HPE AUTOMOTORES DO BRASIL LTDA.

PROJETO NO ESTÁGIO – GUSTAVO BORGES PERES DA SILVA

PROJETO NO ESTÁGIO

Título: relatório sobre Automação de Laudos Técnicos.	Nº do relatório: 1.0
Tipo de relatório: relatório de simples	Início: 27/12/2022
Área: ServiceDesk – TI (suporte técnico)	Fim: 21/12/2023

Autor(es): Gustavo Borges Peres da Silva (matrícula 202109747)

gustavo.silva@discente.ufcat.edu.br (acadêmico); ggustavo.borges13@gmail.com (pessoal)

Universidade Federal de Catalão - UFCAT

UNID. ACAD. ESP/BIOTECNOLOGIA – CATALÃO

Curso de Ciências da Computação – *campus* Catalão

Resumo:

Este projeto é uma ferramenta de automação criada para simplificar e acelerar a manipulação de dados de planilhas especificas, eliminando o uso manual do Excel e Word e integrando uma interface gráfica que permite ao usuário visualizar e editar dados diretamente. A aplicação utiliza várias APIs e bibliotecas Java, como Apache POI e Documents4j, para operações com Excel, Word e PDF, e oferece funcionalidades avançadas, como exportação automatizada de dados para documentos PDF e Word e sua visualização integrada. Desenvolvido para otimizar o fluxo de trabalho da equipe de Service Desk, o projeto foi projetado no Eclipse com Maven para garantir a compatibilidade entre diferentes ambientes e manter as dependências atualizadas. O usuário pode selecionar e editar planilhas, e após as alterações, a ferramenta gera um arquivo PDF ou Word, armazena-o no servidor e exibe o documento final.

Palavras-chaves: Automação de Laudos Técnicos, exportação para PDF e Word, manipulação de documentos, integração com Apache POI, interface gráfica para edição de planilhas, eficiência em suporte técnico e otimização de fluxo de trabalho com Java e Maven.

Observações/Notas: https://github.com/GustavoBorges13/Conversor_XLSX-PDF.

Sumário

ROTEIRO DA AULA PRÁTICA

Precisamos aprimorar nosso serviço de elaboração de laudos técnicos para os colaboradores, pois todo o processo atual é realizado manualmente. Isso envolve acessar uma planilha do Excel e preenchê-la com dados coletados do colaborador e da máquina, como equipamento utilizado, etiquetas de ativos, nome da máquina, componentes, entre outros. Geralmente, esse preenchimento leva de 5 a 10 minutos, e na central de atendimento, onde a demanda é alta, é comum interromper o que estamos fazendo para atender os colaboradores, o que muitas vezes resulta em esquecimentos ou erros. Além disso, a planilha é extensa, tornando fácil perder o foco e comprometer a qualidade do laudo.

Após preencher a planilha, precisamos copiar as informações para um arquivo Word, que serve como modelo do laudo, para gerar a documentação profissional em PDF, que é então enviada ao setor de compras para a aquisição dos equipamentos. Este processo de transferência de dados também é manual e exige diversas operações de copiar e colar, tornando-se extremamente cansativo ao ser repetido várias vezes. Além disso, é necessário gerar o PDF após a conclusão do documento Word, e esse processo gasta aproximadamente 5 minutos, e ao processo todo totaliza-se cerca de 10-15 minutos. Assim, o desafio é encontrar uma solução que automatize essas tarefas repetitivas, buscando eficácia e eficiência em todo o fluxo de trabalho.

LISTA DE FIGURAS

LISTA DE TABELAS

No table of contents entries found.

LISTA DE ALGORITMOS

1. INTRODUÇÃO

Este projeto foi desenvolvido durante meu estágio com o objetivo de automatizar a transferência e manipulação de dados de planilhas do Excel, otimizando a criação de documentos em PDF e Word. A ferramenta visa oferecer uma solução eficaz para equipes de suporte técnico, proporcionando agilidade e praticidade no processamento de dados. Com uma interface gráfica intuitiva, os usuários podem visualizar e editar informações diretamente, eliminando a necessidade de manipulação manual.

