Neste capítulo será retratado o desenvolvimento do sistema focado nos problemas e erros encontrados pelos desenvolvedores e os usuários que utilizaram o aplicativo Linkest.

A partir de agora vamos descrever os erros e problemas encontrados e as suas respectivas soluções:

* Criação de backup (cópia de segurança) do banco de dados na máquina virtual

Como o aplicativo foi implementado no modelo B2B, não pode existir a perda de dados dos usuários que utilizaram o aplicativo. No intuito de resolver este problema foi desenvolvido um script a nível de sistema operacional que diariamente cria um backup do banco de dados e o armazena localmente. Futuramente o armazenamento deste backup será na nuvem.

* Adição de segurança a páginas da API

Um problema encontrado atualmente é a questão de segurança digital, também foi encontrado este problema em nossa aplicação. Sabemos que quando é criado um API, os endpoints vem com default no protocolo HTTP (Hypertext Transfer Protocol), que por sua vez são links sem segurança e com falta de certificados, trazendo insegurança aos usuários e falta de prestígio a aplicação. A solução encontrada para isto foi a adição de um componente chamado Https-portal em um novo container, que por sua vez adiciona certificado e o protocolo HTTPS (Hyper Text Transfer Protocol Secure) aos endpoints criados pela API.

* Atualização e melhoria de código sobre o mongoengine

Na aplicação foi encontrado alguns erros e redundâncias nas linhas de código direcionada sobre a utilização do mongo, os maiores erros encontrados foram pesquisas desnecessárias, tratamento indevido de dados e abertura de conexões com o banco de dados igualmente desnecessários.No início da criação da API foi adicionado o mongoengine que resolve vários desses problemas, porém ele não estava sendo usado pela aplicação, portanto foi adicionado a utilização correta do mongoengine no sistema seguindo sua documentação, modificando as linhas de código que continham a utilização direta de mongo por linhas que contém a utilização de mongoengine.

* Adição de relatório na API

A pedido do usuário, foi adicionado pesquisas sobre datas, quantidades e tipos de links adicionados no sistema, criando relatórios sobre as informações que estavam sendo registradas no Linkest. A implementação deste requisito foi adição de dados como datas e tipos de links ao modelo, como a implementação de funções que manipulam estes dados e retornam relatórios ao usuário.

* Adição de papéis

O usuário nos requereu que fosse adicionado níveis diferentes de usuários na aplicação e que cada nível tivesse um grau de responsabilidade no sistema, para isso adicionamos no modelo um dado informando o nível de usuário de cada cadastrado e seus limites em cada função do sistema.

* Correção de funções para determinados papéis

Para que os usuários indevidos não acessassem alguma funções proibidas em relação aos seus respectivos papéis, foi adicionado vários verificadores e limites na API que avaliavam o papel do usuário e permitia ou não a execução das funções. Na interface, era verificado o papel do usuário e só exibia o que era permitido ao próprio.

* Adição do Sendgrid (empresa que fornece serviço de entrega de e-mail baseado em nuvem que auxilia as empresas na entrega de e-mail)

O cliente nos solicitou que cada cadastro de um novo usuário mandasse um email a si próprio, informando sobre o sucesso do registro e aguardasse a confirmação do seu respectivo administrador. E um outro email ao administrador informando sobre a inclusão de um novo usuário e pedindo a confirmação no sistema Linkest. Para implementação disto foi adicionado a biblioteca Sendgrid a API, sendo utilizada na função de criar um novo usuário o envio de emails, seguindo um template criado no site do Sendgrid.

* Confirmar cadastros de usuários por administradores

Foi adicionado na API a função de confirmar cadastro de novos usuários pelos administradores, permitindo o livre acesso aos mesmos na utilização do Linkest.

* Adição do campo empresa no login do plugin

Com a presença de várias instâncias de banco de dados instituídos a cada empresa cliente do Linkest, foi encontrado a dificuldade de atribuir qual a empresa do usuário quando fazia o login no sistema. Para resolver isto foi adicionado um novo campo no login chamada empresa que o usuário informava a mesma que fazia parte, para não sobrecarregar o sistema na busca desnecessária da empresa pertencente ao usuário.

* Resumo e diminuição dos componentes do plugin

Por falta de experiência e de entendimento da linguagem Javascript e do React, foram criados componentes desnecessários e erros de referência em seus respectivos contextos. Com o passar do tempo foi adquirido mais experiência sobre esta área e foi feito uma revisão e refatoração no código da interface do plugin.

