vir em pouco tempo. No domínio da literatura, um dos criadores do gênero *cyberpunk*, William Gibson, em seu romance *Neuromancer*, conseguiu, ainda em 1984, ter um vislumbre do futuro, criando a ideia do cyberspaço que consiste um espaço virtual composto por cada computador e usuário conectados em uma rede mundial. Desde a década de 90, a evolução de hardware e software, seguindo as leis de *Moore*² e os saltos qualitativos observados por Brooks³, foi cada vez mais rapidamente transformando o mundo, aproximando as pessoas, criação de modelos de negócio completamente novos e novos hábitos na sociedade através da evolução tecnológica das redes e dispositivos computacionais cada vez mais acessíveis e simples de serem utilizados pela população mundial. Esta nova era do mundo digital trouxe novas oportunidades e com certeza muitos desafios, como a da segurança cibernética para o contexto empresarial e pessoal.

No início dos anos 2000, a primeira grande ameaça em forma de *phishing* contra um banco foi realizada⁴ e esse tipo de atividade criminosa foi, ao longo dos anos se tornando mais comuns e ficando cada vez mais fidedignas. A infração de enganar pessoas para que estas compartilhem informações pessoais como senhos, números de cartão de crédito e XPTO não é nova. O termo foi cunhado em 1987 em um artigo e apresentação da *International HP Users Group* e supõe-se que esta prática ocorre desde a década de 60. Estes ataques não possuem apenas uma única categoria de pessoas alvo, como bancários, industriais, comerciantes ou zeladores, eles são

¹ Termo que se refere a explosão demográfica entre os anos 1946 e 1964 nos EUA.

² Lei/observação feita por Gordon Earle Moore em 1965 que consiste no aumento de cem por cento dos transistores dos chips, pelo mesmo custo, a cada dois anos.

³ Referimo-nos ao artigo *No Silver Bullet - Essence and Accident in Software Engineering* publicado por Frederick Phillips Brooks Jr em 1987 pela Universidade da Carolina do Norte.

⁴ No início dos anos 2000 sistemas de pagamento foram o grande foco de ataques de larga escala por *phishing*. Softwares, como o *Turnkey*, foram disponibilizados no mercado negro e a *Gartner* estima que cerca de 3.6 milhões de pessoas perderam 3.2 bilhões de dólares em um período de um ano.

enviados para pessoas de variados níveis sociais e culturais com o objetivo único de ganhar vantagem sobre as pessoas.

Um fato extraordinário aumentou bastante o número de ataques cibernéticos de modo geral, o advento da pandemia de *COVID-19* em dezembro de 2019. Após decretos de *lockdowns* por potências estrangeiras e políticas de confinamento em território nacional, a sociedade precisou se adaptar e digitalizar o máximo de atividades presenciais e manuais possível para que o mínimo da parcela da população precisasse deixar seus lares e assim evitar o contágio da nova variante *SARS-CoV*. Assim sendo, muitas empresas adotaram o trabalho remoto, implantando de forma rápida e muitas vezes insegura as *VPN's* e infraestruturas necessárias para esta nova realidade e em muitas dessas ocasiões o treinamento necessário para adoção de boas práticas e mitigação das ameaças cibernéticas foram negligenciadas.

Assim sendo, neste cenário de uma sociedade cada vez mais conectada à rede mundial de computadores, negócios cuja sobrevivência está estritamente ligada a seus ativos digitais e a privacidade e segurança de pessoas e empresas em constante risco de violação, o desafio deste projeto aplicado é de propor um sistema de gerenciamento de campanhas de *phishing* com uma base sólida, especificamente da psicologia comportamental ou behaviorismo, para que os colaboradores das organizações que possuem restrições financeiras para a contratação de serviços deste tipo ou implantação de sistemas complexos e de alto custo possam ter acesso a software livre e uma base sólida para a criação dos testes, acompanhamento dos resultados e engajamento dos envolvidos além da possibilidade de extrair *insights* e propostas com mais qualidade.

Matriz CSD

Aspirando a uma melhor compreensão do cenário e do problema apresentado a este projeto aplicado, seguir-se-á na apresentação do artefato proposto nesta seção, a saber, a matriz CSD, cujo acrônimo significa Certezas, Suposições e Dúvidas, uma técnica simples na qual três ângulos importantes sobre um determinado projeto são listados de modo a auxiliar na obtenção de informações necessárias que proporcionam o esclarecimento de ideias, bem como o melhor entendimento das partes envolvidas. Sua aplicabilidade se faz por meio de uma representação visual - um quadro ou tabela - em que durante a confecção inicial do projeto os envolvidos possam preencher as certezas, suposições e dúvidas presentes no projeto e inerentes ao problema no qual busca-se uma solução.

	Certezas	Suposições	Dúvidas
Atores	Colaboradores estão expostos a ameaças providas de <i>phishing</i> a todo momento.	Realizar uma pesquisa teórica e empírica sobre a taxonomia dos diversos tipos de <i>phishing</i> pode ser viável.	Quais são as formas mais e menos comuns de ataques a empresas através de <i>phishing</i> ?
Cenário	Todo colaborador é um potencial vetor para ataques à organização a qual prestam serviços.	Colaboradores são pessoas e, assim sendo, estão sujeitos a manipulações de caráter psicológico criadas por criminosos cibernéticos.	Como evitar que os trabalhadores sejam vítimas dos ataques ou chegar mais próximo da mitigação desse risco?
Regra	Definir um modelo conceitual behaviorista para que um sistema de campanhas de <i>phishing</i> seja implementado.	Conhecer modelos tradicionais da psicologia comportamental (Watson e Skinner) e ferramentas técnicas que viabilizem a construção do sistema.	Qual seria o melhor modelo psicológico para tomar como base e quais ferramentas são as mais indicadas para a construção do sistema?

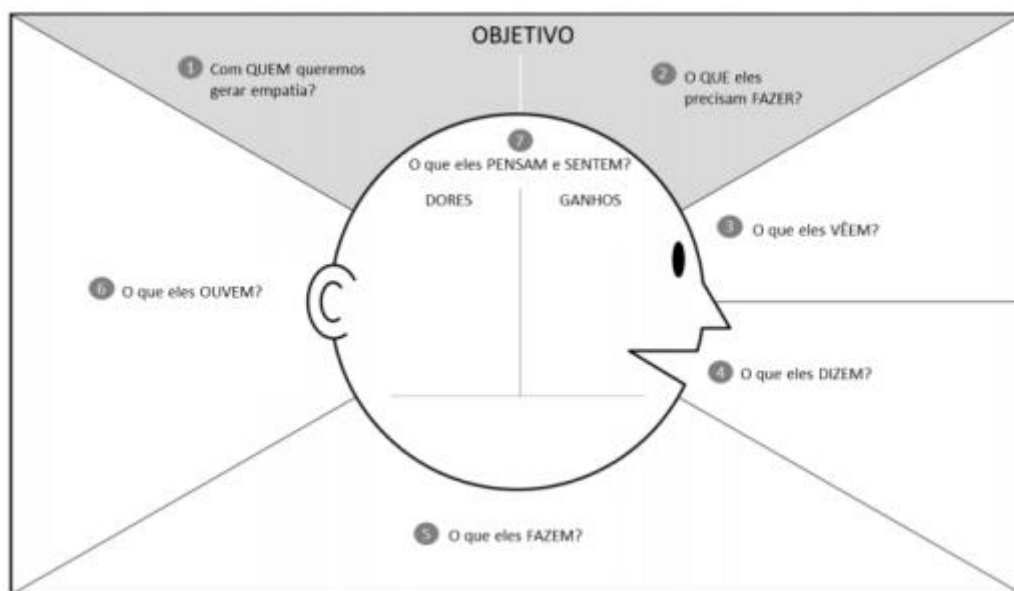
Observação do tipo POEMS

Para que o desafio deste projeto aplicado possa ser concluído, a utilização do POEMS (Pessoas, Objetos, Ambiente, Mensagem e Serviço), um *framework* que tem como objetivo principal orientar e estruturar toda a pesquisa deste trabalho acadêmico, também será utilizado, tornando mais fácil sintetizar as observações feitas por mim bem como a identificar as correlações e contrastes existentes no desafio e em todo o projeto aplicado.

Pessoas	Objetos	Ambientes	Mensagem	Serviços
Quem está presente no contexto da análise?	Que objetos fazem parte do ambiente?	Quais são as características do ambiente?	Que mensagens são comunicadas?	Quais são os serviços oferecidos?
Alta administração da empresa; Colaboradores da área de TI (<i>Blue e Red Team's</i> se houver)	Sistemas operacionais das estações de trabalho; Servidores que mantêm as aplicações da empresa.	Usuários internos acessando sistemas internos da empresa.	Logs de sistemas operacionais das estações; Logs das aplicações corporativas	Sistemas e aplicações da empresa na internet e intranet.
Registros			Insights	
Resultados obtidos após relatórios apresentados pela área de T.I. da empresa após análise inicial de vulnerabilidade. Lições que serão aprendidas no decorrer das sprints.			Por enquanto, ainda não foi possível ter insights.	

1.1.2 Personas

Nesta seção apresentaremos as pessoas envolvidas diretamente no problema apresentado, definindo as características pessoais, sociais, intelectuais e profissionais, de acordo com o mapa de empatia e suas seções.



Mapas de Empatia

Os mapas de empatia pensados para este trabalho são no total de três. O primeiro deles refere-se à alta administração da organização que tomará decisões importantes na adoção ou não do sistema proposto e também são em última instância os mais impactados pelo tipo de ataque que o projeto tem objetivo por mitigar.

O segundo, é o mapa de empatia relacionado à equipe de TI da empresa (segurança mais especificamente). É ela uma das mais importantes áreas, responsável por elaborar, acompanhar e conscientizar todas as outras áreas a respeito da necessidade da defesa cibernética dentro da organização.

Por fim, o último mapa de empatia diz respeito a ameaça que gostaríamos de prevenir. Sua forma é sistêmica pois ela usa o e-mail como veículo de propagação, mas sua natureza é de natureza humana por conter elementos que levam os colaboradores a cair nelas.

Mapa de empatia: Alta administração

Quem	Fazer	Vê	Diz	Faz	Ouve
Board Executivo/Diretoria	Transmitir segurança e seriedade nos negócios aos clientes; Certificar que dados e informações essenciais para o negócio da empresa estejam protegidas.	Oportunidade de aumentar a reputação da corporação e ganho de novos clientes com uma empresa mais protegida; Perda financeira e potencial perda de clientes por quebra da reputação causada por incidentes de intrusão.	Eu preciso que os colaboradores da empresa estejam muito bem preparados para possíveis ataques de <i>phishing</i> que venham a causar impactos; Eu quero que os clientes e a sociedade captem a empresa possui uma boa política de segurança.	Administração e gerenciamento geral da empresa; Planeja as metas estratégicas e cria metas para os departamentos;	Notícias na mídia sobre roubo de dados por e-mails enviados por criminosos; Amigos e conhecidos terem seus negócios arruinados por conta de invasões;
Pensa / Sente					
Dores			Ganhos		
Dados da organização sequestrados por criminosos a espera de altas quantias para o resgate; Dados e informações vazados para empresas concorrentes.			Aumentar a segurança da empresa; ter colaboradores mais preparados para lidar com e-mails externos ou suspeitos; aumentar a credibilidade da empresa de maneira geral.		

Mapa de empatia: Equipe/Área de Segurança da Informação					
Quem	Fazer	Vê	Diz	Faz	Ouve
Analistas técnicos e funcionais de Segurança da Informação	Gerenciar sistemas e tecnologias que ajudam a garantir a proteção dos ativos digitais da empresa; Monitorar a infraestrutura e rede da organização; Responder a incidentes de segurança; Elaborar novas formas de proteger a organização contra ataques cibernéticos.	Colaboradores sem uma preparação adequada para lidar com tentativas de <i>phishing</i> , inclusive no alto escalão da organização; Empresa em constante crescimento, dados importantes sendo adquiridos e cobiçados seja pela concorrência seja por criminosos.	Precisamos garantir a segurança dos ativos digitais da empresa; Ter uma política de <i>phishing</i> com uma base conceitual mais fundamentada, não dependendo apenas da experiência ou empirismo de colaboradores da equipe de segurança.	Monitoram a infraestrutura e rede da organização; Elaboram estratégias para proteger a organização contra ataques cibernéticos;	Corporações sofrem ataques diariamente; Grande parte dos ataques se iniciam através de técnicas de engenharia social; A alta administração preocupada com o preparo de seus colaboradores para lidar com ataques cibernéticos.
Pensa / Sente					
Dores			Ganhos		
A empresa ser vítima de ataques cibernéticos; ter sistemas comprometidos e dados vazados; não ter uma empresa comprometida ou preparada para lidar com a principal porta de entrada dos ataques, i.e., o <i>phishing</i> .			Empresa mais protegida; colaboradores de todos os departamentos colaborando para um ambiente mais seguro; tríade CIA sendo completamente entregue.		

Mapa de empatia: Ameaça					
Quem	Fazer	Vê	Diz	Faz	Ouve
Humana	Explorar vulnerabilidades em servidores e sistemas da organização; Proporcionar ganhos ilícitos para o praticante e perdas financeiras para a organização atacada.	Oportunidades em explorar a organização tendo como porta de entrada cada um de seus colaboradores; falha na avaliação de e-mails pelos colaboradores de todos os níveis hierárquicos da organização.	Eu quero explorar vulnerabilidades, principalmente as que envolvam engenharia social, muito mais eficazes contra pessoas; eu quero obter informações seja para vendê-las para a própria organização após o sequestro de dados ou sistemas ou para o concorrente.	Explora vulnerabilidades, também de caráter humano; aplica golpes em pessoas; coleta e sequestra dados fundamentais para a sobrevivência da organização.	Que a maioria das pessoas ainda estão despreparadas para lidar com ataques de engenharia social; muitas organizações não possuem políticas bem estabelecidas ou campanhas de <i>phishing</i> eficazes.
Pensa / Sente					
Dores			Ganhos		
Ser detectado ou o link com código malicioso não ser aberto pelo colaborador; ser preso por praticar crime.			Experiência ao atacar organizações; ganhos financeiros através da venda de informações e/ou sistemas.		

1.1.3 Benefícios e Justificativas

Esta seção do trabalho tem por objetivo a apresentação das justificativas e dos benefícios que motivam o desenvolvimento do projeto; nela apresentaremos os dados em forma de lista em duas seções que seguem respectivamente.

Como justificativa a realização deste projeto e solução do desafio/problema proposto por ele, destacamos os seguintes pontos:

- a) Aumento exponencial de crimes cibernéticos, principalmente após transformação digital ocorrida em tempo recorde após a pandemia da *COVID-19*.
- b) Preocupação da alta administração com o preparo dos colaboradores da organização para mantê-la segura e a preservação dos recursos de T.I.
- c) Organizações com pouco recurso financeiro para implementar campanhas de *phishing* e organizações com testes sem embasamento teórico.
- d) Organizações cada vez mais dependentes dos ativos digitais para a continuidade do negócio.
- e) Pessoas suscetíveis a ataques de engenharia social.
- f) Programas de *phishing* “para inglês ver”, ou seja, e-mails de teste pouco fidedignos e sem o devido acompanhamento para melhoria contínua dos colaboradores na detecção de ameaças.
- g) Programas de *phishing* sem a correta elaboração ou embasamento.

Como benefícios em decorrência da realização deste projeto e resução do desafio/problema proposto por ele, destacamos os seguintes pontos:

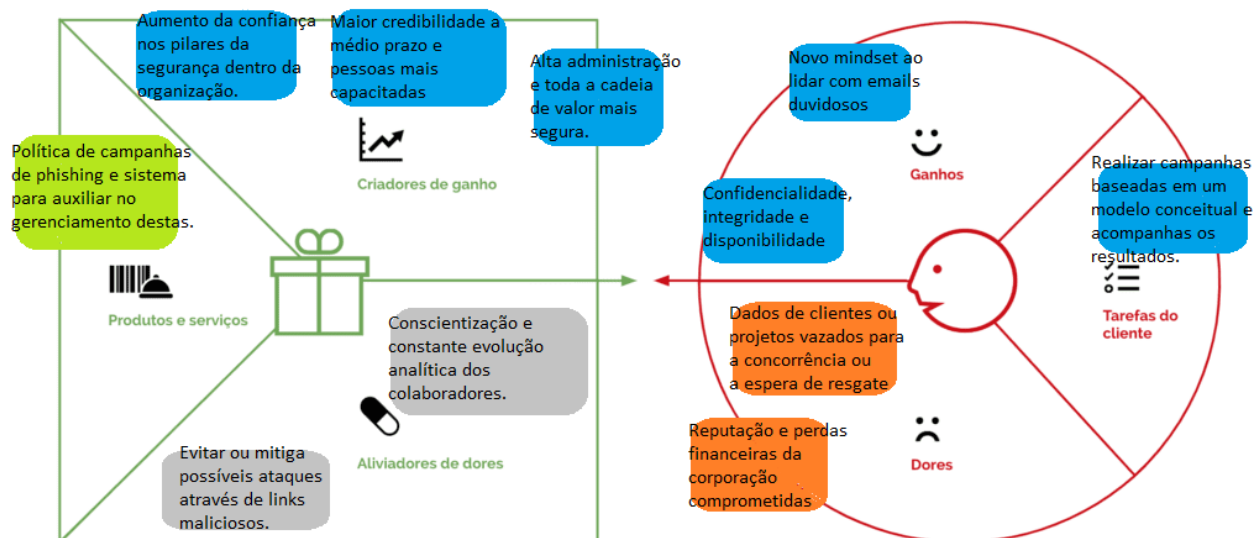
- a) Sistema e política gratuitos, com pouca necessidade de investimento em infraestrutura para a viabilização dos mesmos.
- b) Campanhas mais elaboradas, com base em teorias behavioristas, i.e., a mesma arma utilizada no ataque servirá para a defesa.
- c) Mitigação de riscos envolvendo ataques de engenharia social por *e-mail*.
- d) Credibilidade da organização tende a crescer e se consolidar.
- e) Colaboradores mais capacitados na detecção de ameaças.
- f) Alta administração percebe alto valor em uma proposta que traz ganhos qualitativos à organização sem necessariamente um alto custo envolvido na solução proposta.

Blueprint

Para proporcionar um melhor entendimento, a seguir apresentamos as interações existentes através do Blueprint que permite encontrar pontos de melhorias e oportunidades de inovação para a realização desse projeto. Posteriormente, segue o Canvas Proposta de valor que tem como objetivo auxiliar na criação e posicionamento dos serviços em torno do que a alta administração da Organização Tabajara deseja e precisa em relação a segurança da informação.

Blueprint	Identificar ataques por e-mail	Analisar os e-mails	Validação de tentativa	Feedback sobre o teste
Ações do colaborador	Identificar um e-mail suspeito, analisá-lo e reportar a equipe de segurança da informação a possível tentativa de ataque.			
Objetivos	Mitigar ameaças e ter colaboradores mais capacitados na identificação de ameaças.			
Atividades	Criar modelo conceitual e desenvolver sistema de campanhas.			
Questões	Qual modelo psicológico será adotado na elaboração da política? Qual linguagem de programação será utilizada para construir o sistema? Existirão integrações com outros softwares?			
Barreiras	Prazos para elaboração de todo o material.			
Saídas desejáveis	Garantia da segurança da organização.			
Funcionalidades	Execução de campanhas de <i>phishing</i> para toda a organização.			
Interação	Feedback para colaboradores sobre sucesso ou falha de testes e resultados para a alta administração acompanhar o andamento e evolução do nível de detecção.			
Mensagem	Mitigação de vulnerabilidades.			
Onde ocorre	Na estação de trabalho de todos os colaboradores através do cliente de e-mail.			
Tarefas Aparentes	Escolha de modelo conceitual adequado para execução de campanha.			
Tarefas Escondidas	Acompanhamento dos testes e constante evolução na modelagem de campanhas.			
Processos de suporte	Disposição da equipe de segurança da informação em realizar constante acompanhamento das campanhas.			

CANVAS de proposta de Valor



1.1.4 Hipóteses

A partir do conhecimento aprofundado do contexto do desafio e da definição das personas, nesta seção será mostrada uma tabela contendo as hipóteses levantadas para este projeto aplicado.

Matriz de observações para hipóteses

Observação	Hipóteses
Ameaças por e-mail através de engenharia social são portas de entrada perigosas para os sistemas e dados da organização.	Supõe-se que todo o gênero humano é suscetível a ameaças que venham com gatilhos e mecanismos psicológicos próprios da nossa espécie e de nossa evolução.
Alta administração está ciente da importância da segurança de seus ativos.	Supõe-se que a alta administração tem simpatia por essa nova proposta e ajudará às demais áreas a adotarem e seguirem as orientações do time de segurança no que diz respeito ao treinamento/novo paradigma de campanhas de <i>phishing</i> .
Equipe de segurança da informação possui metodologias e ferramentas de segurança, mas ainda não possui o apoio necessário para lidar com ataques aos colaboradores através de engenharia social.	Supõe-se que a equipe de segurança tem preparo e background suficiente para lidar com o novo sistema e política de campanhas de <i>phishing</i> .
Colaboradores são pessoas e assim sendo são suscetíveis a ataques.	Supõe-se que os colaboradores da organização não possuam treinamento adequado ou suficiente para conter todas ou a maioria das tentativas de ataque.