Ao integrar bibliotecas Java como apache POI e Documents4j, a aplicação não apenas simplifica a geração de relatórios, mas também garante o armazenamento seguro dos arquivos produzidos. Durante meu tempo como estagiário, consegui reduzir o tempo necessário para gerar um único PDF em aproximadamente 10 a 15 minutos, demonstrando como a automação pode transformar operações complexas em processos mais simples e eficientes. Agradeço aos meus colegas de trabalho pelo apoio durante esse projeto, que representou uma valiosa oportunidade de aprendizado e desenvolvimento profissional.

1.1 OBJETIVO DO PROJETO

Objetivo Geral:

- Desenvolver uma solução que automatiza a geração de laudos técnicos, otimizando o fluxo de trabalho ao converter dados de planilhas Excel em documentos Word e PDF de maneira rápida e eficiente. O projeto busca melhorar a experiência dos colaboradores ao reduzir o tempo gasto em tarefas manuais.

Objetivos Específicos:

- **Automatizar a geração de laudos**: criar uma ferramenta que facilite a transferência de dados de planilhas Excel para documentos formatados, eliminando o processo manual de copiar e colar.
- **Integrar com bibliotecas Java**: explorar e utilizar várias APIs e bibliotecas, como apache POI para manipulação de planilhas e Documents4j para conversão de documentos, assegurando a eficácia e a robustez da aplicação.
- **Melhorar a eficiência**: reduzir o tempo necessário para a elaboração de laudos técnicos, minimizando erros e aumentando a produtividade dos colaboradores ao realizar a documentação.
- **Documentar o processo**: registrar todas as etapas do desenvolvimento da ferramenta, incluindo as técnicas utilizadas e os resultados obtidos, para garantir a clareza e a compreensibilidade do projeto.
- **Aplicar em cenários práticos**: demonstrar a eficácia da ferramenta em situações do dia a dia, destacando como a automação pode otimizar o suporte técnico e a produção de documentos oficiais.

2. FUNCIONALIDADES

Neste tópico, apresento brevemente algumas das principais funcionalidades do projeto. Para uma descrição mais detalhada e passo a passo com figuras, consulte o tópico 5 (Fluxo de Execução e Inicialização do Programa).

2.1 INTERFACE GRÁFICA

A interface gráfica permite a exibição de planilhas por meio de uma JTable, proporcionando uma experiência de uso intuitiva e semelhante ao Excel, sem a necessidade de abrir o software original. Os usuários podem abrir, fechar e atualizar a planilha (semelhante à função F5).

Além disso, tem 5 botões de interação com a planilha, dentre eles:

- Editar: permite editar a linha selecionada (uma única linha) ou, alternativamente, dar dois cliques na linha desejada;
- Adicionar: insere uma nova linha após o último elemento existente na planilha;
- Remover: remove a linha selecionada (uma única linha);
- Salvar alterações: se alguma linha foi editada, adicionada ou removida, essa opção ficará habilitada para salvar (sobrescrever) a planilha atual;
- Gerar arquivo PDF: gera um arquivo Word e PDF compostos a partir da linha ou do conjunto de linhas selecionadas na JTable, referentes ao mesmo laudo.