* Mudança de captura de imagem da página no adicionar link

Na adição de novo link era adicionado uma imagem referente a página cadastrada, que por muitas vezes pegava a primeira imagem encontrada no código XML da página e que não retratava o conteúdo corretamente presentes na página. A resolução deste problema foi tirar um print da página em questão com as funcionalidades presentes no navegador Google Chrome e inserir a print na nuvem usando um serviço do mecanismo Cloudinary (empresa que fornece uma solução de gerenciamento de imagem e vídeo baseada em nuvem) e retornando o endpoint da imagem para ser adicionada como o dado referente a imagem do link no sistema Linkest.

* Adição do Dandelion API (API de análise de texto semântico) na criação de tags automáticas no adicionar link

Para a criação de tags não se tornar uma ação repetitiva e cansativa para o usuário na hora de adicionar link, foi adicionado uma função que adiciona tags de forma automática referindo ao contexto das informações presentes no link que será adicionado, utilizando a biblioteca do Dandelion, que lia toda a página e retornava os assuntos relevantes ao texto. O usuário podia usar esse mecanismo ou adicionar tags de forma manual.

* Adição de interfaces intuitivas nos filtros e busca de links no plugin

No desejo de tornar mais amigável e intuitiva a busca e a filtragens de links no plugin, foi necessária a adição da biblioteca chamada Autosuggest, que auto completava o que o usuário escrevia na caixa de texto da busca e sugestionar outras opções de prováveis conteúdos para a busca com os dados presente no Linkest. Para o funcionamento total deste item, foi necessária a adição de uma função na API que era responsável por buscar e listar possíveis conteúdos que o usuário poderia pesquisar.

* Adição de componentes para informar relatórios no plugin

A pedido do cliente, foi adicionado um novo componente que permitia ao usuário a visualização de relatórios sobre os dados adicionado no Linkest. E para tornar a interface mais amigável ao usuário, foi utilizado um conjunto de bibliotecas visuais e funcionais na interface, que permitia a visualização dos dados em infográficos.

* Correção da visibilidade de links

Por causa de erros de referências presentes nas listagens de links, muitas vezes não era listados os links de forma correta ou eram listados links que não poderiam ser visualizados, para resolver este problema, foi necessária uma revisão e refatoração de código da listagem de links na interface e uma refatoração no código na listagem de links na API.

* Adição de Mixpainel (empresa de serviços de análise de negócios) no plugin

Para podermos avaliar e estudar as ações do usuário em nosso sistema, foi adicionado na interface um serviço do mecanismo Mixpainel, que registrava as ações dos usuário e criava relatórios sobre estas ações no site da própria biblioteca.

* Problema no upload do plugin para Google Chrome Store

Próximo ao término da entrega do sistema, o cliente pediu o upload da interface no Google Chrome Store, o upload foi realizado, porém pela falta de diretrizes e configurações obrigatórias pedidas pelo Chrome Store na interface, não foi aceito. Teve que ser realizado uma adição de configurações e refatorações de conteúdos básicos da interface, para que houvesse o aval do plugin pelo Google Chrome Store.

* Adição de mais mensagens no chatbot para informar empresa ao usuário

Pela mesma dificuldade de saber a empresa pertencente do usuário, foi adicionado algumas mensagens perguntando qual a empresa que o usuário trabalhava, assim ele informava a empresa antes de realizar o login e o sistema utilizava a instância do banco de dados correto para realizar as funções.

* Utilização de Flask (microframework para Python) para login no chatbot

Ao utilizar o chat da página do chatbot no Facebook, o usuário se relacionava com o chatbot do sistema. Mas não era possível realizar login sistema pelo chat do Facebook, então foi desenvolvida uma página HTML (Hypertext Markup Language) como template para a tela de login e adicionada no código da API com a utilização do Flask, possibilitando o login do usuário que utilizava o chatbot. A página em Flask abria no navegador padrão do dispositivo e o usuário poderia realizar seu login por lá. Depois do login, o usuário tinha a permissão de utilizar as funções do chatbot do sistema na chat do Facebook.

* Trabalho com limite de links no chatbot

As mensagens do chatbot no Facebook tem limites de tamanho e com isso não eram possíveis exibir dados extensos e vários links de uma só vez, portanto foi necessário impor limites às informações apresentadas, que fossem dentro do padrão do Facebook, e foi realizada uma paginação dos links na API que fossem listados pelo chatbot no Facebook.

* Problema de troca de instâncias do banco de dados a nível dos containers do Docker ou a API

Na adição de várias empresas no sistema, foi ficando cada vez mais difícil trabalhar com a troca das instâncias do banco de dados de forma eficiente. Foi realizado uma pesquisa e um estudo sobre Docker e Eve, porém não foi possível encontrar, no tempo disponível, uma resolução ao problema.