Diante das hipóteses expostas acima, realizou-se um *brainstorm* com o objetivo de priorizar as ideias em relação ao projeto proposto. Neste contexto, as principais ideias levantadas foram:

- 1- Levantar, analisar, compilar e propor um modelo conceitual com base no behaviorismo para o sistema de gerenciamento de campanhas proposto.
- 2- Analisar e empregar tecnologias de caráter *open source* para que a implantação seja possível e sem custos elevados nas organizações.
- 3- Aplicar uma metodologia com fortes bases para elaboração das campanhas.
- 4- Com o avanço das campanhas

Priorização de Ideias

Cenários	
C1	Complexidade na execução do projeto
C2	Urgência na execução do projeto
C3	Investimento necessário a execução do projeto
C4	Benefícios esperados ao final do projeto
C5	Nível de satisfação da alta administração

Escala	Benefícios	Abrangência	Satisfação	Investimento	Clientes (impacto)	Operacional (dificuldade)
5	Valor imediato para o modelo de negócio	Total	Total	Nenhum	Muito fácil	Muito fácil
4	Significativo para o modelo de negócio	Grande	Grande	Baixo	Fácil	Fácil
3	Razoável para o modelo de negócio	Razoável	Razoável	Médio	Médio	Médio
2	Pouco para o modelo de negócio	Pequena	Pequena	Alto	Grande	Grande
1	Baixo para o modelo de negócio	Baixa	Baixa	Elevado	Elevado	Elevado

Ideias	Comparação de Cenários					
	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5	Total
1	4	5	3	5	4	21
2	4	3	3	4	3	17
3	5	4	2	3	4	18
4	5	3	4	5	5	22

1.2 Solução

Esta seção tem o objetivo de apresentar, de maneira bem estruturada, os objetivos do projeto, definindo expectativas claras e objetivas, para maximizar as chances de alcançar os resultados esperados. De modo geral, a proposta de solução para o projeto se divide nas categorias teórica e prática. A primeira referindo-se a análise das correntes comportamentais para que, após entendimento da linha e mecanismos a serem adotados, a segunda parte, a prática possa ser iniciada. Nela, bases de dados e sistema para gerenciamento de campanhas serão desenvolvidos usando os insumos da parte inicial. Para melhor compreensão, o artefato de objetivo SMART será apresentado a seguir.

1.2.1 Objetivo SMART

Esta seção tem o objetivo de apresentar, de maneira bem estruturada, os objetivos do projeto, definindo expectativas claras e objetivas, para maximizar as chances de alcançar os resultados esperados.

S (Specific - Específico)	Tornar a organização mais segura e mitigar ameaças por meio de correio eletrônico.
M (Mensurable - Mensurável)	Satisfação dos diversos departamentos e controle centralizado do sucesso ou não dos colaboradores nos testes de simulação de intrusão por e-mails maliciosos.
A (Attainable - Antecipável)	Realização de disparos de campanhas de <i>phishing</i> .
R (Relevant - Relevante)	Proteção dos ativos de T.I. e aumento da crença de garantia de continuidade dos negócios da organização.
T (Time Based - Temporal)	Aumento da eficiência na detecção de ameaças cibernéticas pelos colaboradores da organização e, por consequência, aumento da reputação da mesma perante a sociedade.

1.2.2 Premissas e Restrições

Esta seção tem o objetivo de apresentar as condições necessárias para que o projeto seja desenvolvido de maneira eficiente. Assim sendo, a matriz de riscos será apresentada a seguir.

O projeto apresenta as seguintes premissas:

- a) A maior parte das tentativas de *phishing* deve ser reconhecida pelos colaboradores após certo período de campanhas.
- b) O sistema, base de conhecimento e estratégias devem ser atualizadas com muita frequência para que os testes não se tornem “viciados” ou facilmente detectados. A verossimilhança com ataques de criminosos verdadeiros deve ser almejada sempre.
- c) O resultado deve ser satisfatório.

O projeto apresenta as seguintes restrições:

- a) Deve utilizar uma aproximação teórica confiável para concepção e desenvolvimento do projeto.
- b) Deve ser *open source* e bem documentado para utilização em qualquer organização que quiser adotá-lo.
- c) Deve ser realizado com o menor custo financeiro possível.

Matriz de Riscos

De acordo com as premissas e restrições do projeto, os riscos foram identificados e correlacionados entre impacto e probabilidade. O resultado pode ser encontrado logo abaixo em forma tabular.

Risco	Probabilidade	Impacto	Ação
Falso/Positivo durante a fase inicial de implementação do sistema.	Alto	Médio	Análise cuidadosa e detalhada da equipe de segurança da informação durante as etapas iniciais de desenvolvimento e implantação.
Invasão por criminoso por <i>phishing</i> antes que o projeto e seus efeitos desejados sejam alcançados.	Médio	Alto	Constante alerta, como é feito atualmente na organização, para mitigação, proteção e resposta a incidentes até que os frutos do projeto sejam alcançados.
Falha no design ou arquitetura baseada no modelo conceitual comportamental.	Baixo	Médio	Estudo detalhado durante a primeira sprint, a fase mais conceitual do trabalho para evitar a propagação de erros e falhas para as fases posteriores.

1.2.3 Backlog de Produto

Esta seção tem o objetivo de apresentar, de maneira bem detalhada, o backlog de requisitos idealizados para o desenvolvimento da solução. Aqui está sendo considerado o total de três sprints para a realização das atividades.

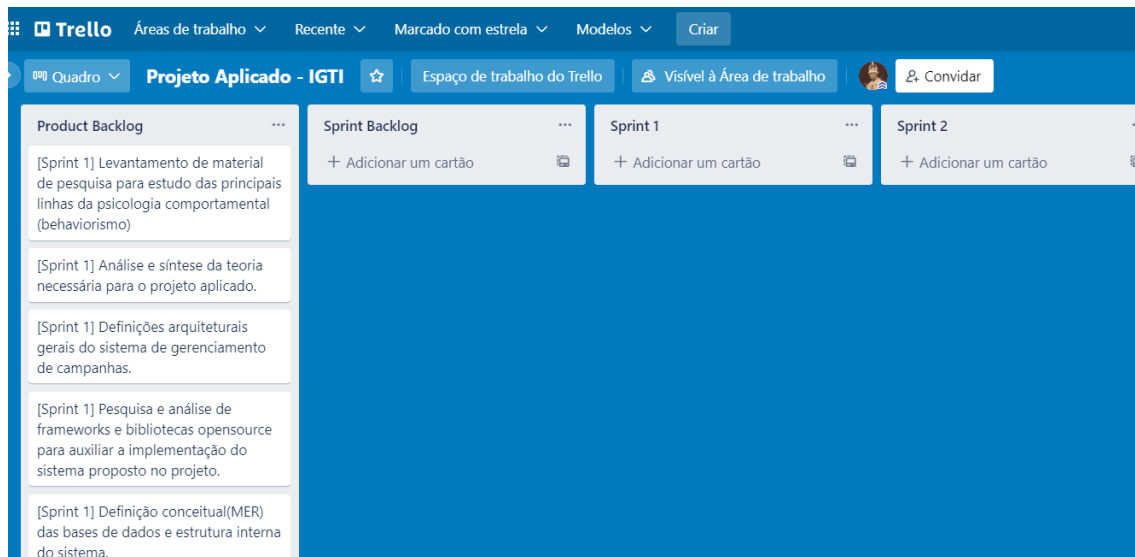
A fase inicial deste projeto tem como objetivo os ajustes necessários na primeira etapa após avaliação do orientador, como análise de contexto, matriz CSD, personas, apresentação da solução, benefícios e justificativas, hipóteses, premissas e restrições, Canvas de proposta e valor e todos os artefatos necessários para a entrega deste projeto.

A primeira sprint será a parte fundacional deste projeto. O estudo, análise e elaboração de um modelo baseado na psicologia comportamental para a elaboração do sistema de gerenciamento de campanhas de *phishing*. Pretende-se, além da elaboração deste modelo, a definição de bases de dados(modelo de entidade e relacionamento) necessárias para a construção do software, além do estudo de viabilidade para utilização de outros projetos *open source* para a construção do sistema.

A segunda sprint terá como objetivo uma parte mais técnica, do início da construção do sistema propriamente dito. Por meio de uma máquina virtual pretende-se criar as bases de dados no SGBD escolhido na primeira parte além do desenvolvimento do sistema com uma linguagem de programação que também será definida na primeira sprint.

Na terceira e última sprint será desenvolvido o restante do sistema além da parte final deste projeto, as considerações finais que consiste nos seguintes itens: resultados, contribuições e próximos passos.

Trello



The screenshot shows a Trello workspace for 'Projeto Aplicado - IGTI'. The board is organized into four columns: Product Backlog, Sprint Backlog, Sprint 1, and Sprint 2. The Product Backlog column contains five tasks for [Sprint 1]. The Sprint Backlog, Sprint 1, and Sprint 2 columns are currently empty, each with a button to 'Adicionar um cartão' (Add a card).

Column	Items
Product Backlog	<ul style="list-style-type: none"> [Sprint 1] Levantamento de material de pesquisa para estudo das principais linhas da psicologia comportamental (behaviorismo) [Sprint 1] Análise e síntese da teoria necessária para o projeto aplicado. [Sprint 1] Definições arquiteturais gerais do sistema de gerenciamento de campanhas. [Sprint 1] Pesquisa e análise de frameworks e bibliotecas opensource para auxiliar a implementação do sistema proposto no projeto. [Sprint 1] Definição conceitual(MER) das bases de dados e estrutura interna do sistema.
Sprint Backlog	+ Adicionar um cartão
Sprint 1	+ Adicionar um cartão
Sprint 2	+ Adicionar um cartão

2. Área de Experimentação

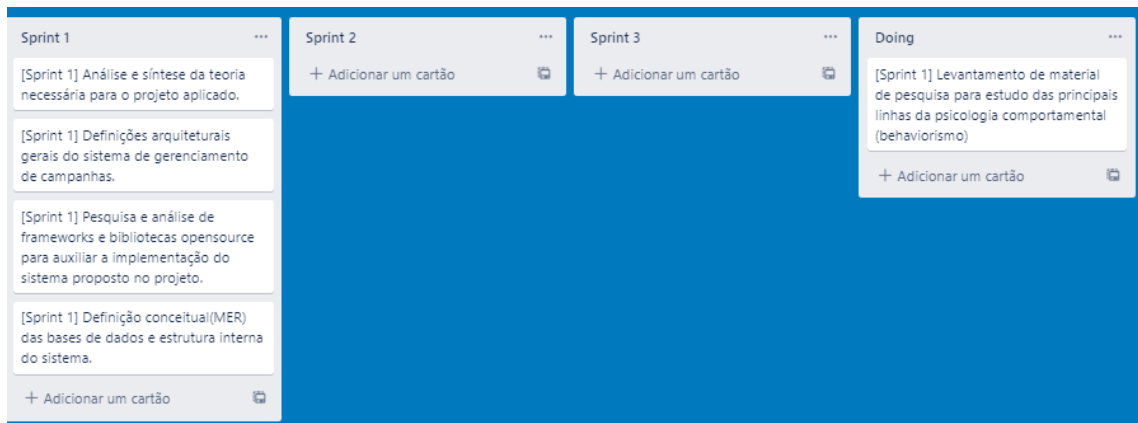
Esta seção tem o objetivo de apresentar as evidências do planejamento dos requisitos selecionados do Backlog de Produto, além de mostrar a maneira como eles foram desenvolvidos e registrar os resultados alcançados.

2.1 Sprint 1

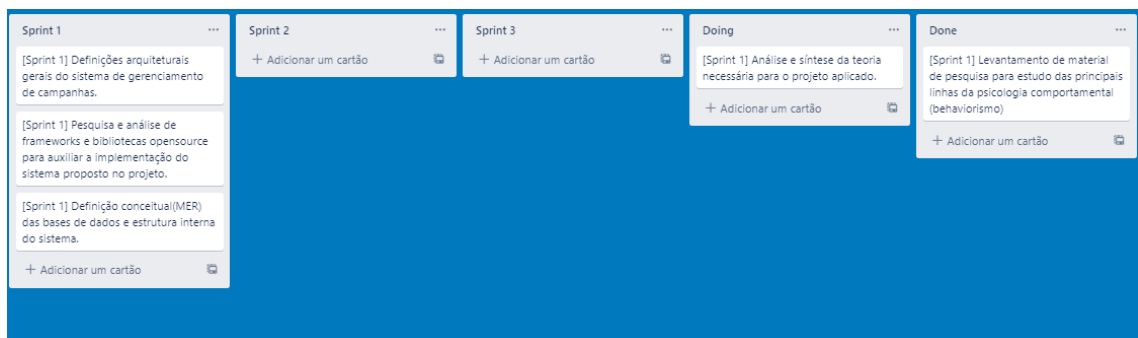
2.1.1 Solução

- Evidência do planejamento:

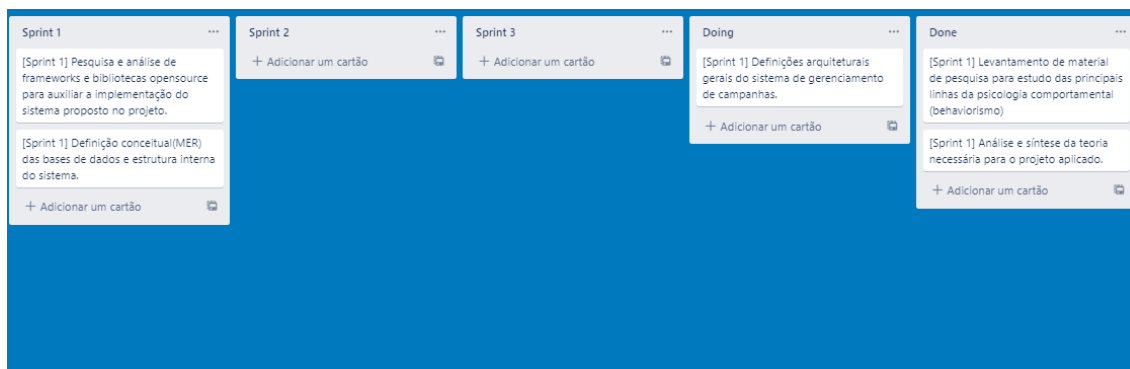
Para o item 1 da primeira sprint, a saber, “Levantamento de material de pesquisa para estudo das principais linhas da psicologia comportamental(behaviorismo), o planejamento via *Trello* encontra-se abaixo:



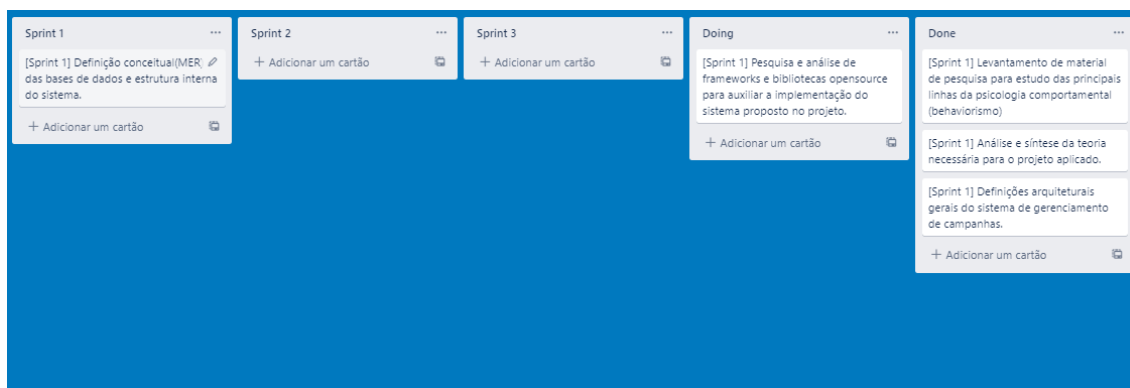
Para o item 2 da primeira sprint, a saber, “Análise e síntese da teoria necessária para o projeto aplicado”, o planejamento via *Trello* encontra-se abaixo:



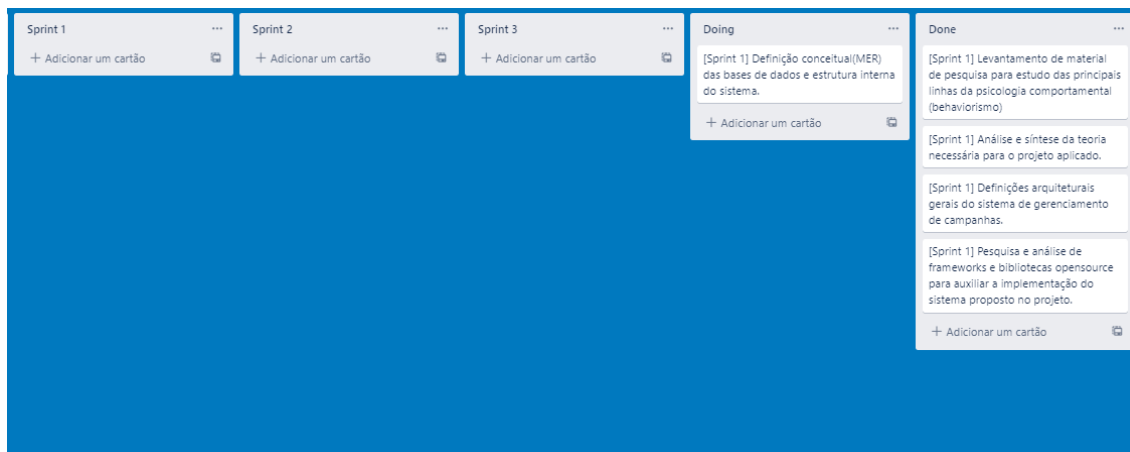
Para o item 3 da primeira sprint, a saber, “Definições arquiteturais gerais do sistema de gerenciamento de campanhas”, o planejamento via *Trello* encontra-se abaixo:



Para o item 4 da primeira sprint, a saber, “Pesquisa e análise de frameworks e bibliotecas *opensource* para auxiliar a implementação do sistema proposto no projeto”, o planejamento via *Trello* encontra-se abaixo:



Para o item 5 da primeira sprint, a saber, “Pesquisa e análise de frameworks e bibliotecas *opensource* para auxiliar a implementação do sistema proposto no projeto”, o planejamento via *Trello* encontra-se abaixo:



Para que os itens possam ser facilmente correlatos nas seções posteriores, eles serão identificados de acordo com a tabela que segue abaixo:

Código	Descrição
A (Item 1)	Levantamento de material de pesquisa para estudo das principais linhas da psicologia comportamental (behaviorismo).
B (Item 2)	Análise e síntese da teoria necessária para o projeto aplicado.
C (Item 3)	Definições arquiteturais gerais do sistema de gerenciamento de campanhas.
D (Item 4)	Pesquisa e análise de frameworks e bibliotecas open source para auxiliar a implementação do sistema proposto no projeto.
E (Item 5)	Definição conceitual (MER) das bases de dados e estrutura interna do sistema.

- Evidência da execução de cada requisito:

A (Item 1)

Para o item inicial desta lista, aquele que será a base fundadora e funcional para o sistema de gerenciamento de *phishing* com base na psicologia behaviorista, adotou-se a metodologia de revisão bibliográfica para aquisição de conhecimento mais profundo sobre a psicologia comportamental.

Desse modo, para dar estrutura a esse estudo, adotou-se a seguinte estratégia para obter o conhecimento fundamental necessário para o projeto: levantamento e aquisição de bibliografias sobre história da psicologia e sobre história da psicologia comportamental, levantamento e aquisição de obras mais especializadas sobre os dois maiores expoentes dessa linha de estudo, Watson e Skinner, além da leitura de outros artigos encontrados na internet.

Em suma, as obras utilizadas para estudo do tema proposto serão listadas abaixo utilizando a ordenação alfabética autoral:

SAUL, Cristoff. Análise Comportamental: entenda como a ciência do comportamento pode impactar as pessoas e suas relações. São Paulo: Literare Books, 2021.

SKINNER, B.F. Sobre o Behaviorismo. São Paulo: Editora Cultrix, 2011.

WATSON, John B. O behaviorismo clássico. São Paulo: Hogrefe, 2021.

Além desse levantamento inicial, durante a aula inaugural da disciplina, tivemos a oportunidade de apresentar as ideias para o professor orientador e receber feedback instantâneo para que o projeto pudesse ser realizado da melhor forma possível. Assim sendo, após a apresentação da ideia deste projeto, o professor Maximiliano citou um outro grande psicólogo que ajudaria na pesquisa teórica sobre o tema, Abraham Maslow, que criou uma teoria da necessidade humana que vai em encontro às teorias behavioristas mais consolidadas na academia. Desse modo, utilizamos também o conhecimento proposto por Maslow para ajudar a esclarecer e entender melhor o comportamento humano com o objetivo de propor um melhor sistema de gerenciamento de phishing behaviorista. As obras estudadas para entender um pouco melhor a teoria de Maslow estão listadas abaixo:

MASLOW, Abraham H. A theory of Human Motivation. Mansfield Centre: Martino Publishing, 2013.

MASLOW, Abraham, H. Toward a Psychology of Being. Mansfield Centre: Martino Publishing, 2011.

B (Item 2)

O objetivo deste item, “Análise e síntese da teoria necessária para o projeto aplicado” é o de consolidar a base teórica fundamental para formular a política de *phishing* e também para arquitetar a estrutura interna do sistema de gerenciamento proposto. Este item, junto com o primeiro (A - Item 1: levantamento de materiais de pesquisa), terão seus resultados apresentados de forma detalhada na próxima seção (Evidência dos resultados).