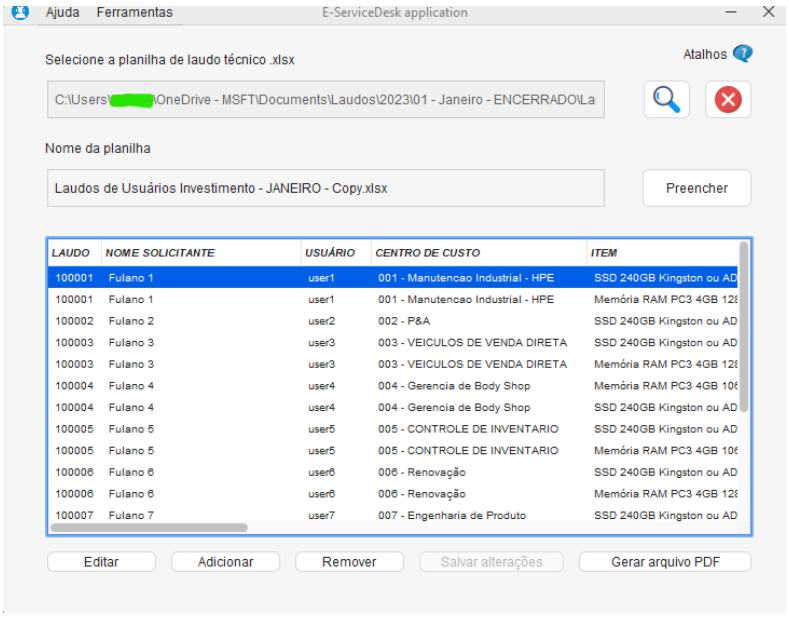

Figura 1: demonstração interface gráfica.

2.2 AUTOMAÇÃO E EXPORTAÇÃO

Esta funcionalidade gera automaticamente documentos Word e PDF após a edição dos dados. Os usuários podem visualizar o PDF diretamente na interface do aplicativo e salvar os arquivos gerados no servidor.

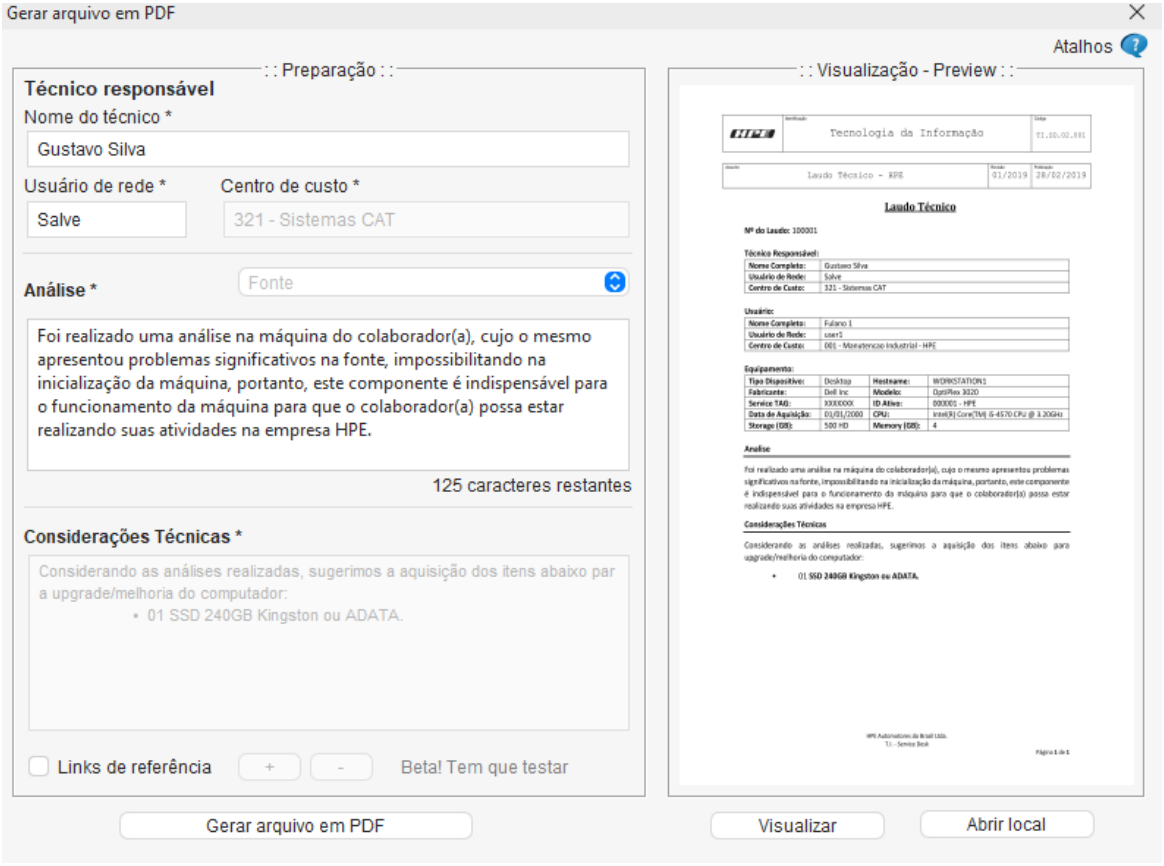

Figura 2: demonstração da automatização e exportação.