C (Item 3)

O objetivo deste item é o de discutir as definições arquiteturais do sistema de gerenciamento de campanhas de *phishing*. Após análise e consideração de diversos padrões arquiteturais, decidiu-se partir para uma estrutura de sistema distribuído, usando o paradigma de microsserviços e microfrontends para que o sistema possa ser desenvolvido com a maior flexibilidade possível, tanto de tecnologia quanto cronologicamente. Ao construir um microsserviço responsável por atender as requisições de um cliente, i.e., separando as responsabilidades do sistema em blocos bem definidos, não é necessário adotar uma linguagem de programação específica ou framework para desenvolver cada uma das partes.

Evidentemente, para que o sistema deste projeto possa ser construído, precisamos de uma *stack* tecnológica. Adotaram-se as seguintes tecnologias que estão listadas em forma tabular abaixo:

Python	Javascript
AWS - EC2 instances	AWS RDS - MySql
Angular 12	AIO HTTP

Além disso, discriminamos os componentes que o formará o sistema. Cada um deles será descrito e o detalhamento técnico será mostrado em forma de diagramas UML na seção de “Evidência e resultados”.

Microsserviço: bff-phishing-campaing

Este componente é responsável por ser a API que servirá as funções de gerenciamento para o frontend e para a ferramenta de interface de linha de comando a ser utilizada pelos analistas e gerentes da informação da companhia. Nele estarão expostos os recursos necessários para guardar templates de phishing, editá-los, excluí-los, bem como realizar o disparo de campanhas.

Database RDS - MySQL

Este componente é o banco de dados relacional escolhido para guardar as informações do sistema, como os *templates* criados para as campanhas, suas categorias e subcategorias e todas as informações de apoio para que o sistema funcione e opere corretamente. Foi escolhido um banco de dados como serviço do provedor *Amazon Web Services* por conta de seu baixo custo, segurança e por ser um *PaaS*, liberando desenvolvedores e empresa de manutenção do banco, como atualização de versão, instalação de *patches* de segurança e toda a infraestrutura necessária.

Frontend: frontend-phishing-campaign

Este é o componente responsável pela apresentação GUI do sistema para seus utilizadores. Como este tipo de componente é sempre mais complexo, pois exige a criação de telas e lógica para cada tipo de funcionalidade, além deste, propõe-se a criação de uma interface mais simples, descrita logo a seguir, para que o valor seja entregue mais rapidamente aos usuários.

CLI: core-command-line-interface

Este é o componente responsável pela interação com o sistema de gerenciamento por meio de comandos em linha de terminal. Como este tipo de componente tende a ser mais simples do ponto de vista de desenvolvimento, espera-se entregá-lo primeiro para usuários possam utilizar o sistema mais rapidamente.

External System: Gophish

Este é o componente responsável por auxiliar o sistema proposto. Será descrito com um maior detalhamento no próximo item desta lista (D (Item 4)).

D (Item 4)

O penúltimo item, denominado “Pesquisa e análise de frameworks e bibliotecas open-source para auxiliar a implementação do sistema proposto no projeto”, foi realizado através de pesquisas em ferramentas de busca online e também recebeu orientação do professor Maximiliano. Após analisar alguns programas e frameworks

disponibilizados em forma de *software* livre, decidiu-se por utilizar o framework chamado *GoPhish*. Este é um programa bastante completo, que nos permite envio de e-mails para uma determinada lista de destinatários, acompanhar o progresso e sucesso de cliques além de ter um grande diferencial, uma API de fácil uso escrita na linguagem de programação Python. Assim sendo, na seção de evidências, será evidenciado o esquema de uso desta ferramenta, bem como na seção arquitetural mostra-se a integração dela no sistema.

E (Item 5)

Por fim, o último item planejado para a sprint, “Definição conceitual (MER) das bases de dados e estrutura interna do sistema”. Com base na teoria vista e na proposta de divisão por categorias que será evidenciada nos **itens 1 e 2(A e B)**, o modelo de entidade relacional para o sistema é planejado e criado. A evidência do mesmo será apresentada na sua respectiva área na seção de **Evidência dos Resultados**.

- **Evidência dos resultados:**

Como resultado desta primeira iteração do projeto, os resultados serão evidenciados da seguinte maneira nesta seção: Os itens **A** e **B** terão como resultado a metodologia consolidada em forma de texto, o item **C** será evidenciado através de dois diagramas, o de componentes e o de implantação, o item **D** xpto, e o item **E** terá como output final o diagrama de entidade relacional que será utilizado na próxima iteração (sprint 2) para o desenvolvimento do sistema em si.

Itens 1 e 2(A e B)

Conforme explicado no parágrafo introdutório, apresentar-se-á em uma única seção os resultados obtidos na realização dos itens A e B da sprint 1. Para que o texto fique organizado, este será dividido em três pontos, o primeiro será uma breve apresentação da teoria por trás do projeto, o segundo será a apresentação de uma política estruturada com base na teoria apresentada anteriormente e, por fim, a teoria aplicada a proposta do sistema de gerenciamento de phishing.

Behaviorismo: de Watson à Skinner

A teoria adotada neste projeto nasceu no século XIX, período em que a psicologia se consolidava como ciência e muitos de seus representantes procuravam retirar as amarras metafísicas e adotavam com cada vez mais vigor o método científico já consolidado em outras ciências da natureza.

Dito isso, o behaviorismo, anglicismo para a palavra original na língua inglesa *behaviorism*, i.e., comportamentalismo, é a linha psicológica que tem como objetivo o estudo do comportamento. Diferente de outras doutrinas e linhas de pesquisa com bastante carga subjetiva, a linha de estudos iniciada por John B. Watson acredita que

a psicologia humana pode ser estudada objetivamente por meio da observação das ações dos espécimes humanos, observando o comportamento.

John Broadus Watson, iniciador deste movimento em 1913 com o seu artigo *Psychology as the Behaviorist Views it*⁵, utiliza as teorias de Vladimir Mikhailovich Bechterev e Ivan Petrovich Pavlov (estudos sobre o mecanismo de condicionamento animal) para propor a universalização desta teoria ao gênero humano, criando assim seu principal conceito, o de condicionamento reflexo. Watson explica que o mecanismo de estímulo e resposta, como mostrar um fragmento de carne a um cachorro(estímulo) e observar a salivção(resposta), poderia sofrer engenharia ao acrescentar um novo estímulo neutro associado ao estímulo original. Dessa forma, ao mostrar uma pequena porção de carne a um cachorro e ao mesmo tempo acrescentar um novo estímulo, como o tocar de uma campainha, o estímulo original expande seus efeitos ao novo estímulo condicionado. Dessa forma, após algum tempo de treino e repetição, o estímulo da campainha por si só traria o efeito da salivção ao cão sem que o cheiro ou a visão da fração de carne fossem apresentados. Evidentemente, os humanos também estão sujeitos a esse tipo de mecanismo e alteração/expansão de estímulos para se chegar num mesmo efeito. Essa teoria, hoje chamada de behaviorismo clássico, tem algumas limitações como considerar fundamentalmente estímulos organolépticos e ainda ter uma base metafísica, como considerar que os seres vivos nascem com determinados reflexos, i.e., uma espécie de inatismo. No entanto, em 1945, um outro pensador deu mais base a essa teoria e é a que utilizaremos como base no projeto.

No primeiro dos três grandes artigos escritos por Burrhus Frederic Skinner⁶, intitulado *The Operational Analysis of Psychological Terms*, o autor inicia uma nova interpretação do behaviorismo. Na teoria tradicional, o estudo do comportamento dos seres vivos era resumido ao comportamento e reflexo, como vimos anteriormente. O mecanismo clássico pode ser traduzido nas seguintes fórmulas:

I - Estrutura pré-existente na psique animal e humana

Estímulo incondicionado => resposta incondicionada

II - Adição do elemento neutro em conjunto com o estímulo incondicionado

Estímulo incondicionado + estímulo neutro => resposta incondicionada

III - Transformação do estímulo neutro em estímulo condicionado substituindo o incondicionado

Estímulo condicionado => resposta incondicionada

⁵ Artigo publicado na revista *Psychological Review*. Pode ser lido atrav[es deste link? <https://www.ufrgs.br/psicoeduc/chasqueweb/edu01011/behaviorist-watson.pdf>

⁶ Os artigos que referidos são: *The operational analysis of psychological terms*(1945), *Behaviorism at fifty*(1963) e *About behaviorism*(1974).

Na nova interpretação, Skinner nos traz o conceito de condicionamento operante que traz, além dos elementos “inatos” do behaviorismo clássico, os pensamentos e emoções. Os comportamentos que podemos observar não são somente estimulados por forças biológicas/genéticas. O comportamento em si possui um mecanismo de reforço para que possa se repetir e isso acontece devido às consequências deste.

Todo indivíduo busca sobreviver, se autorrealizar e outras ações que cada qual sente necessidade. Então, à medida que determinado comportamento ajude o indivíduo a suprir sua necessidade, este entrará numa espécie de coleção comportamental de onde será consultado e reutilizado na mesma ou numa necessidade semelhante. No vocabulário de Skinner, o mecanismo de repetição é o operante. Além disso, diferente do caráter determinístico defendido por Watson, os comportamentos para Skinner seguem um padrão probabilístico. A depender do reforço positivo (sucesso no alcance do objetivo) ou negativo (punição por determinado comportamento), a probabilidade de o comportamento voltar a acontecer é alterada; se seguir pelo reforço positivo a probabilidade aumenta, caso siga pelo reforço negativo, diminui. Isso não significa que o comportamento será recorrente ou extinto necessariamente.

Para formalizar o mecanismo skinneriano do behaviorismo radical, apresenta-se a seguinte fórmula:

$$(I) S(d/\delta) - (II) R \rightarrow (III) C$$

(I) S: Estímulo. Pode ser um estímulo do tipo **d** (discriminativo) ou do tipo **δ** (delta). O primeiro se refere a um estímulo, que na sua presença aumenta a probabilidade de o comportamento ocorrer. O segundo se refere a um estímulo que na sua presença infere que o estímulo consequente reforçador não estará presente.

(II) Resposta ou forma. É a reação a determinado estímulo. Na passagem do estímulo para a resposta podemos observar que se utiliza o traço (-) ao invés da flecha (->), justamente para indicar que esta transição é probabilística, não determinística.

(III) Consequência que pode vir na forma de reforço positivo ou negativo.

Por fim, para finalizar a contribuição de Skinner na parte conceitual deste projeto, uma outra categorização importante para o sistema será apresentada, as classes de estímulos. Anteriormente, na primeira fase do behaviorismo, considerava-se apenas uma classe, a proximidade física, também chamada de generalização (como a porção de carne sendo aproximada do cão para causar a salivação). Mas, após o avanço desta ciência, construiu-se uma nova categoria, conhecida como funcional.

Esta segunda categoria é um pouco mais complexa e abarca todos os níveis da vida. Ela é dividida em três itens. O primeiro deles é a filogênese, que abarca as características genéticas que são transmitidas de geração em geração, os chamados comportamentos padrões ou genéticos, como o medo “inato” herdado através das mais

diferentes fobias. Entre outros, alguns destes são exemplos dessa subcategoria: início da vida, aptidão e sucesso reprodutivo, reflexos condicionados inatos, aptidão. O segundo item é a ontogênese, que são aprendizados individuais do organismo com seu meio. É a seleção comportamental por consequência, é o condicionamento operante skinneriano por definição; um comportamento que teve sucesso tende a ser selecionado ou reforçado e tem maior probabilidade de ocorrer numa circunstância similar, i.e., a lei do feito. O último item é a cultura, presente apenas no gênero humano. Esta subcategoria se traduz naquilo que traz benefícios para o grupo e contribui para solução de problemas coletivos, o mais profundo e complexo item da categoria funcional.

Por fim, para citar Abraham Harold Maslow, autor citado pelo orientador deste projeto, utilizamos sua famosa hierarquia de necessidades para ajudar no entendimento dos estímulos que poderiam gerar determinados comportamentos nos seres humanos. Este conhecimento está traduzido de forma didática na conhecida pirâmide de Maslow, que busca dar forma a esta hierarquia de necessidades humanas. Com cinco níveis, esta figura geométrica busca dar profundidade e hierarquizar as necessidades. Para que uma necessidade seja despertada, a necessidade anterior necessita ser satisfeita. Iniciando da base da pirâmide, temos as necessidades fisiológicas, que são constituídas da respiração, comida, água, sexo, sono e excreção. O próximo nível é o da segurança, composta por segurança corporal, do ofício, dos recursos, da moralidade, família, saúde e propriedade. Logo acima, encontramos o nível do relacionamento, que contém a amizade, família e intimidade sexual. O penúltimo nível da pirâmide guarda a estima, que é formada pela autoestima, confiança, conquista, respeito aos outros e dos outros. Por fim, no topo da pirâmide está a realização pessoal, onde podemos encontrar a moralidade, criatividade, espontaneidade, solução de problemas, ausência de preconceitos e aceitação dos fatos.

Para realizar uma síntese de Maslow com Skinner, observamos que a hierarquia de necessidades do primeiro se relacionam diretamente com a categoria de comportamentos funcionais do segundo. A subcategoria filogenética está para a base da pirâmide como a última categoria cultural está para o topo do poliedro. As três categorias no meio da figura geométrica se relacionam com a categoria ontogenética. Desse modo, a estruturação da política e, por consequência o modelo de entidade relacional podem ser melhor estruturados para guardar as informações necessárias para a proposta deste projeto.

Política de campanhas baseada no comportamentalismo

A política de campanhas pensada para os analistas de segurança e funcionais que elaboração os testes e educação dos colaboradores da organização é, em seu primeiro modelo, simples, mas dá estrutura ao caos que encontramos nas organizações contemporâneas que muitas vezes realizam testes ineficazes e não promovem a

conscientização necessária de seus funcionários, objetivo final das campanhas para que dados e sistemas não sejam comprometidos por meio de engenharia social.

De início, propõe-se a criação de um calendário segmentado para a realização das diversas campanhas que devem ocorrer com regularidade ao longo do ano fiscal. O tempo é sem dúvida algo intrínseco à espécie humana; desde a regulação do funcionamento interna do corpo até as convenções sociais e culturais como horário do almoço ou jantar, chá ou período desprendido no trabalho. Utilizar-se deste recurso, da regularidade, traz ordem ao caos algumas vezes chamado de aleatoriedade ou testes surpresa. A fim de dar essa estrutura, propõe-se a divisão por *quarter* do ano fiscal ou qualquer outra divisão que segmente o ano em 4 partes. Cada uma destas partes será composta por um nível de regularidade de campanhas. Por exemplo, o primeiro trimestre terá nível 1 de campanhas, i.e., *phishings* serão enviados a cada duas semanas. O segundo trimestre terá nível 2, i.e., *phishings* serão enviados a cada 1,5 semanas inclusive aos finais de semana. E assim por diante, até o último semestre, o mais severo de todos de acordo com esta lógica. Evidentemente, outras variáveis devem ser acrescentadas nesse sistema de divisão do tempo, como o tipo de categoria que será explorado para verificar a recorrência de pessoas que clicam nos links dos testes.

A seguir, propõe-se a estruturação dos *phishings* por categorias. No entanto, antes de seguir com a proposta, faz-se necessário realizar a analogia ou junção da teoria vista no tópico anterior com a proposta desta política. O mecanismo de phishing é bem semelhante à estrutura estudada pelos psicólogos comportamentais. Assim como elucidamos a fórmula para formalização de Skinner,

$$S(d/\delta) - R \rightarrow C$$

podemos facilmente substituir as variáveis por

E-mail malicioso - clique em link -> dano

afinal, os criminosos responsáveis por este tipo de ataque não estão apenas técnicos altamente especializados na engenharia de malwares, vírus ou outros artefatos capazes de causar danos à organização ou seus membros, estas pessoas conhecem a natureza humana, ao menos num nível básico, e utilizam esse tipo de vulnerabilidade para obter acesso a dados confidenciais e prejudicar seus alvos. Dito isso, prosseguindo com a proposta da divisão dos testes por categorias, Skinner e Maslow podem ajudar os analistas responsáveis pela elaboração dos testes no seguinte sentido. Para que os funcionários da organização possam ser conscientizados de maneira mais completa, a maior gama possível de simulações deve ser aplicada. Assim sendo, a categorização proposta é a seguinte:

Categoria 1: Classe funcional filogenética

De acordo com o que foi visto na teoria behaviorista e na teoria da hierarquia das necessidades humanas, estas duas contribuições se relacionam em seu primeiro nível. A filogênese, i.e., as características genéticas que são transmitidas de geração em geração, gerando comportamentos padrões ou reflexos incondicionados, abarca a base da pirâmide de Maslow. Desse modo, a primeira categoria seria composta de textos de e-mails sobre os temas:

- Respiração, comida, água, sexo, sono e funções básicas do organismo

Categoria 2: Classe funcional ontogenética

Na segunda categoria, a ontogenética, i.e., as aprendizagens individuais do organismo com seu meio, gerando a seleção por consequência, podemos encaixar o meio da pirâmide de Maslow, i.e., os níveis de segurança, amor/relacionamento e estima. Assim sendo, os itens de e-mail a serem elaborados nesta categoria são os seguintes:

- Segurança: corporal, emprego, recursos, moralidade, família, saúde e propriedade.
- Amor: amizade, família, intimidade
- Estima: autoconfiança, conquista, respeito ao e dos outros

Categoria 3: Classe funcional cultural

Por fim, a última categoria funcional se refere a cultura. A mais complexa das categorias se relaciona ao topo da pirâmide de Maslow, os bens mais intangíveis que a humanidade almeja. Assim sendo, os itens a serem elaborados nesta categoria devem seguir os seguintes parâmetros:

- Realização pessoal: moralidade, criatividade, espontaneidade, solução de problemas, aceitação dos fatos.

Concluindo, propõe-se uma estrutura para que os analistas de segurança técnicos e funcionais possam elaborar e catalogar os textos de e-mails que serão lançados via campanha ao longo da divisão temporal proposta. Evidentemente, são grandes temas e cada organização deve elaborar suas campanhas de acordo com a realidade de sua empresa, de seus funcionários e de acordo com o nível de granularidade que se deseja conscientizar os colaboradores da organização.

Item 3 - C

Conforme descrito na seção de “Evidência da execução de cada requisito”, este item é responsável por evidenciar o planejamento arquitetural através dos diagramas UML de componentes e implantação. Ambos serão representados abaixo.

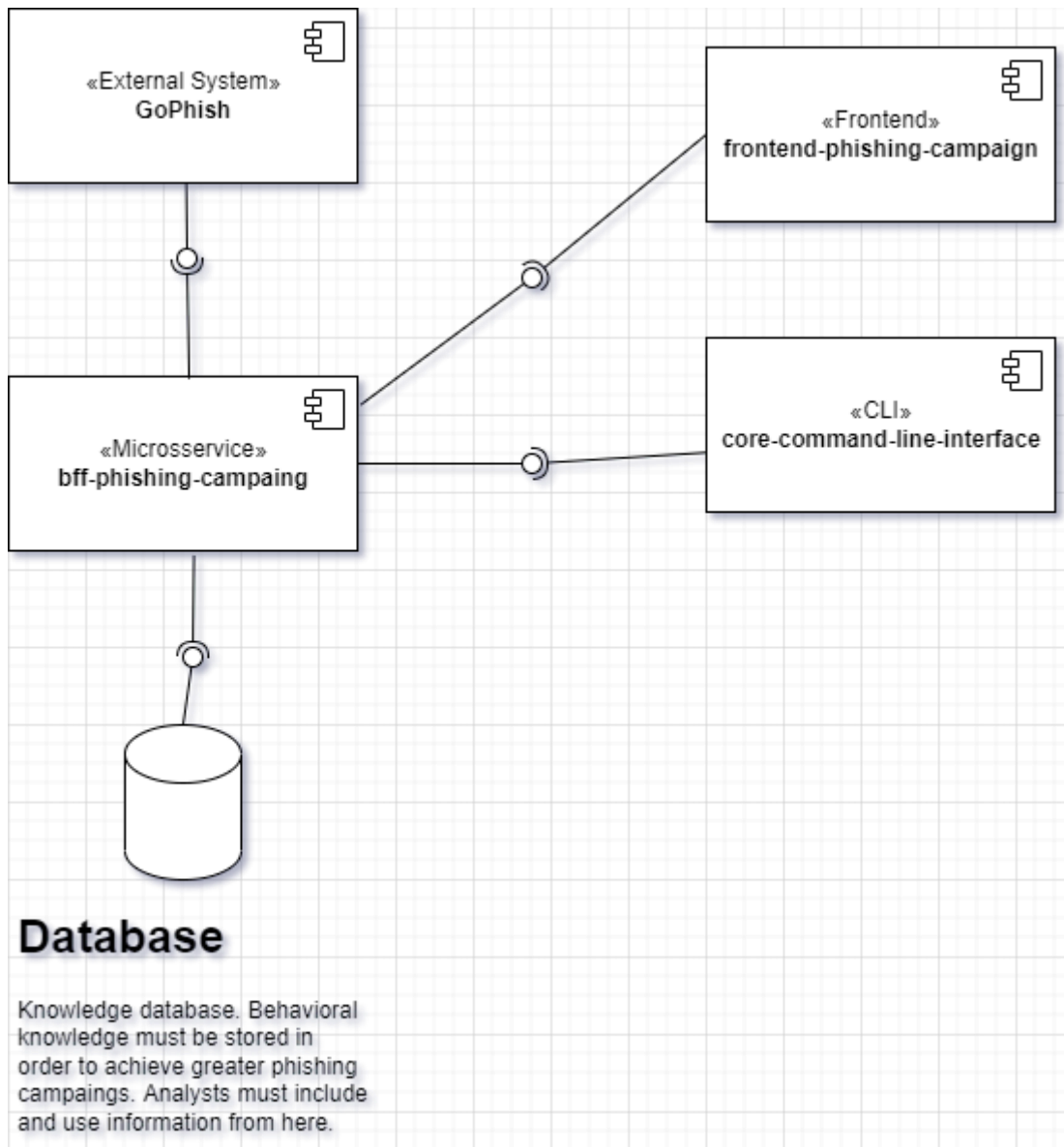


Figura 1 - Diagrama de Componentes

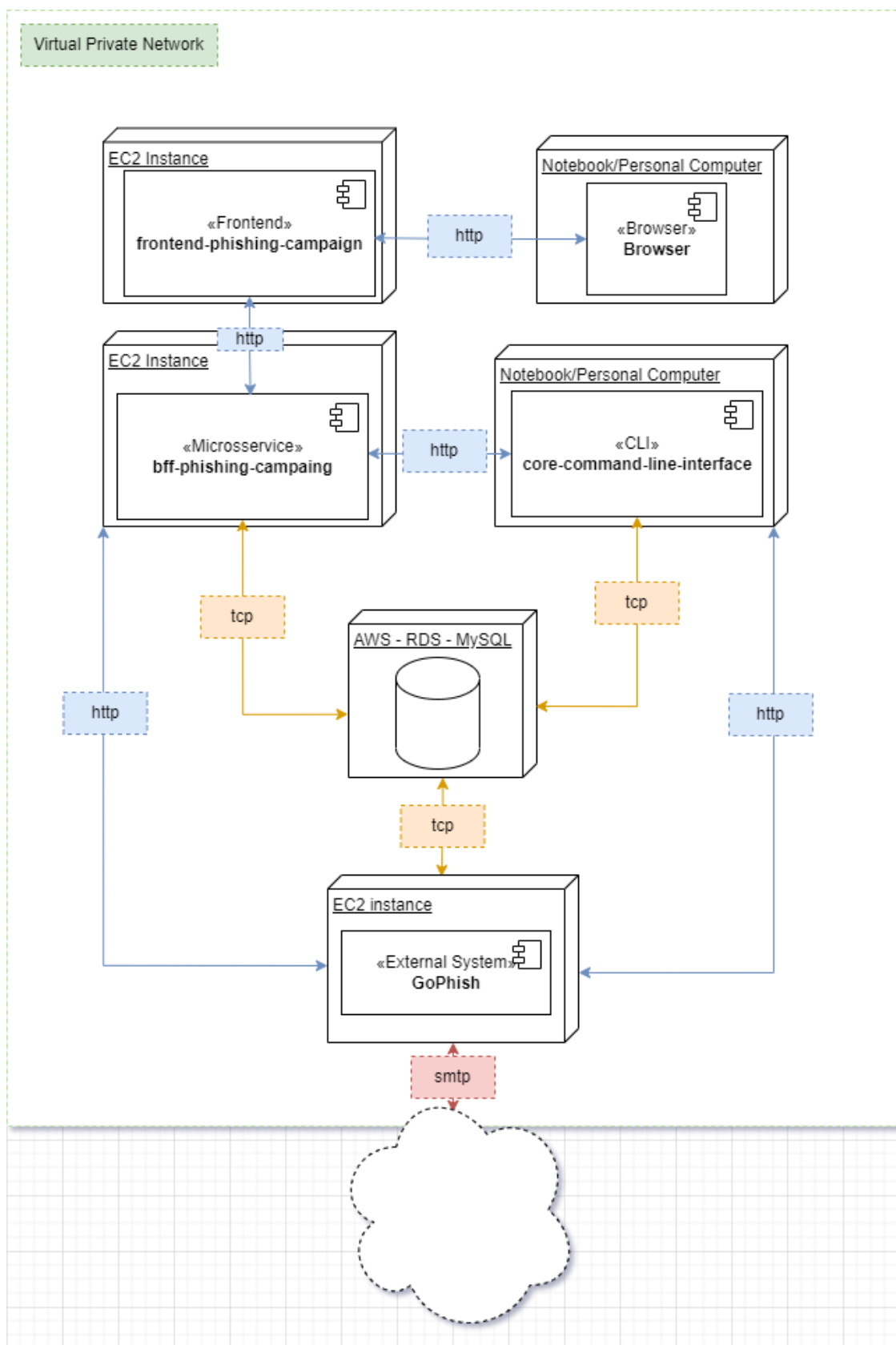


Figura 2 - Diagrama de Implantação

Item 4 - D

Esta seção tem por objetivo apresentar evidência da escolha do *framework Gophish*, citado anteriormente. Conforme citado, uma das grandes vantagens na escolha de sistema auxiliar, além da grande variedade de funções que ele oferece gratuitamente, é a disponibilização de uma API baseada em Python. Assim sendo, espera-se tirar grande vantagem realizando esta integração. Em termos mais técnicos, vemos na documentação da ferramenta as seguintes possibilidades de integração⁷:

- Conexão com o *Gophish*
- Administração de campanhas
- Administração de grupos
- Administração de *templates*
- Administração de *Landing Pages*
- Administração de perfis (Quais servidores *SMTP* serão usados ao enviar *emails*)

Item 5 - E

Por fim, esta seção é responsável por evidenciar a criação do modelo de entidade relacional pensado para o sistema de gerenciamento de phishing. Com este último artefato de software, espera-se ter todos os insumos necessários para a codificação do sistema nas próximas sprints.

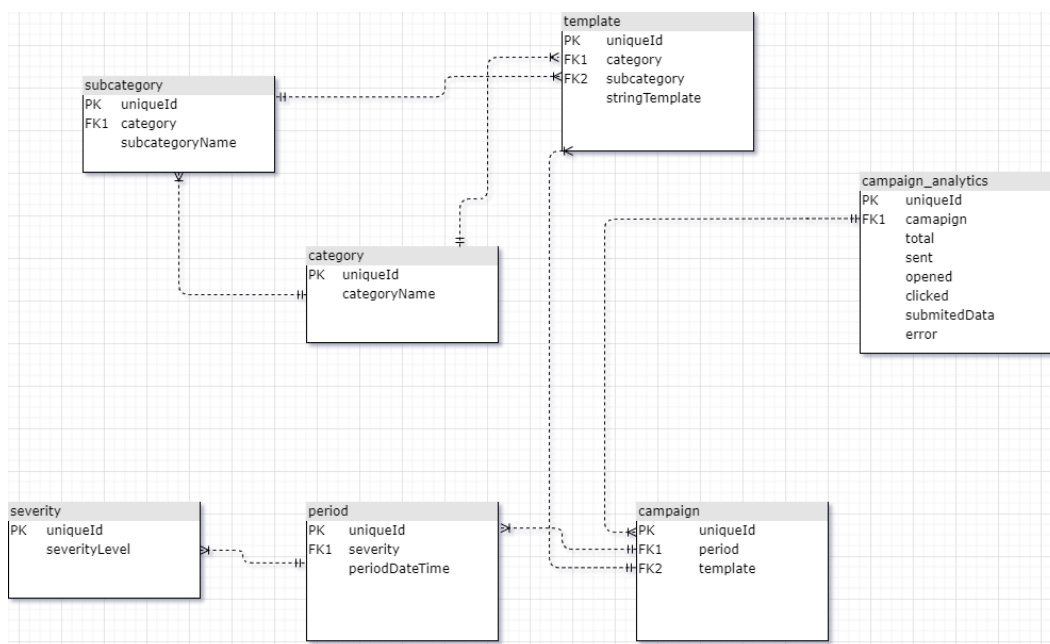


Figura 3 - Modelo de Entidade Relacional

⁷ Para estas e mais informações, o site oficial disponibiliza grande variedade de features: <https://docs.getgophish.com/python-api-client/>

2.1.2 Experiências vivenciadas

Como lições ou aprendidas ou experiências vivenciadas durante a realização desta primeira *sprint*, podemos destacar três principais pontos. O primeiro se refere à teoria estudada, que possui importância fundamental para o entendimento do mecanismo mental e sua relação com os ataques cibernéticos do tipo *phishing*. Através deste estudo, a base para definição arquitetural, escolha de tecnologias e modelagem relacional da base de dados.

O segundo ponto, refere-se ao planejamento técnico, cujas evidências podem ser encontradas em forma de artefatos de software (diagramas de componentes, implantação, modelo de entidade relacional). Durante esta fase, foi possível finalizar a *sprint* com uma visão mais clara do todo. Com o desenho do modelo de dados, houve um novo insight que será analisado na próxima *sprint*. Como a base armazena a campanha e seu período, será estudado uma forma de criar campanhas antecipadamente e deixar que a execução das mesmas seja realizada por um *cronjob* ou ferramenta semelhante.

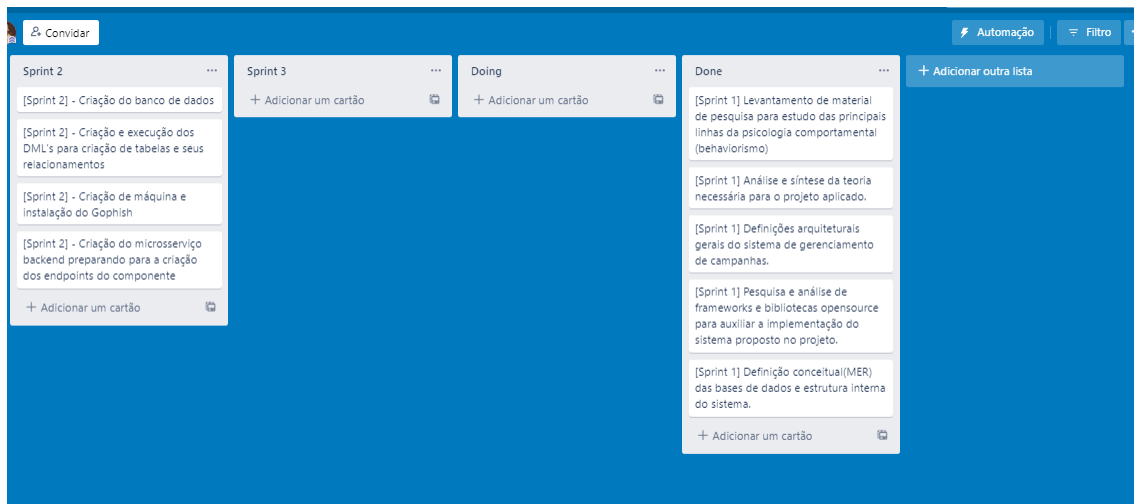
Por último, como lição aprendida, durante a execução da primeira *sprint*, foi realizada uma pesquisa para verificar se existia algum framework ou sistema *opensource* para auxiliar na implementação do sistema de gerenciamento de campanhas de *phishing*. Após alguns dias, chegou-se na conclusão da utilização do framework *Gophish*, mesma sugestão que o professor orientador havia dado na fase inicial do projeto.

2.2 Sprint 2

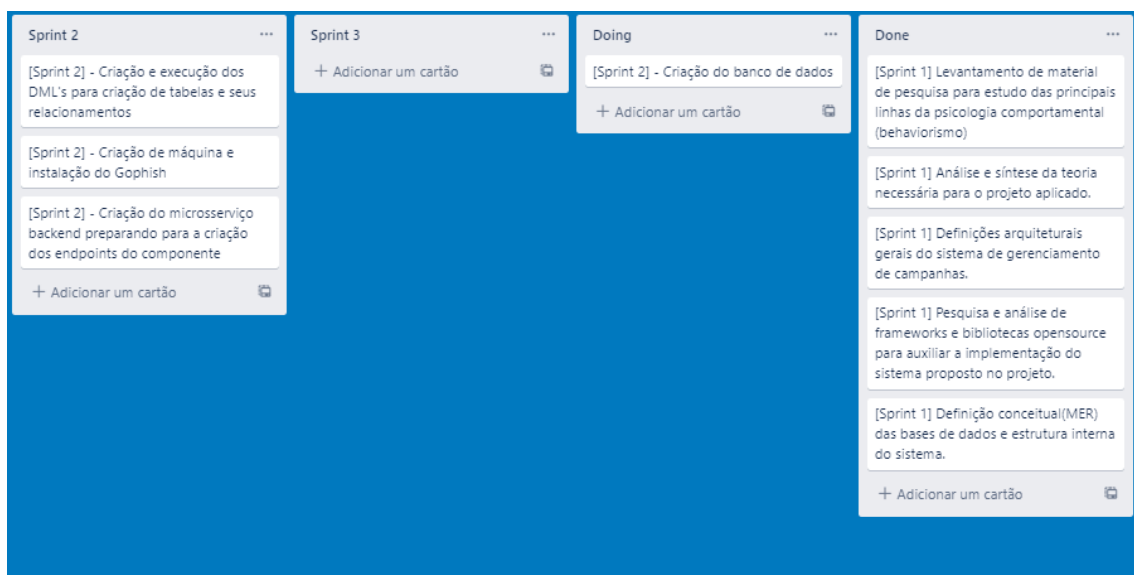
2.2.1 Solução

- Evidência do planejamento:

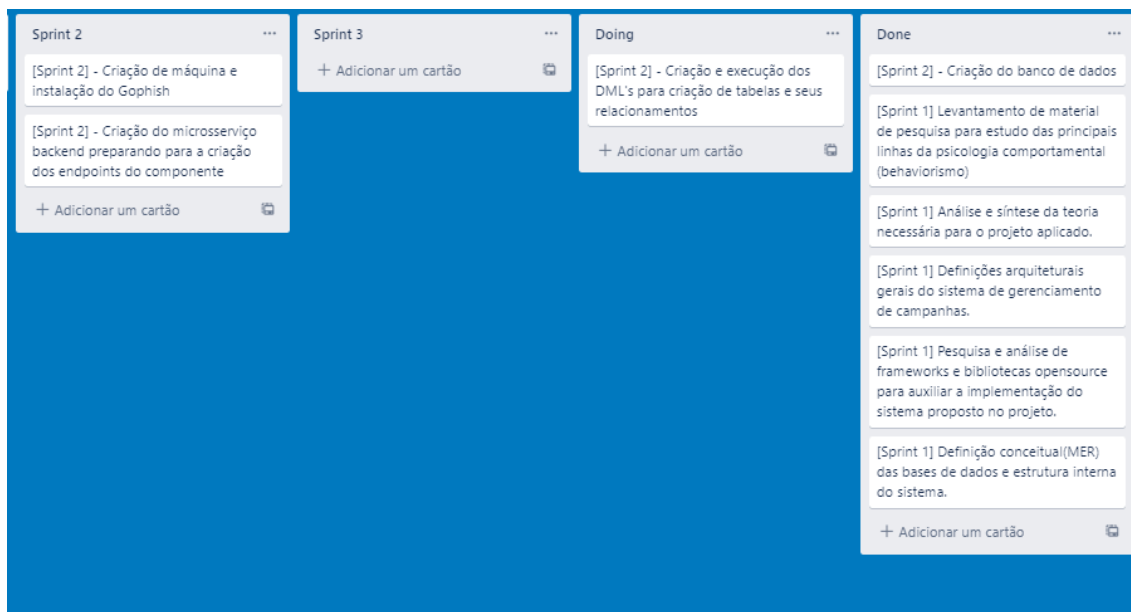
Inicialmente, antes de evidenciar o planejamento item a item, será disponibilizado logo a seguir a captura de tela contendo todas as atividades planejadas para a sprint de número dois.



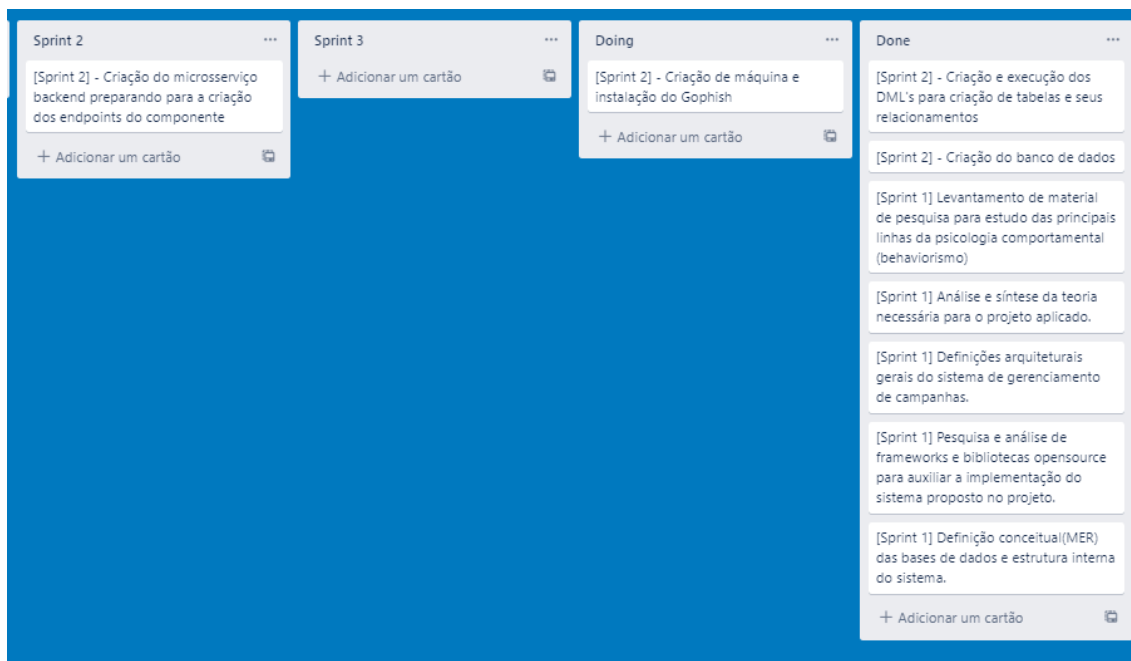
Para o item 1 da segunda sprint, a saber, “Criação do Banco de Dados”, o planejamento via *Trello* encontra-se abaixo:



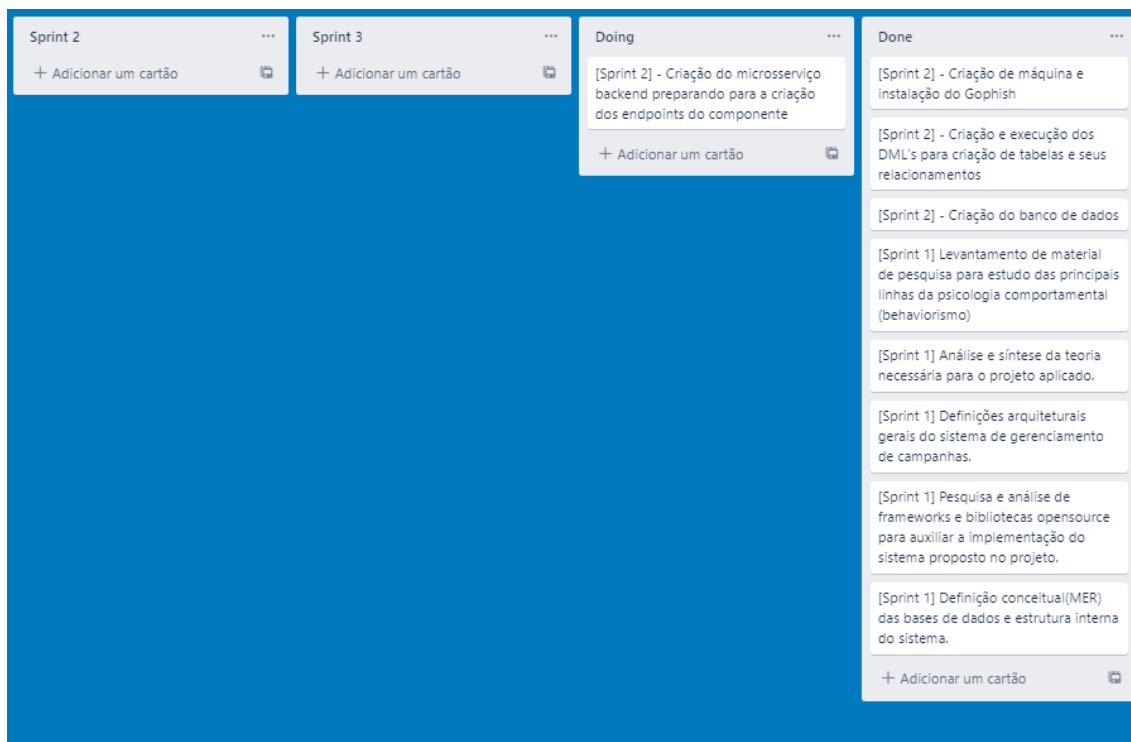
Para o item 2 da segunda sprint, a saber, “Criação e execução dos *DML*’s para criação de tabelas e seus relacionamentos”, o planejamento via *Trello* encontra-se abaixo:



Para o item 3 da segunda sprint, a saber, “Criação de máquina e instalação do *Gophish*”, o planejamento via *Trello* encontra-se abaixo:

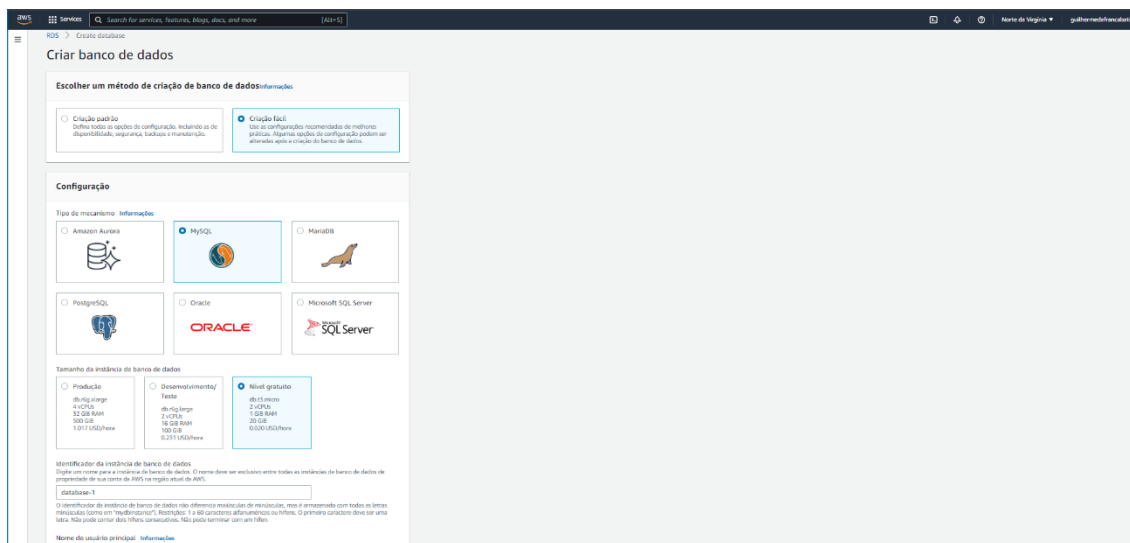


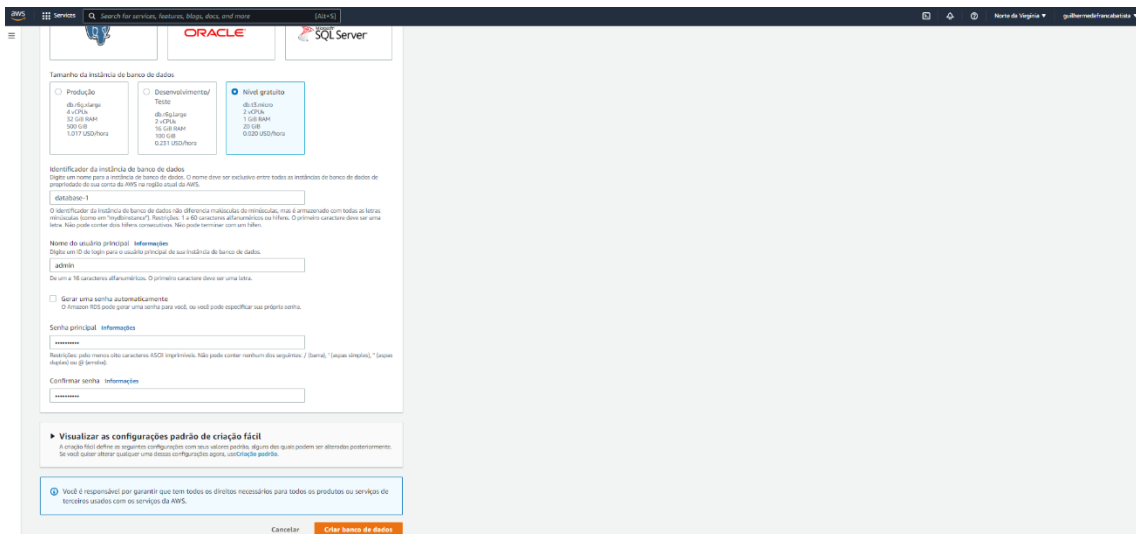
Para o item 4 da segunda sprint, a saber, “Criação do microserviço backend preparando para a criação dos *endpoints* do componente”, o planejamento via *Trello* encontra-se abaixo:



- Evidência da execução de cada requisito:

Para o primeiro item, “Criação do Banco de Dados”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.





Tamanho da instância de banco de dados

☐ Produção ☐ Desenvolvimento ☒ **Nível gratuito**

db.t3.micro
2 vCPUs
2 GiB RAM
15 GiB EBS
100 IOPS
0.031 I/Os/seg

Identificador da instância de banco de dados
Digite um nome para a instância de banco de dados. O nome deve ser exclusivo entre todos as instâncias de banco de dados de propriedade de sua conta da AWS na região atual da AWS.
database-1

Nome de usuário principal **Informações**
Digite um ID de login para o usuário principal de sua instância de banco de dados.
admin

☐ **Gerar uma senha automaticamente**
O Amazon RDS pode gerar uma senha para você, ou você pode especificar sua própria senha.

Senha principal **Informações**
Restrições: pelo menos oito caracteres ASCII imprimíveis. Não pode conter nenhum dos seguintes: / (barra), * (aspa simples), " (aspa dupla) ou @ (arroba).

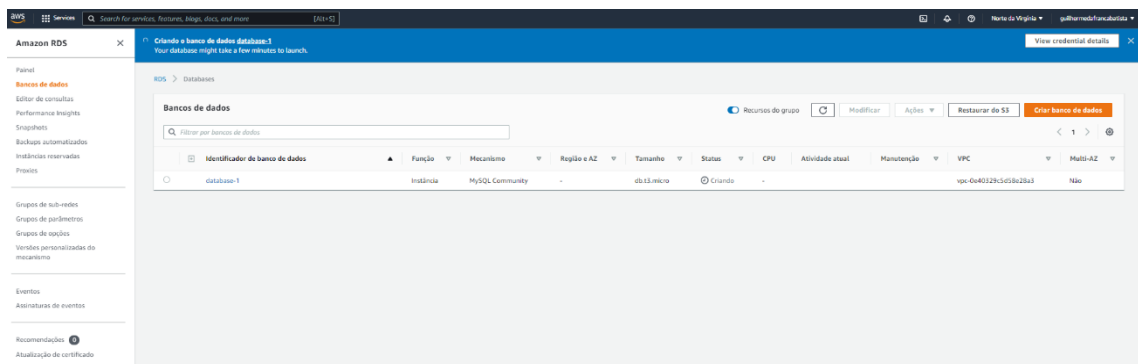
Confirmar senha **Informações**

Visualizar as configurações padrão de criação fácil
A criação fácil define as seguintes configurações com seus valores padrão, alguns dos quais podem ser alterados posteriormente. Se você quiser alterar ou substituir uma dessas configurações, clique em **Configurar parâmetros**.

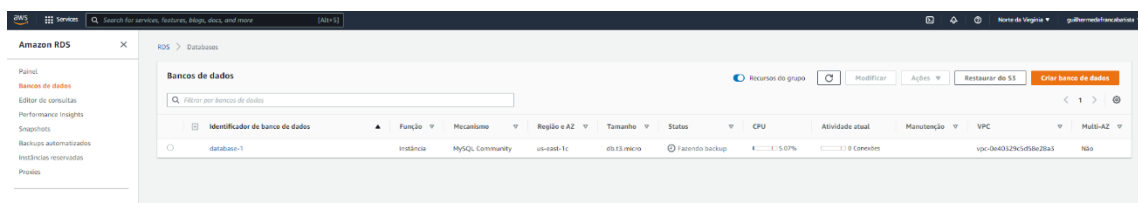
Visualizar as configurações padrão de criação fácil
Você é responsável por garantir que tem todos os direitos necessários para todos os produtos ou serviços de terceiros usados com os serviços da AWS.

Cancelar **Criar banco de dados**

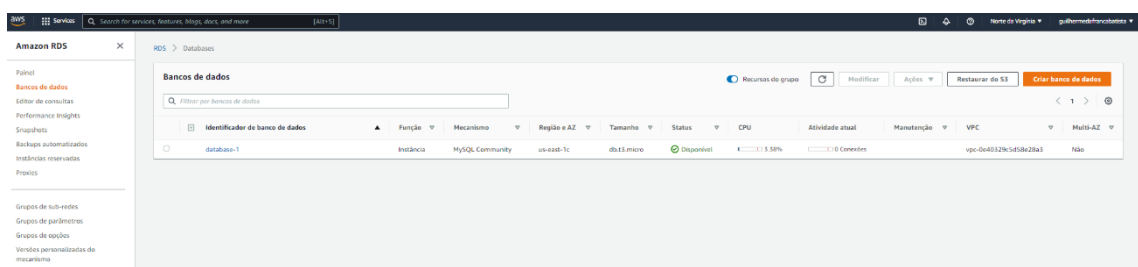
Conforme foi definido na primeira sprint, utilizaremos o banco de dados MySQL como motor. Desse modo, esta opção foi a escolhida no console da AWS e os próximos prints fazem parte do passo-a-passo da criação dessa infraestrutura PaaS.



Identificador de banco de dados	Função	Mecanismo	Região e AZ	Tamanho	Status	CPU	Atividade atual	Manutenção	VPC	Multi-AZ
database-1	Instância	MySQL Community	us-east-1c	db.t3.micro	Criando	100%	0%	0%	vpc-0a4329c5d8e28a3	Não



Identificador de banco de dados	Função	Mecanismo	Região e AZ	Tamanho	Status	CPU	Atividade atual	Manutenção	VPC	Multi-AZ
database-1	Instância	MySQL Community	us-east-1c	db.t3.micro	Agendo backup	100%	100%	0%	vpc-0a4329c5d8e28a3	Não



Identificador de banco de dados	Função	Mecanismo	Região e AZ	Tamanho	Status	CPU	Atividade atual	Manutenção	VPC	Multi-AZ
database-1	Instância	MySQL Community	us-east-1c	db.t3.micro	Disponível	100%	100%	0%	vpc-0a4329c5d8e28a3	Não

Para o segundo item, a saber, “Criação e execução dos DML's para criação de tabelas e seus relacionamentos”, listaremos abaixo todos os comandos utilizados criados para a criação das tabelas e seus relacionamentos.

Criação da base e uso para execução subsequente dos DML's listados abaixo:

```
1 CREATE DATABASE phishingmanagement;
2 USE phishingmanagement;
3
```

Tabela 1 - severity

```
4
5
6 CREATE TABLE severity (
7     uniqueid int NOT NULL AUTO_INCREMENT,
8     severitylevel varchar(255) NOT NULL,
9     PRIMARY KEY (uniqueid)
10 );
11
12
```

Tabela 2 - period

```
13
14 CREATE TABLE period (
15     uniqueid int NOT NULL AUTO_INCREMENT,
16     severity int,
17     perioddatetime varchar(255) NOT NULL,
18     PRIMARY KEY (uniqueid),
19     FOREIGN KEY (severity) REFERENCES severity(uniqueid)
20 );
21
22
```

Tabela 3 - category

```
23
24 CREATE TABLE category (
25     uniqueid int NOT NULL AUTO_INCREMENT,
26     name varchar(255) NOT NULL,
27     PRIMARY KEY (uniqueid)
28 );
29
30
```

Tabela 4 - subcategory

```

31
32 CREATE TABLE subcategory (
33     uniqueid int NOT NULL AUTO_INCREMENT,
34     category int,
35     name varchar(255) NOT NULL,
36     PRIMARY KEY (uniqueid),
37     FOREIGN KEY (category) REFERENCES category(uniqueid)
38 );
39
40
41

```

Tabela 5 - template

```

41
42 CREATE TABLE template (
43     uniqueid int NOT NULL AUTO_INCREMENT,
44     category int,
45     subcategory int,
46     stringtemplate varchar(255) NOT NULL,
47     PRIMARY KEY (uniqueid),
48     FOREIGN KEY (category) REFERENCES category(uniqueid),
49     FOREIGN KEY (subcategory) REFERENCES subcategory(uniqueid)
50 );
51
52

```

Tabela 6 - campaign

```

53
54 CREATE TABLE campaign (
55     uniqueid int NOT NULL AUTO_INCREMENT,
56     template int,
57     period int,
58     PRIMARY KEY (uniqueid),
59     FOREIGN KEY (template) REFERENCES template(uniqueid),
60     FOREIGN KEY (period) REFERENCES period(uniqueid)
61 );
62
63

```

Tabela 7 - campaignanalytics

```

64
65 CREATE TABLE campaignanalytics (
66     uniqueid int NOT NULL AUTO_INCREMENT,
67     campaign int,
68     total varchar(150),
69     sent varchar(150),
70     opened varchar(150),
71     clicked varchar(150),
72     submitteddata varchar(150),
73     error varchar(150),
74     PRIMARY KEY (uniqueid),
75     FOREIGN KEY (campaign) REFERENCES campaign(uniqueid)
76 );
77

```

Para o terceiro item, a saber, “Criação de máquina e instalação do *Gophish*”, listaremos abaixo todos os passos realizados para a criação da máquina, instalação do *Gophish* e etapas adicionais com integrações necessárias. É importante dizer aqui que, apesar de o desenho arquitetural orientar a criação de uma máquina *EC2* para isso, todos os passos podem ser realizados em máquinas virtuais em servidores da empresa que utilizará este sistema, pois, assim como o ambiente local, o ambiente *AWS* também só poderá ser acessado dentro da empresa através de sua rede por questões de segurança. Dito isso, caso não se tenha recursos necessários para criar máquinas *EC2*, a criação e desenvolvimento *On Premise* é uma alternativa viável.

Dentro do sistema operacional, neste exemplo, sistemas baseados em Debian, antes de mais nada, realize a atualização de pacotes e instalação dos mesmos necessários para o bom funcionamento do sistema operacional.

```

1 sudo apt-get update
2 sudo apt-get upgrade
3

```

Após, faz-se necessário realizar o download do *Gophish*. Para isso, verificar se os programas básicos de rede estão instalados antes de realizar o seguinte comando:

```

4
5 wget https://github.com/gophish/gophish/releases/download/0.7.1/gophish-v0.7.1-linux-64bit.zip
6
7

```

Com este binário baixado, faz-se necessário realizar a descompactação do mesmo que se encontra no formato *.zip*. Para isto também, verificar se o compactador e descompactador de *zip* está instalado no sistema operacional e realiza os seguintes procedimentos:

```

7
8 sudo unzip gophish-v0.7.1-linux-64bit.zip -d /opt/gophish
9

```

Para verificar o sucesso do procedimento, entre no diretório onde o inário *zip* foi descompactado e liste os itens. O resultado deve ser algo parecido como isto:

```

10
11 cd /opt/gophish
12 ls
13 config.json    db    gophish    LICENSE    README.md    static    templates    VERSION
14

```

Por fim, antes de executar o *Gophish* de facto, necessitamos realizar a configuração da base de dados, que é o *MySQL*. Para isto, a alteração de um arquivo chamado *config.json*. Para isso, as linhas que contem o nome e caminho do banco de dados devem ser alteradas. As alterações devem seguir o seguinte padrão:

```

15
16
17 "db_name" : "mysql",
18 "db_path" : "root:@(:3306)/gophish?charset=utf8&parseTime=True&loc=UTC"

```

O próximo passo é a execução para utilização do *Gophish*, que é realizado através do seguinte procedimento. Entrar no diretório */opt/gophish* e executar seu binário:

```

20
21 cd /opt/gophish
22 sudo ./gophish

```

Em relação ao último item, a saber, “Criação do microserviço *backend* preparando para a criação dos *endpoints* do componente”, um projeto baseado na linguagem de programação python foi criado, com a divisão de diretórios e domínios iniciais para o desenvolvimento da lógica do sistema de gerenciamento de phishing. De modo geral, os diretórios foram pensados da seguinte maneira:

/controller - este diretório é responsável por lidar com o direcionamento das requisições feitas ao microserviço, sendo responsável por resolver para qual domínio ou orquestração de domínios a ação deverá ser realizada.

/management - este diretório é responsável por lidar com o domínio do sistema de gerenciamento propriamente dito. Aqui devem estar as entidades, serviços e interfaces em subdiretórios para que o código fique organizado e cada arquivo responsável por sua função.

/gophish - este diretório é responsável por lidar com a integração com o sistema *Gophish* através de sua api nativa. Integração, gatilhos, orquestrações e atualizações. Assim como o diretório anterior, este conterá subdiretórios e arquivos separados por funções bem definidas.

- Evidência dos resultados:

Como evidência do primeiro requisito, a “Criação do Banco de Dados”, utilizaremos o console para provar que a base de dados foi criada e pudemos nos conectar a ela.

```
guilherme@guilherme:~$ mysql -h database-1.chaffbk5kmsq.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Como evidência do seguinte requisito, a “Criação e execução dos DML's para criação de tabelas e seus relacionamentos”, mostraremos as tabelas criadas bem como a base de dados criada para o projeto.

Criação e uso do banco de dados phishingmanagement

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phishingmanagement |
| sys |
+-----+
5 rows in set (0,17 sec)

mysql> use phishingmanagement;
Database changed
mysql>
```

Criação da tabela 1 - severity

```
mysql> CREATE TABLE severity (
  -> uniqueid int NOT NULL AUTO_INCREMENT,
  -> severitylevel varchar(255) NOT NULL,
  -> PRIMARY KEY (uniqueid)
  -> );
Query OK, 0 rows affected (0,20 sec)

mysql> desc severity;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| uniqueid       | int           | NO   | PRI | NULL    | auto_increment |
| severitylevel  | varchar(255)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,20 sec)
```

Criação da tabela 2 - period

```
mysql> CREATE TABLE period (
  -> uniqueid int NOT NULL AUTO_INCREMENT,
  -> severity int,
  -> perioddatetime varchar(255) NOT NULL,
  -> PRIMARY KEY (uniqueid),
  -> FOREIGN KEY (severity) REFERENCES severity(uniqueid)
  -> );
Query OK, 0 rows affected (0,21 sec)

mysql> desc period;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| uniqueid       | int           | NO   | PRI | NULL    | auto_increment |
| severity       | int           | YES  | MUL | NULL    |                |
| perioddatetime | varchar(255)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0,18 sec)
```

Criação da tabela 3 - category

```
mysql> CREATE TABLE category (
  -> uniqueid int NOT NULL AUTO_INCREMENT,
  -> name varchar(255) NOT NULL,
  -> PRIMARY KEY (uniqueid)
  -> );
Query OK, 0 rows affected (0,20 sec)

mysql> desc category;
+-----+-----+-----+-----+-----+-----+
| Field  | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| uniqueid | int           | NO   | PRI | NULL    | auto_increment |
| name    | varchar(255)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,17 sec)
```


Criação da tabela 4 - subcategory

```
mysql> CREATE TABLE subcategory (
-> uniqueid int NOT NULL AUTO_INCREMENT,
-> category int,
-> name varchar(255) NOT NULL,
-> PRIMARY KEY (uniqueid),
-> FOREIGN KEY (category) REFERENCES category(uniqueid)
-> );
```

Query OK, 0 rows affected (0,20 sec)

```
mysql> desc subcategory;
```

Field	Type	Null	Key	Default	Extra
uniqueid	int	NO	PRI	NULL	auto_increment
category	int	YES	MUL	NULL	
name	varchar(255)	NO		NULL	

3 rows in set (0,16 sec)

Criação da tabela 5 - template

```
mysql> CREATE TABLE template (
-> uniqueid int NOT NULL AUTO_INCREMENT,
-> category int,
-> subcategory int,
-> stringtemplate varchar(255) NOT NULL,
-> PRIMARY KEY (uniqueid),
-> FOREIGN KEY (category) REFERENCES category(uniqueid),
-> FOREIGN KEY (subcategory) REFERENCES subcategory(uniqueid)
-> );
```

Query OK, 0 rows affected (0,20 sec)

```
mysql> desc template;
```

Field	Type	Null	Key	Default	Extra
uniqueid	int	NO	PRI	NULL	auto_increment
category	int	YES	MUL	NULL	
subcategory	int	YES	MUL	NULL	
stringtemplate	varchar(255)	NO		NULL	

4 rows in set (0,16 sec)

Criação da tabela 6 - campaign

```
mysql> CREATE TABLE campaign (
  -> uniqueid int NOT NULL AUTO_INCREMENT,
  -> template int,
  -> period int,
  -> PRIMARY KEY (uniqueid),
  -> FOREIGN KEY (template) REFERENCES template(uniqueid),
  -> FOREIGN KEY (period) REFERENCES period(uniqueid)
  -> );
Query OK, 0 rows affected (0,20 sec)

mysql> desc campaign;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| uniqueid | int | NO | PRI | NULL | auto_increment |
| template | int | YES | MUL | NULL | |
| period | int | YES | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0,17 sec)
```

Criação da tabela 7 - campaignanalytics

```
mysql> CREATE TABLE campaignanalytics (
  -> uniqueid int NOT NULL AUTO_INCREMENT,
  -> campaign int,
  -> total varchar(150),
  -> sent varchar(150),
  -> opened varchar(150),
  -> clicked varchar(150),
  -> submitteddata varchar(150),
  -> error varchar(150),
  -> PRIMARY KEY (uniqueid),
  -> FOREIGN KEY (campaign) REFERENCES campaign(uniqueid)
  -> );
Query OK, 0 rows affected (0,20 sec)

mysql> desc campaignanalytics;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| uniqueid | int | NO | PRI | NULL | auto_increment |
| campaign | int | YES | MUL | NULL | |
| total | varchar(150) | YES | | NULL | |
| sent | varchar(150) | YES | | NULL | |
| opened | varchar(150) | YES | | NULL | |
| clicked | varchar(150) | YES | | NULL | |
| submitteddata | varchar(150) | YES | | NULL | |
| error | varchar(150) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0,36 sec)
```

Em relação ao requisito “Criação de máquina e instalação do *Gophish*”, evidenciamos através de prints os passos e procedimentos necessários para que o *Gophish* possa ser instalado em sistemas operacionais baseados em *Debian*. As evidências servirão também como uma espécie de tutorial para que possa ser reproduzido sem grandes problemas.