2.3 CONFIGURAÇÕES DE AMBIENTE

Esta funcionalidade oferece a possibilidade de configurar o JDK e JRE compatíveis, além de permitir a configuração do build com Maven, assegurando a consistência entre diferentes ambientes de desenvolvimento e produção.

2.4 EDIÇÃO DE CÉLULAS

Os usuários podem selecionar planilhas, editar células e utilizar ferramentas adicionais para a edição e geração de documentos de forma eficiente. O programa é capaz de identificar se o modelo de laudo utilizado é compatível e se a planilha atende aos requisitos necessários. Isso inclui verificar se todas as colunas obrigatórias estão presentes e se os campos das células estão

no formato esperado (por exemplo, se uma célula deve conter uma string, um número ou uma data).

Além disso, o programa conta com diversas exceções implementadas para auxiliar os usuários durante o processo, evitando que a aplicação seja encerrada abruptamente em caso de erros. Essas exceções oferecem feedback claro sobre o que pode ter dado errado, permitindo que os usuários façam as correções necessárias de maneira prática e eficiente.

3 TECNOLOGIAS E DEPENDÊNCIAS UTILIZADAS

- Java (JDK 20+)
- Maven para gerenciamento de dependências e automação de build.
- Apache POI para manipulação de arquivos Excel.
- Apache PDFBox para criação e manipulação de PDFs, que inclui a funcionalidade de renderização de PDFs na interface gráfica.
- Documents4j para conversão de documentos Word para PDF.
- PDF Renderer é parte do Apache PDFBox, utilizada para exibir PDFs na interface gráfica.
- FlatMacDarkLaf / FlatMacLightLaf são temas que simulam a aparência de aplicações nativas do macOS em modos escuro ou claro, ideais para diferentes condições de iluminação.

3.1 RELATÓRIO DE DEPENDENCIAS (SBOM)

i. org.apache.poi: API - O Apache POI é uma API para criar e manipular documentos do Microsoft Office, como arquivos do Word e Excel. Ele permite que você trabalhe com esses formatos de arquivo em Java.
ii. org.apache.poi-ooxml: API - Uma extensão do Apache POI que lida especificamente com formatos de arquivo Office Open XML, como arquivos .xlsx (Excel).
iii. org.apache.poi.ooxml-schemas: Utilitário - Este pacote fornece as bibliotecas de esquema XML necessárias para trabalhar com arquivos Office Open XML. É uma dependência de apoio para o Apache POI.
iv. org.apache.pdfbox: API - O Apache PDFBox é uma biblioteca Java que permite criar e manipular arquivos PDF. É uma API para trabalhar com PDFs em Java. Agradeço muito essa biblioteca pois é a melhor que eu encontrei e gratuita.
v. org.swinglabs.pdf-renderer: Framework - O PDF Renderer é um framework Java para renderizar arquivos PDF em componentes Swing. Ele fornece uma visualização de

PDF em uma interface gráfica. Usei ele em especifico para mostrar uma visualizacao final do PDF que foi gerado na conversão para nao precisar abrir.

vi. com.documents4j-local: Framework - O Documents4j é uma biblioteca que permite a conversão de documentos entre diferentes formatos, como Word para PDF. Esta é uma dependência para uso local do Documents4j.

vii. com.documents4j-transformer-msoffice-word: Framework - Uma extensão do Documents4j para a transformação de documentos do Microsoft Word para outros formatos. É uma dependência para o uso de documentos do Word com o Documents4j.

viii. uk.gov.gchq.gaffer.bitmap-library: Utilitário - Esta é uma biblioteca para trabalhar com mapas de bits. É um utilitário para manipulação de imagens em Java.