Os dois primeiros prints se referem ao download do binário em formato .zip do *Gophish*. Nesta etapa utilizamos o utilitário de rede *wget* para fazer a requisição *https* ao repositório do GitHub onde se encontra a versão que será utilizada.

```
guilherme@guilherme:~$ wget https://github.com/gophish/gophish/releases/download/v0.7.1/gophish-v0.7.1-linux-64bit.zip
--2022-07-03 13:40:08-- https://github.com/gophish/gophish/releases/download/v0.7.1/gophish-v0.7.1-linux-64bit.zip
Resolving github.com (github.com)... 20.201.28.151
Connecting to github.com (github.com)|20.201.28.151|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/14508450/3d4d4b00-b427-11e8-972c-cdc06ef53c5b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20220703%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220703T164008Z&X-Amz-Expires=300&X-Amz-Signature=152f827c48f6307f985023508c58478392cecdf6c414329c3d5d3881a5548889&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=14508450&response-content-disposition=attachment%3B%20filename%3Dgophish-v0.7.1-linux-64bit.zip&response-content-type=application%2Foctet-stream [following]
--2022-07-03 13:40:08-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/14508450/3d4d4b00-b427-11e8-972c-cdc06ef53c5b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20220703%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220703T164008Z&X-Amz-Expires=300&X-Amz-Signature=152f827c48f6307f985023508c58478392cecdf6c414329c3d5d3881a5548889&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=14508450&response-content-disposition=attachment%3B%20filename%3Dgophish-v0.7.1-linux-64bit.zip&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.110.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28260687 (27M) [application/octet-stream]

Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28260687 (27M) [application/octet-stream]
Saving to: 'gophish-v0.7.1-linux-64bit.zip.1'

gophish-v0.7.1-linu 100%[=====>] 26,95M  998KB/s  in 41s

2022-07-03 13:40:50 (679 KB/s) - 'gophish-v0.7.1-linux-64bit.zip.1' saved [28260687/28260687]

guilherme@guilherme:~$
```

Este passo se refere à descompactação do arquivo previamente baixado. Nele utilizamos o utilitário de descompactação `zip` para realizar o procedimento e transferir o conteúdo para um novo diretório chamado `gophish` dentro do diretório `/opt`.

```
guilherme@guilherme:~$ sudo unzip gophish-v0.7.1-linux-64bit.zip -d
/opt/gophish
[sudo] password for guilherme:
Archive:  gophish-v0.7.1-linux-64bit.zip
replace /opt/gophish/static/js/dist/vendor.min.js? [y]es, [n]o, [A]ll, [N]one,
[r]ename: A
  inflating: /opt/gophish/static/js/dist/vendor.min.js
  inflating: /opt/gophish/static/js/dist/app/users.min.js
  inflating: /opt/gophish/static/js/dist/app/dashboard.min.js
  inflating: /opt/gophish/static/js/dist/app/templates.min.js
  inflating: /opt/gophish/static/js/dist/app/campaigns.min.js
  inflating: /opt/gophish/static/js/dist/app/landing_pages.min.js
  inflating: /opt/gophish/static/js/dist/app/gophish.min.js
  inflating: /opt/gophish/static/js/dist/app/campaign_results.min.js
  inflating: /opt/gophish/static/js/dist/app/settings.min.js
  inflating: /opt/gophish/static/js/dist/app/sending_profiles.min.js
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/contents.css
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/LICENSE.md
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/styles.js
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/CHANGES.md
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/ckeditor.js
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/build-config.js
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/config.js
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/README.md
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/plugins/icons.png
  inflating: /opt/gophish/static/js/src/vendor/ckeditor/plugins/icons_hidpi.png
ders.sql
  inflating: /opt/gophish/db/db_mysql/migrations/20180524203752_0.7.0_result_la
st_modified.sql
  inflating: /opt/gophish/db/db_mysql/migrations/20170828220440_0.4_utc_dates.s
ql
  inflating: /opt/gophish/db/db_mysql/migrations/20160217211342_0.1.2_create_fr
om_col_results.sql
  inflating: /opt/gophish/db/db_mysql/migrations/20160317214457_0.2_redirect_ur
l.sql
  inflating: /opt/gophish/templates/dashboard.html
  inflating: /opt/gophish/templates/campaign_results.html
  inflating: /opt/gophish/templates/landing_pages.html
  inflating: /opt/gophish/templates/login.html
  inflating: /opt/gophish/templates/docs.html
  inflating: /opt/gophish/templates/flashs.html
  inflating: /opt/gophish/templates/base.html
  inflating: /opt/gophish/templates/campaigns.html
  inflating: /opt/gophish/templates/settings.html
  inflating: /opt/gophish/templates/sending_profiles.html
  inflating: /opt/gophish/templates/gophish.apib
  inflating: /opt/gophish/templates/templates.html
  inflating: /opt/gophish/templates/register.html
  inflating: /opt/gophish/templates/users.html
  inflating: /opt/gophish/README.md
  inflating: /opt/gophish/VERSION
  inflating: /opt/gophish/LICENSE
  inflating: /opt/gophish/config.json
  inflating: /opt/gophish/gophish
guilherme@guilherme:~$
```

Este é o passo de verificação das etapas anteriores. Nele, entramos no diretório onde o binário foi descompactado e listamos os itens. O resultado deve ser algo muito semelhante ao print abaixo.

```
guilherme@guilherme:~$ cd /opt/gophish/
guilherme@guilherme:~/opt/gophish$ ls
config.json  gophish_admin.crt  LICENSE  templates
db           gophish_admin.key  README.md  VERSION
gophish      gophish.db         static
guilherme@guilherme:~/opt/gophish$
```

Este é o penúltimo passo, a mudança do banco de dados 'nativo' da aplicação para o RDS MySQL que está sendo utilizado no projeto. Como dito na seção de evidência da execução, duas linhas devem ser alteradas, a que define o motor de banco de dados e o caminho ou path do mesmo. Os prints a seguir evidenciam como a troca deve ser feita. Evidentemente, neste e em todos os outros os outros prints, informações sensíveis estão sendo censuradas para evitar que o autor do projeto tenha suas contas e informações expostas.



```
1 {
2     "admin_server": {
3         "listen_url": "[REDACTED]:3333",
4         "use_tls": true,
5         "cert_path": "gophish_admin.crt",
6         "key_path": "gophish_admin.key"
7     },
8     "phish_server": {
9         "listen_url": "0.0.0.0:80",
10        "use_tls": false,
11        "cert_path": "example.crt",
12        "key_path": "example.key"
13    },
14    "db_name": "sqlite3",
15    "db_path": "gophish.db",
16    "migrations_prefix": "db/db_",
17    "contact_address": ""
18 }
```



```

Open  [icon] *config.json [Read-Only] /opt/gophish Save [icon] [icon] [icon]
1 {
2     "admin_server": {
3         "listen_url": "[redacted]:3333",
4         "use_tls": true,
5         "cert_path": "gophish_admin.crt",
6         "key_path": "gophish_admin.key"
7     },
8     "phish_server": {
9         "listen_url": "0.0.0.0:80",
10        "use_tls": false,
11        "cert_path": "example.crt",
12        "key_path": "example.key"
13    },
14    "db_name": "mysql",
15    "db_path": "aqui-a-string-de-conexao-com-o-mysql-rds",
16    "migrations_prefix": "db/db_",
17    "contact_address": ""
18 }

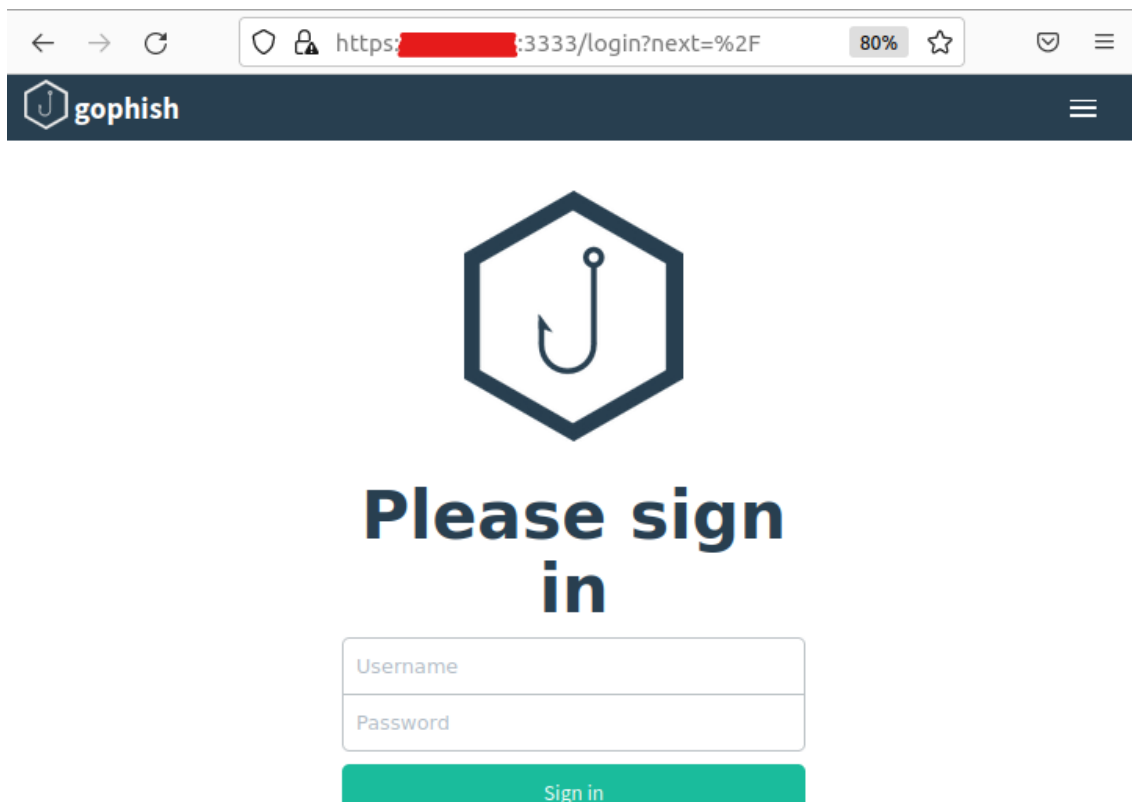
```

Por fim, a execução do Gophish propriamente dito. Após a realização de todas estas configurações iniciais, deve-se ao diretório do Gophish e executá-lo. Os prints a seguir demonstram a evidência do resultado do último passo deste requisito.

```

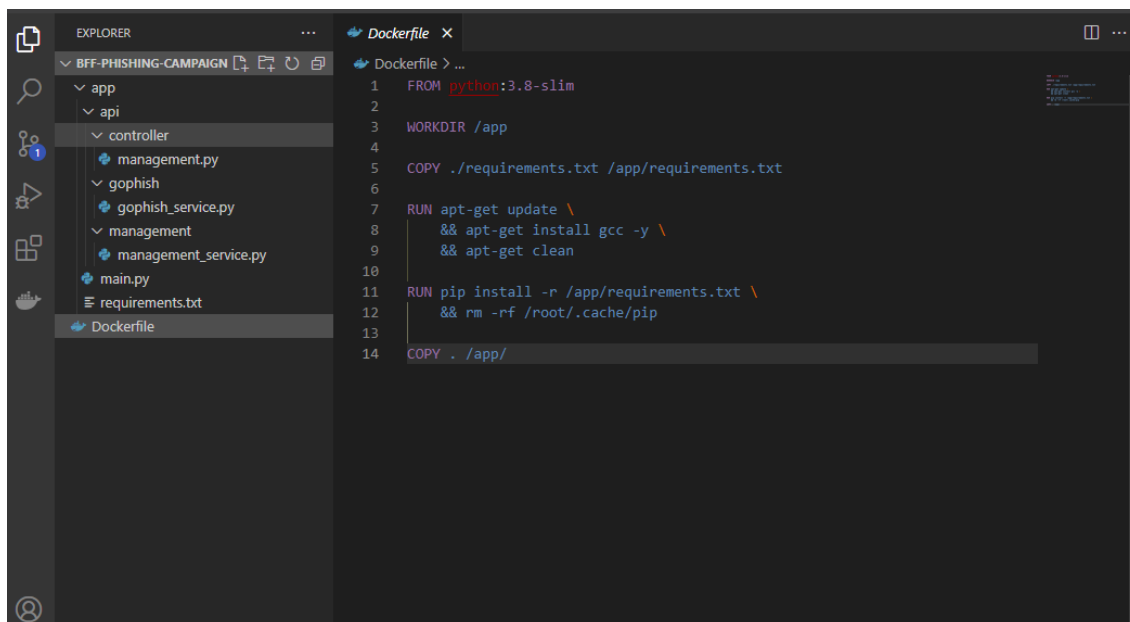
guilherme@guilherme [redacted]:/opt/gophish$ sudo ./gophish
time="2022-07-03T13:55:33-03:00" level=warning msg="No contact address has been
configured."
time="2022-07-03T13:55:33-03:00" level=warning msg="Please consider adding a co
ntact_address entry in your config.json"
goose: no migrations to run. current version: 20180830215615
time="2022-07-03T13:55:33-03:00" level=info msg="Background Worker Started Succ
essfully - Waiting for Campaigns"
time="2022-07-03T13:55:33-03:00" level=info msg="Starting phishing server at ht
tp:[redacted]:80"
time="2022-07-03T13:55:33-03:00" level=info msg="Starting admin server at https
:[redacted]:3333"

```



Em relação ao último item proposto para esta segunda sprint, a saber, “Criar microserviço *backend* preparando para a criação dos *endpoints* do componente”, será mostrado a seguir os passos seguidos na criação desta base para posterior desenvolvimento na terceira sprint.

Criou-se um diretório com o nome do projeto, chamado “*bff-phishing-campaign*” e, dentro do mesmo, criamos a estrutura inicial conforme descrito na seção xpto. Além do projeto em si, procurou-se deixar o microserviço simples de executar em qualquer contexto através da containerização do mesmo através de um *Dockerfile*. Como evidência desta etapa, o print da seção inicial pode ser encontrado abaixo.



A evolução e resultados do desenvolvimento do código, bem como de todo este trabalho, pode ser realizada através do seguinte repositório no GitHub: https://github.com/akademichensch/igti-mba-cybersecurity/tree/main/project_and_conceptual_model/bff-phishing-campaign.

Para o projeto como um todo, o diretório raiz é este: <https://github.com/akademichensch/igti-mba-cybersecurity>.

2.2.2 Experiências vivenciadas

Como lições aprendidas ou experiências vivenciadas durante a execução desta segunda sprint, podemos destacar dois principais pontos. O primeiro deles se refere a execução desta etapa em relação à execução da anterior. Notou-se que a primeira sprint foi de fato essencial para a segunda; desde a síntese teórica, definição de política até a construção dos artefatos iniciais de software, cada um destes elementos foi aproveitado para a execução da sprint subsequente e espera-se que estes frutos sejam colhidos na execução da última sprint.

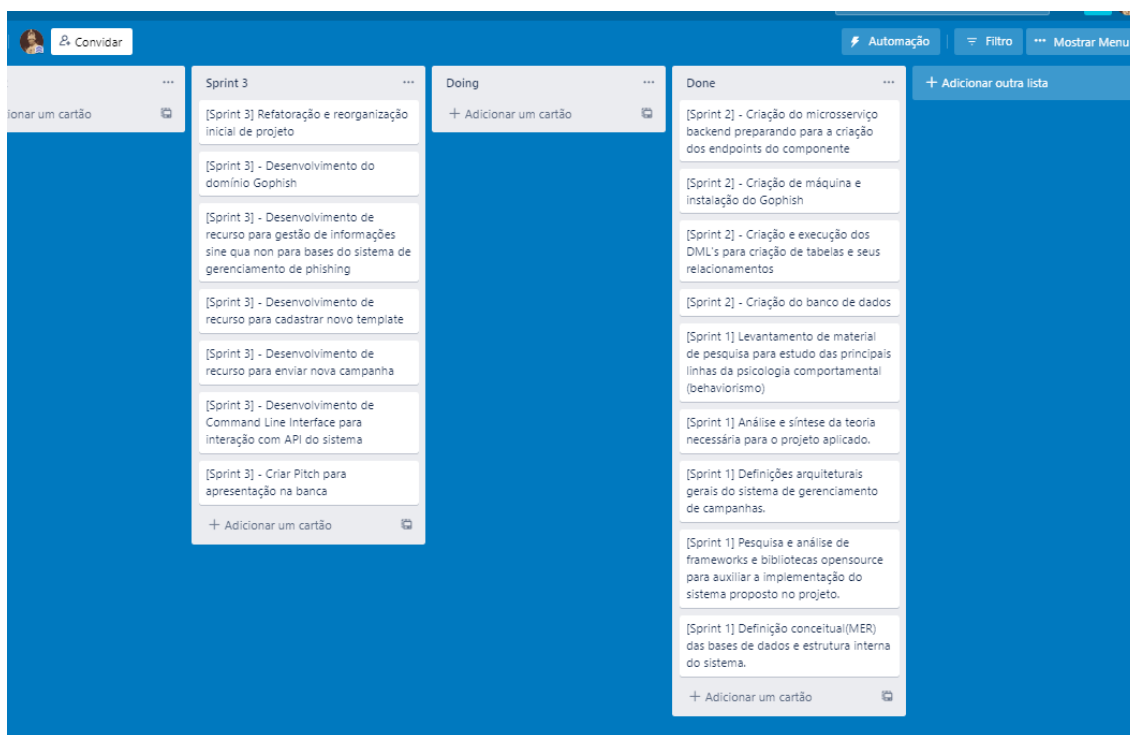
O segundo e último ponto, refere-se ao planejamento do projeto. Apesar de bem planejado e escopo definido, notou-se que o tempo é um recurso que sofre interferências externas não anunciadas previamente. Dito isso, é possível que todo o desenho arquitetural não seja contemplado, mas o mínimo produto viável para uso através de uma interface de linha de comando será realizado evidentemente.

2.3 Sprint 3

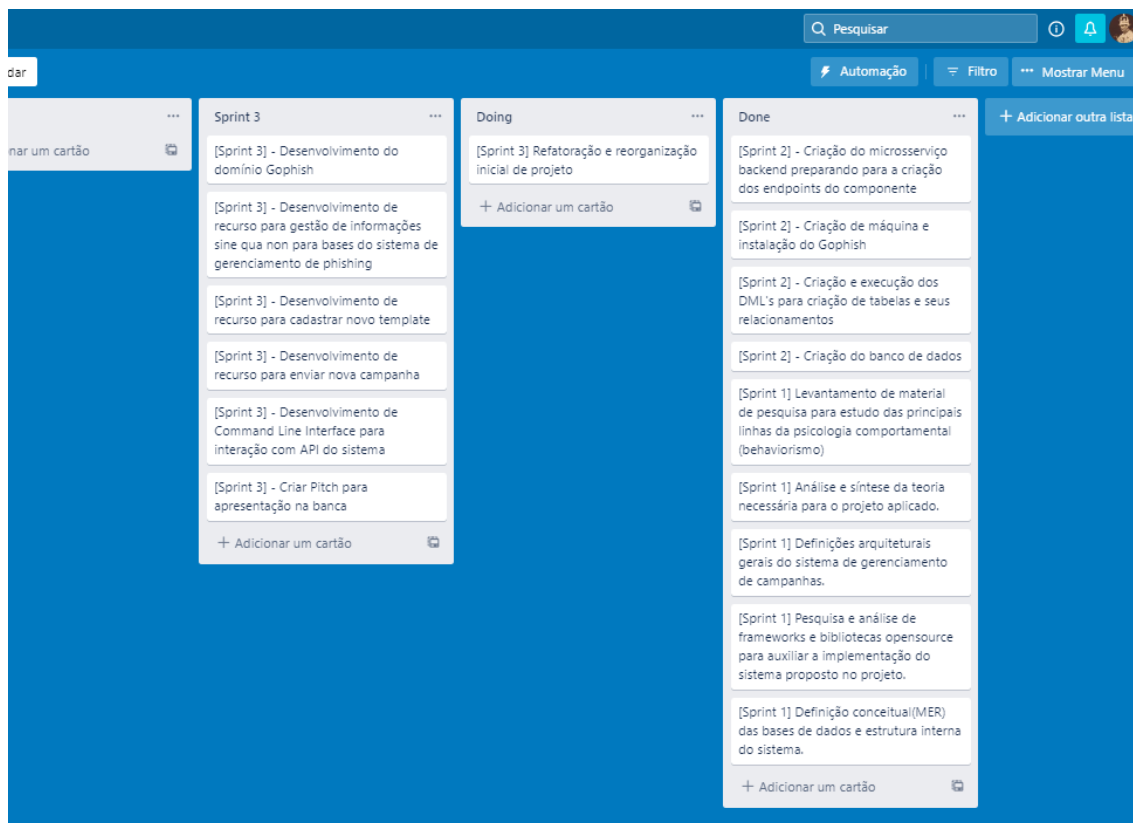
2.3.1 Solução

- Evidência do planejamento:

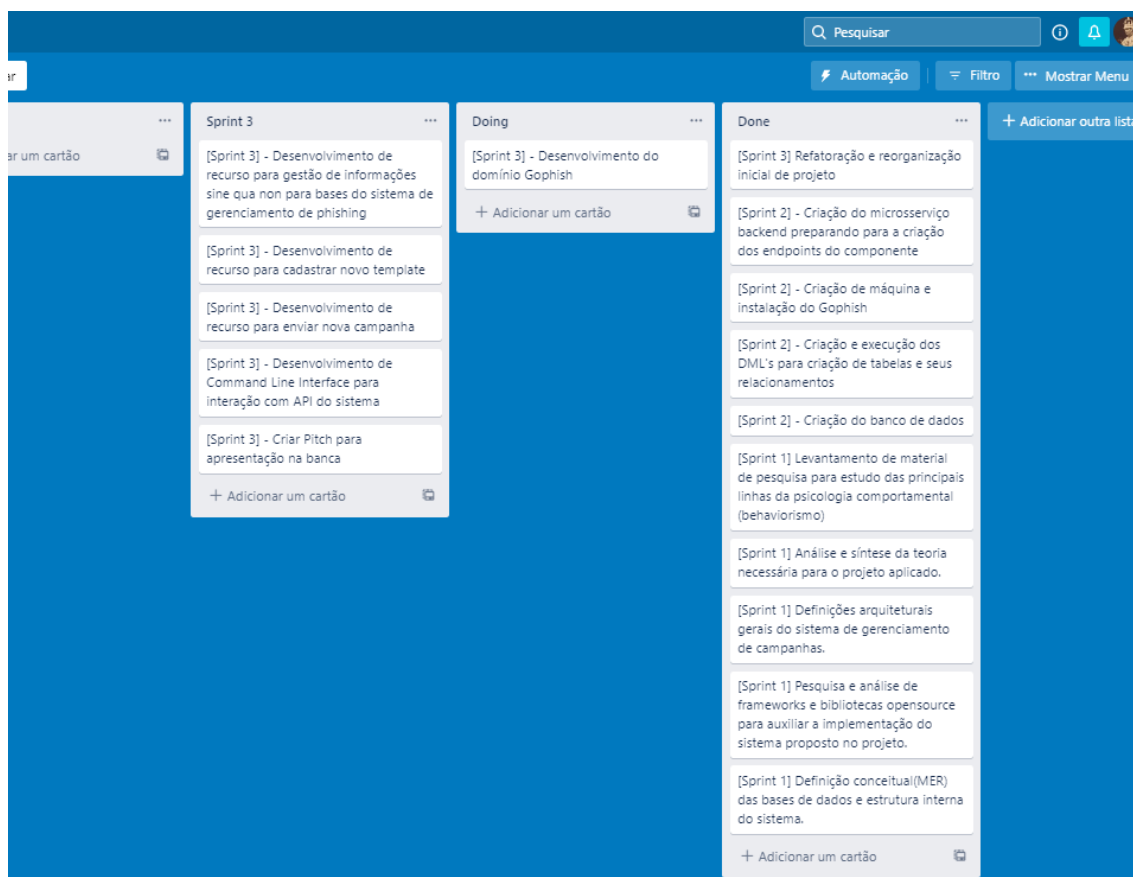
Inicialmente, antes de evidenciar o planejamento item a item, será disponibilizado logo a seguir a captura de tela contendo todas as atividades planejadas para a última sprint, de número três.



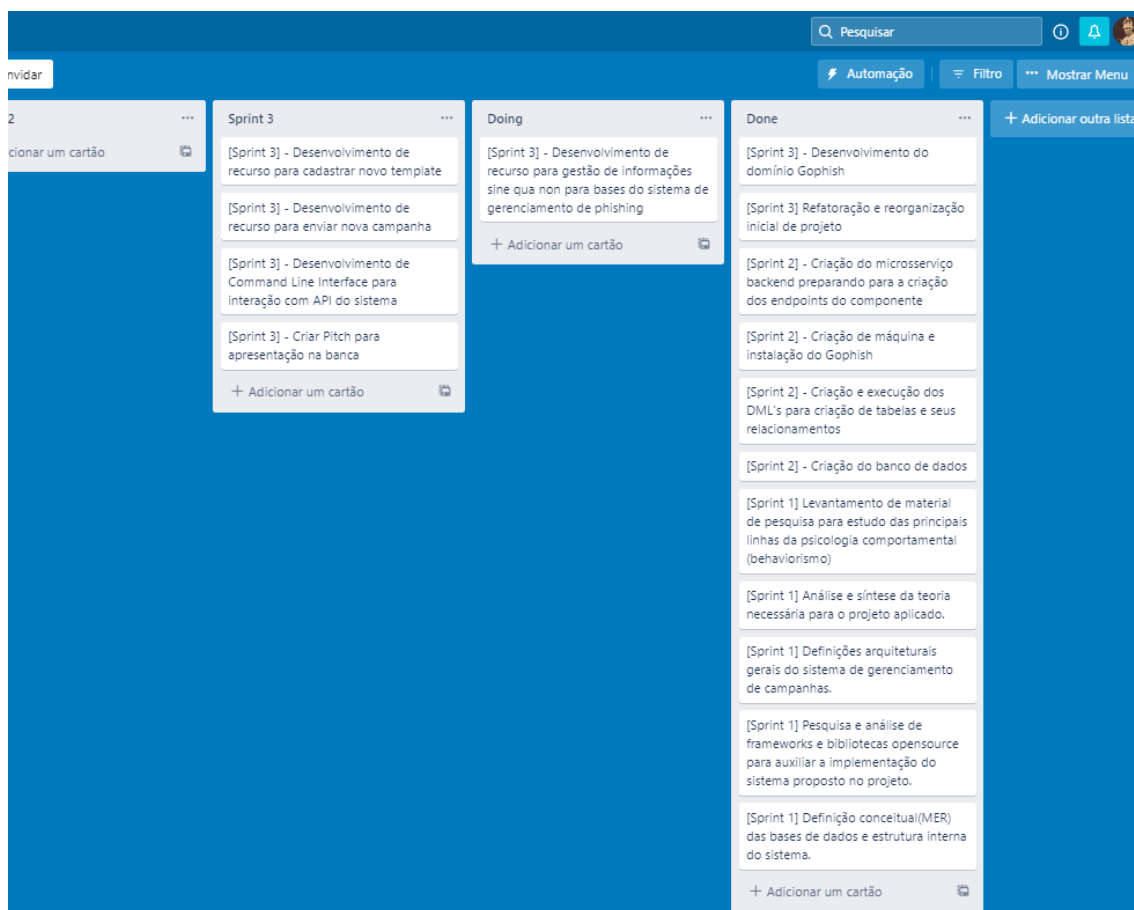
Para o item 1 da terceira sprint, a saber, “Refatoração e reorganização inicial de projeto”, o planejamento via *Trello* encontra-se abaixo:



Para o item 2 da terceira sprint, a saber, “Desenvolvimento do domínio Gophish”, o planejamento via *Trello* encontra-se abaixo:



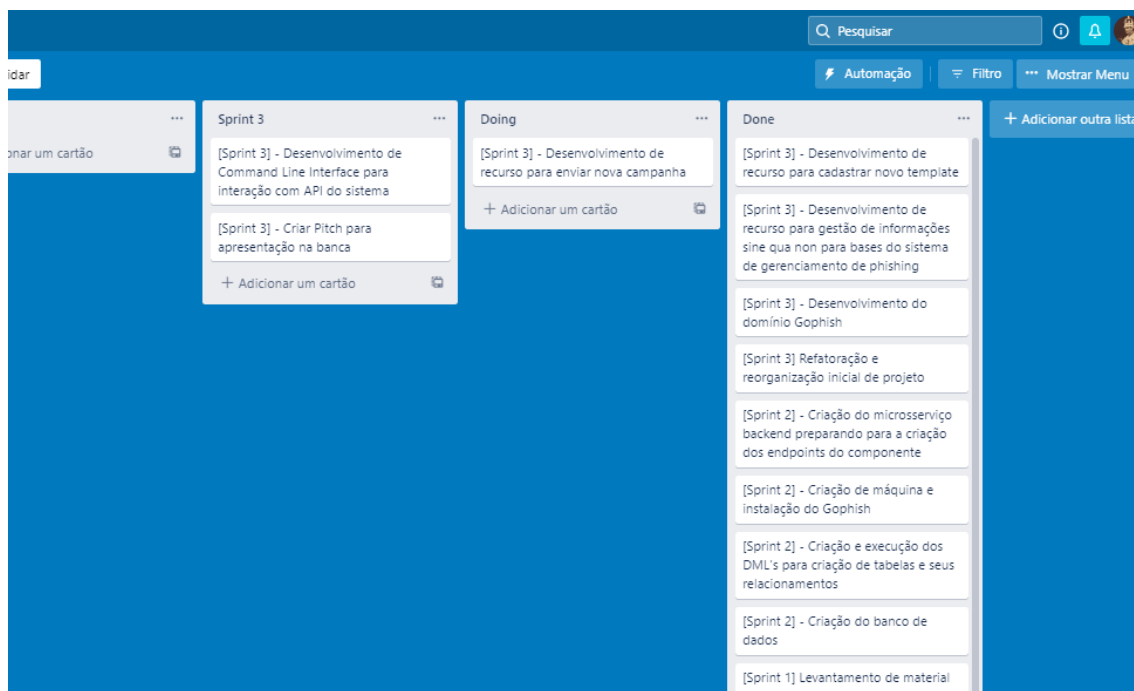
Para o item 3 da terceira sprint, a saber, “Desenvolvimento de recurso para gestão de informações *sine qua non* para bases do sistema de gerenciamento de *phishing*”, o planejamento via *Trello* encontra-se abaixo:



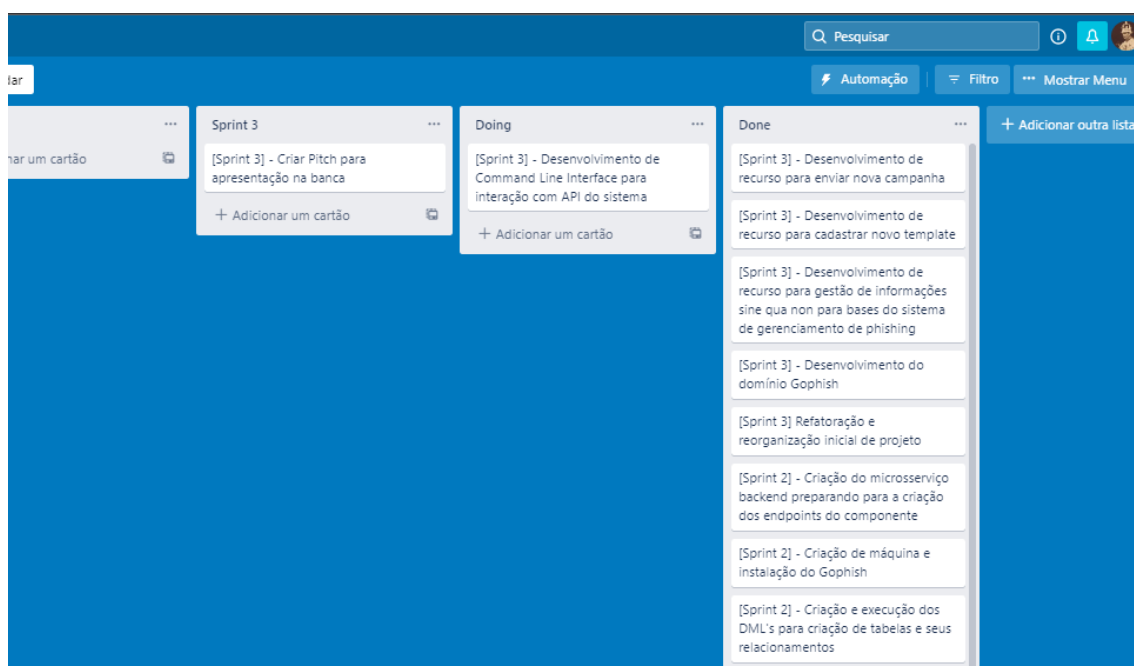
Para o item 4 da terceira sprint, a saber, “Desenvolvimento de recurso para cadastrar novo *template*”, o planejamento via *Trello* encontra-se abaixo:



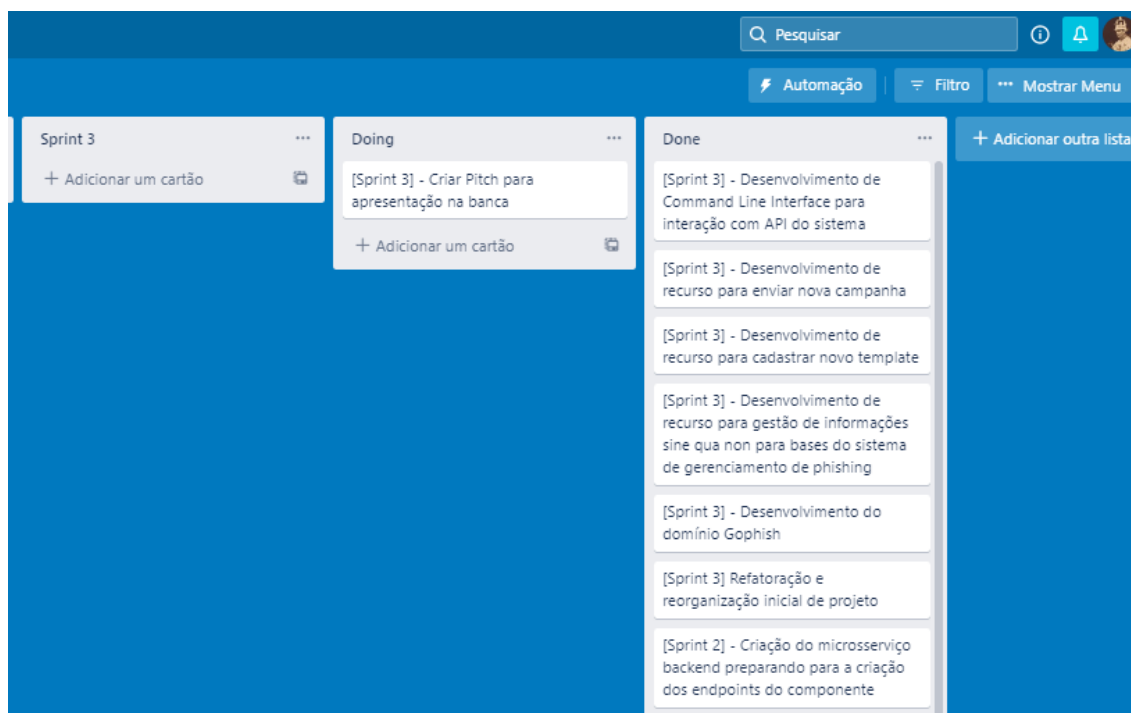
Para o item 5 da terceira sprint, a saber, “Desenvolvimento de recurso para enviar nova campanha”, o planejamento via *Trello* encontra-se abaixo:



Para o item 6 da terceira sprint, a saber, “Desenvolvimento de *Command Line Interface* para interação com *API* do sistema”, o planejamento via *Trello* encontra-se abaixo:



Para o último item da terceira sprint, de número 7, a saber, “Criar Pitch para apresentação na banca”, o planejamento via *Trello* encontra-se abaixo::



- Evidência da execução de cada requisito:

Para o primeiro item, “Refatoração e reorganização inicial de projeto”, as evidências de sua execução seguem através da descrição a seguir.

Após longa reflexão sobre a organização inicial do microserviço, pensou-se numa nova organização do mesmo. As bases estruturais permanecem, mas, pensando na evolução deste *backend*, optou-se por um padrão modular. Deste modo, pode-se evoluir o serviço *bff-phishing-campaign-app* facilmente, sem impactar o código existente com a criação de novos módulos para atender novos domínios e necessidades de negócio. Assim sendo, este microserviço possui o módulo *campaign-service* e um módulo chamado *ortogonal-advance-service* como exemplo de um crescimento ortogonal de funcionalidades sem afetar diretamente as funcionalidades existentes. A evidência da nova estrutura estará disponível na seção de ‘Evidência dos Resultados’.

Para o segundo item, “Desenvolvimento do domínio *Gophish*”, as evidências de sua execução seguem através da descrição a seguir.

Conforme demonstrado nos diagramas arquiteturais da sprint anterior, *Gophish* e sistema de gerenciamento de *phishing* compartilham da mesma base de dados. Para que os dados durante as transações permaneçam sincronizados, criou-se um módulo para lidar com a comunicação direta através do *client* em *python* disponibilizado pelo próprio *framework* (*Gophish*). Desse modo, as ações de criar grupos, enviar campanhas e criar *templates* foram codificadas para serem orquestradas junto com as transações

do sistema de gerenciamento de campanhas. A evidência do módulo estará disponível na seção de ‘Evidência dos Resultados’.

Para o terceiro item, “Desenvolvimento de recurso para gestão de informações *sine qua non* para bases do sistema de gerenciamento de *phishing*”, as evidências de sua execução seguem através da descrição a seguir.

Para que uma campanha seja enviada apropriadamente, algumas estruturas devem ser criadas no sistema, conforme modelo de dados apresentado na *sprint* anterior. Assim sendo, recursos de criação de Categoria e Subcategoria foram criados, para que os analistas de segurança possam realizar a criação e gerenciamento de engenharia social de forma livre. A evidência de execução deste requisito será apresentada na seção de ‘Evidência dos Resultados’

Para o quarto item, “Desenvolvimento de recurso para cadastrar novo *template*”, as evidências de sua execução seguem através da descrição a seguir.

Após a criação de categorias e subcategorias identificadas pelo time de *cyber security*, i.e., novas entradas além daquelas identificadas inicialmente no estudo do behaviorismo radical e pirâmide das necessidades de Maslow, faz-se necessário expor um recurso destinado para a criação de *templates*, o texto que irá ser enviado aos funcionários da organização. Seu formato é o mesmo definido pelo *Gophish*, i.e., texto formatado usando *HyperText Markup Language (HTML)*. Nesta fase existirá também uma orquestração entre a base interna utilizada pelo *Gophish* e pela base de dados do sistema de gerenciamento de *phishing*, ambos usando o mesmo motor ou sistema gerenciador de banco de dados, o *RDS MySQL*. A evidência de execução da criação do recurso de *Template* será apresentada na seção de ‘Evidência dos Resultados’

Para o quinto item, “Desenvolvimento de recurso para enviar nova campanha”, as evidências de sua execução seguem através da descrição a seguir.

Por fim, nesta primeira iteração de criação do produto, criou-se o recurso para criação e lançamento de campanhas. Este se relaciona com todos os outros desenvolvidos anteriormente, e.g. grupos, categorias, subcategorias. Este recurso, como o anterior, comunica-se tanto com a *API python* do servidor *Gophish* quanto com a base de dados do sistema de gerenciamento de *phishing* de maneira orquestrada. Além de categoria e subcategoria, o período é um dado obrigatório para o lançamento de uma campanha. Nesta primeira versão, o período deve ser definido com valor *default* do *timestamp* atual. No final do trabalho, na seção de próximos passos, propor-se-á a criação de campanhas agendadas. A evidência do recurso de lançamento de campanhas estará disponível na seção de ‘Evidência dos Resultados’.

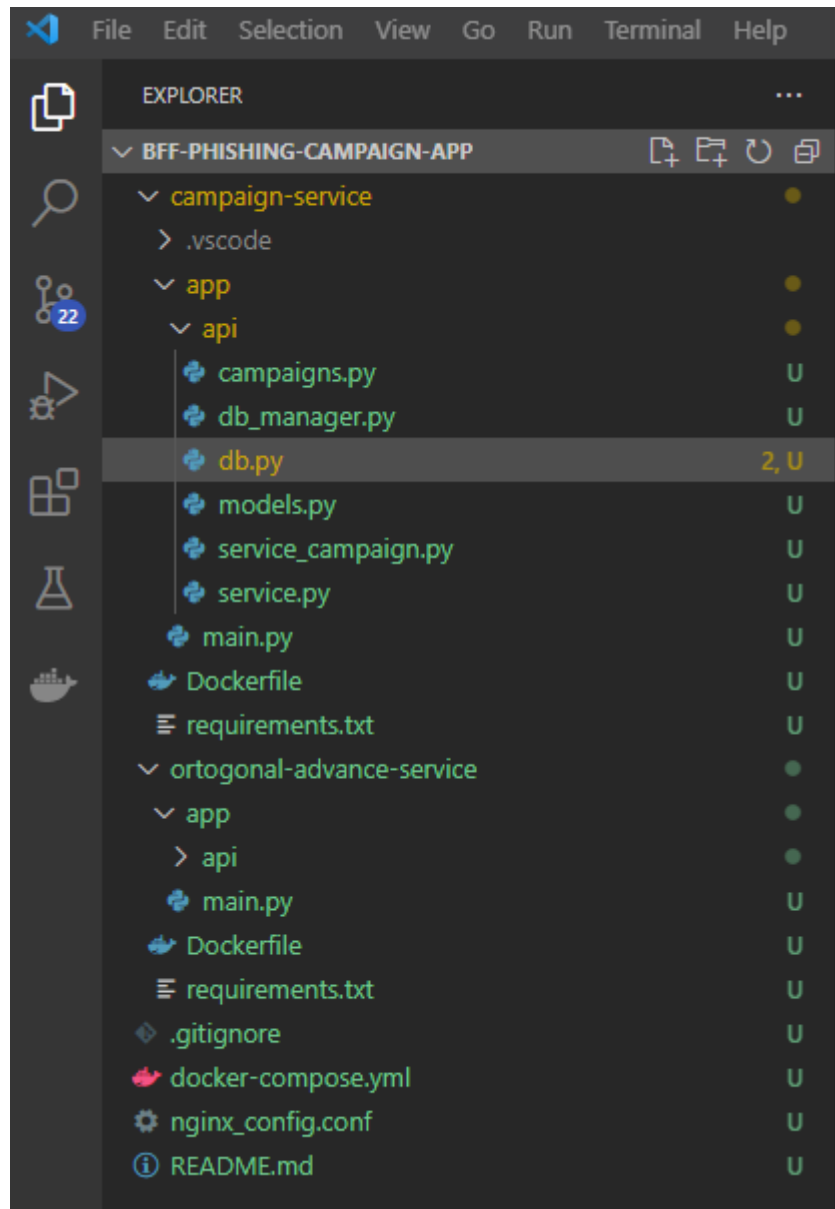
Para o sexto item, “Desenvolvimento de *Command Line Interface* para interação com *API* do sistema”, as evidências de sua execução seguem através da descrição a seguir.

Por fim, no que concerne o escopo deste trabalho, criou-se uma interface via linha de comando para que os analistas possam se comunicar com o sistema de gerenciamento e aproveitem de suas funcionalidades e orquestrações. Como dito na seção de lições aprendidas da segunda sprint, decidiu-se por ajustar o escopo, retirando o *microfrontend* da entrega, mantendo a interface de comunicação do usuário através de linha de comando. A evidência do CLI estará disponível na seção de ‘Evidência dos Resultados’

Para o último item, “Criar *Pitch* para apresentação na banca”, a criação do último artefato da última sprint, tomou-se como base o *template* fornecido pela instituição de ensino (IGTI/XPE) e também o exemplo disponibilizado pelo orientador da disciplina de Projeto Aplicado. A evidência do resultado será apresentada seção a seguir.

- Evidência dos resultados:

Para o primeiro item, “Refatoração e reorganização inicial de projeto”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.



Para o segundo item, “Desenvolvimento do domínio *Gophish*”, as evidências de sua execução seguem através do print a seguir.

```
campaign-service > app > api > service.gophish.py
1 from getopt import getopt
2 import string
3 from typing import List
4 from app.api.service.gophish import api
5 from app.api.models import GophishUser, GophishGroup
6 from gophish.models import *
7
8
9 newGroup = List(GophishGroup)
10 defaultgroups = []
11 group = Group(name='Empresa a ser testada')
12 defaultgroups.append(group)
13 def prepareUserForGroup(first_name: string, last_name: string, email: string, position: string):
14     return newGroup.append(GophishUser(first_name, last_name, email, position))
15
16 def createGroup(name: string, targets: List(GophishGroup)):
17     group = Group(name='Empresa a ser testada', targets=targets)
18     group = api.groups.post(group)
19     return
20
21 def sendCampaign(template: Template, camp: Campaign):
22     groups = defaultgroups
23     template = temp
24     smtp = SMTP(name='Empresa a ser testada')
25     url = 'http://phishing_server'
26     campaign = Campaign(
27         name=camp.name, groups=groups,
28         template=template, smtp=smtp)
29
30     campaign = api.campaigns.post(campaign)
31     print campaign.id
32
33     return
34
35 def createTemplate(template_name: string, html_string_template: string):
36     template = Template(name=template_name,
37         html=html_string_template)
38
39     template = api.templates.post(template)
40     return
41
```

Para o terceiro item, “Desenvolvimento de recurso para gestão de informações *sine qua non* para bases do sistema de gerenciamento de *phishing*”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.

```
1 from typing import List
2 from fastapi import APIRouter, HTTPException
3
4 from app.api.models import CategoryCreated, CreateCategory, SubCategoryCreated, CreateSubCategory, CategoryList, SubCategoryList
5 from app.api import db_manager
6
7 campaigns = APIRouter()
8
9
10 @campaigns.post('/category', response_model=CategoryCreated, status_code=201)
11 async def create_category(payload: CreateCategory):
12     category = await db_manager.create_category(payload)
13     return CategoryCreated('created')
14
15 @campaigns.post('/subcategory', response_model=SubCategoryCreated, status_code=201)
16 async def create_subcategory(payload: CreateSubCategory):
17     category = await db_manager.create_subcategory(payload)
18     return SubCategoryCreated('created')
19
20 @campaigns.get('/category', response_model=List[CategoryList], status_code=200)
21 async def list_categories():
22     categories = await db_manager.list_categories()
23     return categories
24
25 @campaigns.get('/subcategory', response_model=List[SubCategoryList], status_code=200)
26 async def list_subcategories():
27     subcategories = await db_manager.list_subcategories()
28     return subcategories
29
```

```
db.py 2 db.gophish.py 1 campaign.py 1, M db_manager.py 8, M X service_campaign.py
campaign-service > app > api > db_manager.py > ...
1 from app.api.models import CreateSubCategory, CreateCategory
2 from app.api.db import category, subcategory, database
3
4
5 async def create_category(payload: CreateCategory):
6     query = category.insert().values(**payload.dict())
7
8
9 async def create_subcategory(payload: CreateSubCategory):
10     query = subcategory.insert().values(**payload.dict())
11
12
13 async def list_categories():
14     query = category.select()
15     return await database.fetch_all(query=query)
16
17
18 async def list_subcategories():
19     query = subcategory.select()
20     return await subcategory.fetch_all(query=query)
21
22
```

Para o quarto item, “Desenvolvimento de recurso para cadastrar novo *template*”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.

```

campaigns.py 7, M • models.py 1, M • service_gophish.py 1, M • db_manager.py M
campaign-service > app > api > campaigns.py > ...
1 from typing import List
2 from fastapi import APIRouter, HTTPException
3
4 from app.api.models import CategoryCreated, CreateCategory, SubCategoryCreated, CreateSubCategory, CategoryList, SubCategoryList, CreateTemplate, TemplateCreated, CreateTemplateList
5 from app.api import db_manager
6 from app.api import service_gophish
7
8 campaigns = APIRouter()
9
10 @campaigns.post('/category', response_model=CategoryCreated, status_code=201)
11 async def create_category(payload: CreateCategory):
12     category = await db_manager.create_category(payload)
13     return CategoryCreated('created')
14
15 @campaigns.post('/subcategory', response_model=SubCategoryCreated, status_code=201)
16 async def create_subcategory(payload: CreateSubCategory):
17     category = await db_manager.create_subcategory(payload)
18     return SubCategoryCreated('created')
19
20 @campaigns.get('/category', response_model=List[CategoryList], status_code=200)
21 async def list_categories():
22     categories = await db_manager.list_categories()
23     return categories
24
25 @campaigns.get('/subcategory', response_model=List[SubCategoryList], status_code=200)
26 async def list_subcategories():
27     subcategories = await db_manager.list_subcategories()
28     return subcategories
29
30 @campaigns.post('/template', response_model=TemplateCreated, status_code=201)
31 async def create_template(payload: CreateTemplate):
32     template = await db_manager.create_template(payload)
33     service_gophish.create_template(payload.template_name, payload.template_string)
34     return TemplateCreated('created')
35
36 @campaigns.get('/template', response_model=List[TemplateList], status_code=200)
37 async def list_template():
38     templates = await db_manager.list_template()
39     return templates
40
41
42
43

```

```

db.py 2 • campaigns.py 7, M • models.py 1, M • service_gophish.py 1, M • db_manager.py M X
campaign-service > app > api > db_manager.py > create_template
1 from app.api.models import CreateSubCategory, CreateCategory, CreateTemplate
2 from app.api.db import category, subcategory, template, campaign, database
3
4
5 async def create_category(payload: CreateCategory):
6     query = category.insert().values(**payload.dict())
7
8
9 async def create_subcategory(payload: CreateSubCategory):
10    query = subcategory.insert().values(**payload.dict())
11
12
13 async def list_categories():
14    query = category.select()
15    return await database.fetch_all(query=query)
16
17
18 async def list_subcategories():
19    query = subcategory.select()
20    return await subcategory.fetch_all(query=query)
21
22
23 async def create_template(payload: CreateTemplate):
24    query = template.insert().values(**payload.dict())
25
26
27 async def list_template():
28    query = template.select()
29    return await database.fetch_all(query=query)
30
31

```

```

campaign-service > app > api > service.gophish.py > createTemplate
10 def createTemplate(template_name: str, targets: List[str]):
11     group = Group(name="Empresa a ser testada", targets=targets)
12     group = api.groups.post(group)
13     return group
14
15 def sendCampaign(template: Template, campaign: Campaign):
16     groups = defaultgroups
17     template = template
18     smtp = SMTP(name="Empresa a ser testada")
19     url = "http://phishing_server"
20     campaign = Campaign(
21         name=campaign.name, groups=groups,
22         template=template, smtp=smtp)
23     campaign = api.campaigns.post(campaign)
24     return campaign
25
26 def createTemplate(template_name: str, html_string_template: str):
27     template = Template(name=template_name,
28                         html=html_string_template)
29     template = api.templates.post(template)
30     return template
31
32 def sendProfile():
33     smtp = SMTP(name="Test SMTP")
34     smtp.host = "localhost:25"
35     smtp.from_address = "John Doe <johndoe@example.com>"
36     smtp.interface_type = "SMTP"
37     smtp.ignore_cert_errors = True
38     smtp = api.smtp.post(smtp)
39     return smtp
40
41
42
43
44
45
46
47
48
49
50

```

Para o quinto item, “Desenvolvimento de recurso para enviar nova campanha”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.

```

campaign-service > app > api > campaigns.py > list_template
11 @campaigns.post('/category', response_model=CategoryCreated, status_code=201)
12 async def create_category(payload: CreateCategory):
13     category = await db_manager.create_category(payload)
14     return categoryCreated('created')
15
16 @campaigns.post('/subcategory', response_model=SubCategoryCreated, status_code=201)
17 async def create_subcategory(payload: CreateSubCategory):
18     category = await db_manager.create_subcategory(payload)
19     return subcategoryCreated('created')
20
21 @campaigns.get('/category', response_model=List[CategoryList], status_code=200)
22 async def list_categories():
23     categories = await db_manager.list_categories()
24     return categories
25
26 @campaigns.get('/subcategory', response_model=List[SubCategoryList], status_code=200)
27 async def list_subcategories():
28     subcategories = await db_manager.list_subcategories()
29     return subcategories
30
31 @campaigns.post('/template', response_model=TemplateCreated, status_code=201)
32 async def create_template(payload: CreateTemplate):
33     template = await db_manager.create_template(payload)
34     service.gophish.createTemplate(payload.template_name, payload.template_string)
35     return templateCreated('created')
36
37 @campaigns.get('/template', response_model=List[TemplateList], status_code=200)
38 async def list_template():
39     subcategories = await db_manager.list_template()
40     return subcategories
41
42
43
44
45 @campaigns.post('/', response_model=CampaignCreated, status_code=201)
46 async def create_campaign(payload: CreateCampaign):
47     subcategories = await db_manager.create_campaign()
48     service.gophish.sendCampaign(payload.campaign_name, payload.templateid)
49     return campaignCreated('created')
50
51
52
53 @campaigns.get('/', response_model=List[CampaignList], status_code=200)
54 async def list_campaign():
55     campaigns = await db_manager.list_campaign()
56     return campaigns
57

```

```

campaign-service > app > api > db_manager.py > create_campaign
1 from app.api.models import createSubcategory, createCategory, createTemplate
2 from app.api.db import category, subcategory, template, campaign, database
3
4
5 async def create_category(payload: createCategory):
6     query = category.insert().values(**payload.dict())
7
8
9 async def create_subcategory(payload: createSubcategory):
10    query = subcategory.insert().values(**payload.dict())
11
12
13 async def list_categories():
14    query = category.select()
15    return await database.fetch_all(query=query)
16
17
18 async def list_subcategories():
19    query = category.select()
20    return await subcategory.fetch_all(query=query)
21
22
23 async def create_template(payload: createTemplate):
24    query = template.insert().values(**payload.dict())
25
26
27 async def list_template():
28    query = template.select()
29    return await database.fetch_all(query=query)
30
31
32 async def create_campaign(payload: createCampaign):
33    query = campaign.insert().values(**payload.dict())
34
35
36 async def list_campaign():
37    query = campaign.select()
38    return await database.fetch_all(query=query)

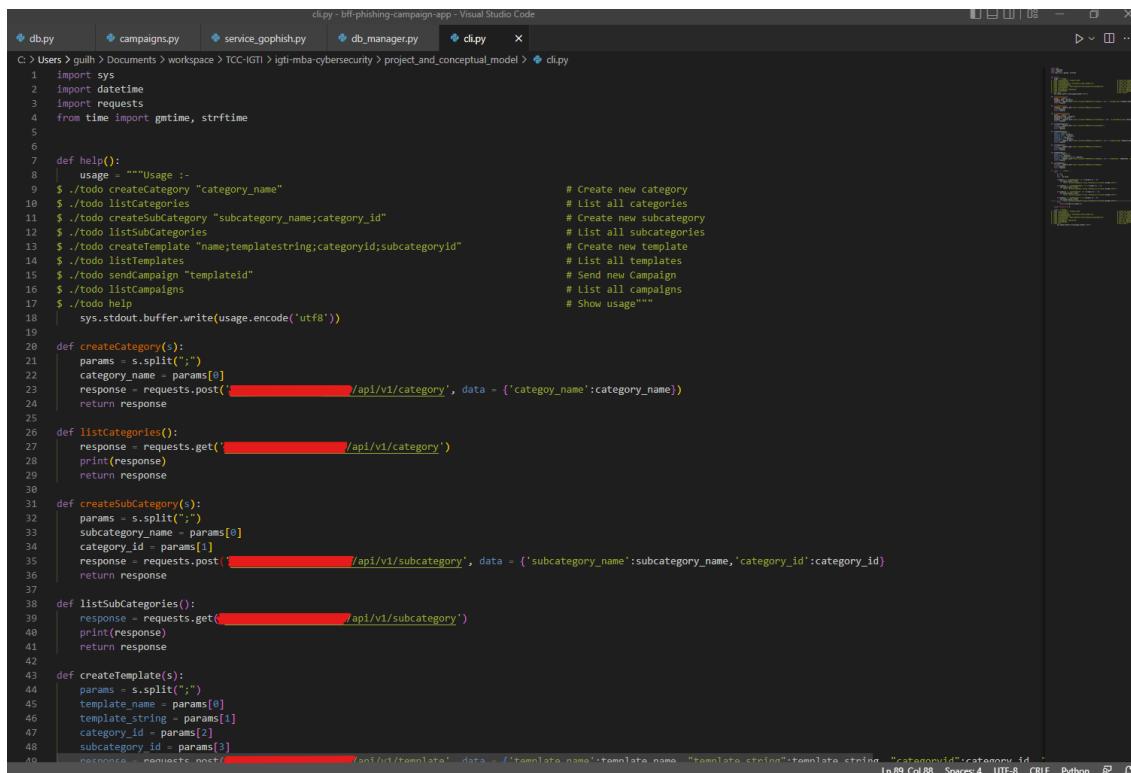
```

```

campaign-service > app > api > service_gophish.py > createTemplate
1 from getopts import getopts
2 import string
3 from typing import List
4 from app.api.service_gophish import api
5 from app.api.models import GophishUser, GophishGroup
6 from gophish.models import *
7
8
9 newGroup = List(GophishGroup)
10 defaultGroups = []
11 group = Group(name = 'Empresa a ser testada')
12 defaultGroups.append(group)
13 def prepareUserForGroup(first_name : string, last_name : string, email: string, position: string):
14     return newGroup.append(GophishUser(first_name, last_name, email, position))
15
16
17 def createGroup(name: string, targets: List(GophishGroup)):
18     group = Group(name='Empresa a ser testada', targets=targets)
19     group = api.groups.post(group)
20     return
21
22
23 def sendCampaign(temp: Template, camp: Campaign):
24     groups = defaultGroups
25     template = temp
26     smtp = SMTP(name='Empresa a ser testada')
27     url = 'http://phishing server'
28     campaign = Campaign(
29         name=camp.name, groups=groups,
30         template=temp, smtp=smtp)
31
32     campaign = api.campaigns.post(campaign)
33
34     return

```

Para o sexto item, “Desenvolvimento de *Command Line Interface* para interação com *API* do sistema”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.



```

1 import sys
2 import datetime
3 import requests
4 from time import gmtime, strftime
5
6
7 def help():
8     usage = """Usage :-
9     $ ./todo createCategory "category_name"           # Create new category
10    $ ./todo listCategories                             # List all categories
11    $ ./todo createSubCategory "subcategory_name;category_id" # Create new subcategory
12    $ ./todo listSubCategories                         # List all subcategories
13    $ ./todo createTemplate "name;templatestring;categoryid;subcategoryid" # Create new template
14    $ ./todo listTemplates                             # List all templates
15    $ ./todo sendCampaign "templateid"                # Send new campaign
16    $ ./todo listCampaigns                             # List all campaigns
17    $ ./todo help                                     # Show usage"""
18     sys.stdout.buffer.write(usage.encode('utf8'))
19
20 def createCategory(s):
21     params = s.split(";")
22     category_name = params[0]
23     response = requests.post('http://localhost:8080/api/v1/category', data = {'category_name':category_name})
24     return response
25
26 def listCategories():
27     response = requests.get('http://localhost:8080/api/v1/category')
28     print(response)
29     return response
30
31 def createSubCategory(s):
32     params = s.split(";")
33     subcategory_name = params[0]
34     category_id = params[1]
35     response = requests.post('http://localhost:8080/api/v1/subcategory', data = {'subcategory_name':subcategory_name,'category_id':category_id})
36     return response
37
38 def listSubCategories():
39     response = requests.get('http://localhost:8080/api/v1/subcategory')
40     print(response)
41     return response
42
43 def createTemplate(s):
44     params = s.split(";")
45     template_name = params[0]
46     template_string = params[1]
47     category_id = params[2]
48     subcategory_id = params[3]
49     response = requests.post('http://localhost:8080/api/v1/template', data = {'template_name':template_name,'template_string':template_string,'category_id':category_id,'subcategory_id':subcategory_id})
50     return response

```

Para o último item, “Criar *Pitch* para apresentação na banca”, as evidências de sua execução seguem através dos prints que serão disponibilizados a seguir.



2.3.2 Experiências vivenciadas

Como lições ou aprendidas ou experiências vivenciadas durante a realização desta terceira *sprint*, podemos destacar um principal ponto. Conforme previsto, a construção do código do microsserviço e do *CLI* foi realizado, atendendo às necessidades principais do sistema. Porém, para que o sistema continua evoluindo nos próximos passos, mais funcionalidades devem ser acrescentadas para atender as novas necessidades de *frontend*, *landingpages* e agendamento de campanhas, que serão na última seção deste trabalho.

3. Considerações Finais

3.1 Resultados

Durante a realização deste projeto, percebemos os resultados a seguir: (a) Dar substância à forma é de importância fundamental para qualquer proposta. Propôs-se adotar uma base teórica ao revisitar o behaviorismo clássico e radical para a criação de uma política e arquitetar o sistema baseado nesta; (b) Como este é um projeto *open source*, qualquer organização pode se beneficiar da ideia e das estruturas criadas até o momento, além de poder trocar elementos *Cloud Native*, como o *RDS Mysql*, por bancos de dados *onPremise* ou até mesmo outros motores de fornecedores de mercado; (c) por fim, destaca-se a possibilidade de evoluir o projeto ortogonalmente, i.e., novas funcionalidades podem ser adicionadas através de módulos bem definidos, como demonstrado na *sprint* de número 3. Como ponto negativo, não há como não apontar o tempo. Infelizmente a curta duração para pensar numa ideia inovadora, organizar suas etapas e executar as tarefas definidas faz com que qualquer projeto dessa natureza deixe a desejar quanto a sua completude em relação ao escopo definido inicialmente.

3.2 Contribuições

Com relação às contribuições do projeto em relação à instituição de ensino e sociedade de modo geral, destaca-se os pontos levantados na seção de resultados, como o baixo custo de implementação do sistema de gerenciamento de *phishing*, um arcabouço mínimo para se pensar em campanhas eficientes e eficazes na organização visando o objetivo final deste trabalho, que é a conscientização dos colaboradores, um treino constante e organizado para que a organização fique menos vulnerável a ataques de engenharia social. Com isto, podemos dizer que a reputação da organização crescerá ao longo do tempo e seus sistemas e dados de clientes também terão um grau a mais de segurança após a mitigação de ataques de natureza humana como é o *phishing*.

3.3 Próximos passos

Como próximos passos, identificamos os seguintes pontos de melhoria para o projeto:

- a) **Front-end.** Para que o sistema seja melhor e mais rapidamente utilizado, o desenvolvimento de uma interface gráfica é indispensável. Este componente estava pensando desde a concepção deste projeto, mas por motivo de curto tempo, apenas a interface de linha de comando foi desenvolvida. Deste modo, como próximo passo, deve-se criar o componente Angular que fará comunicação com o microserviço de gerenciamento de campanhas.
- b) **Agendamento de Campanhas.** Esta é uma ideia que foi levantada no final da segunda sprint. Notou-se que a estrutura de dados criada para o sistema, com base no modelo teórico e na integração com o sistema *Gophish*, comporta o agendamento de campanhas através da entidade período composta na entidade de campanha. Algumas estruturas devem ser atualizadas, como a criação de um status para controle interno (ENVIADO, PROCESSANDO, AGENDADO), mas o impacto não seria grande. Evidentemente, um *scheduler*, aos moldes de um *cron job*, deve ser criado no microserviço responsável pelo sistema; estas tarefas devem ser calculadas e desenvolvidas para a implementação desta *feature*.
- c) **Landing Pages.** Por fim, para que o sistema fique mais completo, pode-se passar a utilizar *landing pages* para capturar cliques dos colaboradores. O *framework Gophish* já disponibiliza esta *feature*, e a implementação desta nova funcionalidade deve ser calculada. A princípio, os componentes microserviço, *front-end* e *CLI* devem sofrer alteração, i.e., escrita de código novo, mas o impacto não deve ser grande, já que todas as partes do sistema estão modularizadas e este pode crescer ortogonalmente.