ix. commons-lang: Utilitário - Apache Commons Lang é uma biblioteca de utilitários para fornecer funcionalidades adicionais à linguagem Java padrão. Ele oferece uma variedade de classes e métodos auxiliares.

x. com.toedter.jcalendar: Framework - O JCalendar é um framework para trabalhar com calendários e escolher datas em aplicações Java Swing. Não cheguei a usar ainda, estou testando ele pois tem uma visualização dinamica de calendario.

xi. com.jgoodies.jgoodies-forms: Framework - O JGoodies Forms é um framework para criar interfaces gráficas de usuário (GUI) baseadas em Swing em Java. Ele fornece um layout de formulário flexível.

xii. com.miglayout-swing: Framework - O MiGLayout é um layout manager para aplicações Java Swing que oferece um layout flexível e poderoso.

xiii. com.formdev.flatlaf: Framework - FlatLaf é um framework para criar interfaces de usuário em estilo "flat" (sem sombreamento) em aplicações Java Swing. Ele fornece um visual moderno para interfaces gráficas. Usei ele para os temas das janelas.

4. IMPLEMENTAÇÃO

O projeto foi desenvolvido no Eclipse, com Maven para automação de tarefas. A aplicação possui uma interface gráfica que carrega as planilhas escolhidas em uma JTable, permitindo que o usuário edite e gere documentos formatados automaticamente. Inclui configurações específicas para manter a compatibilidade com diferentes ambientes.

4.1 ESTRUTURA DO PROJETO

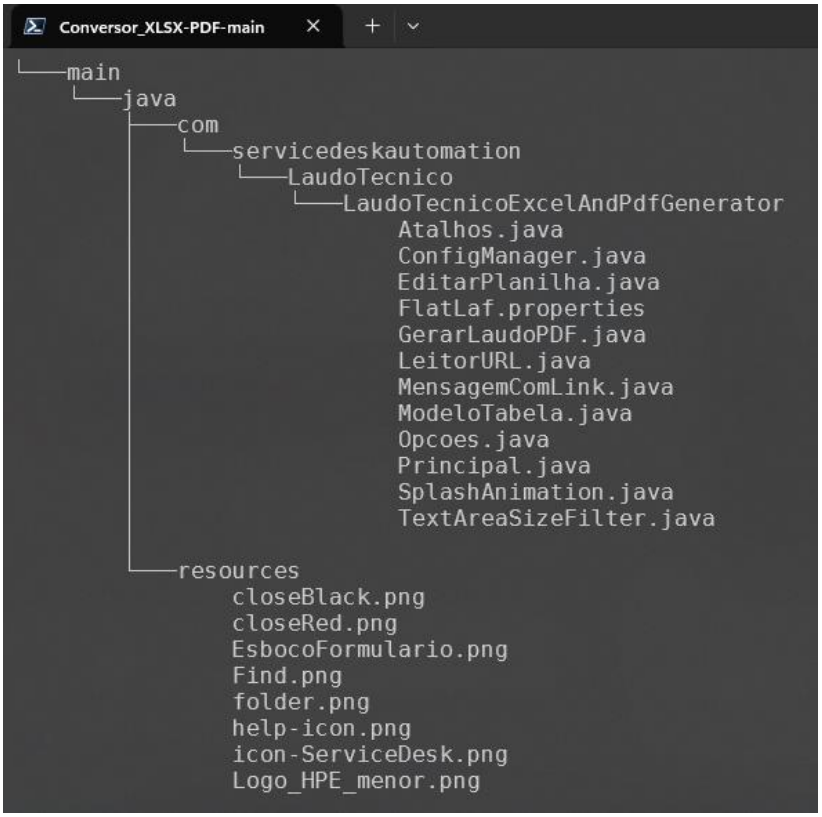

Figura 2: demonstração da automatização e exportação.

- main/java/com/servicedeskautomation/LaudoTecnico/LaudoTecnicoExcelAndPdfGenerator: Contém as classes Java principais para a geração de laudos em Excel e PDF.
    - Atalhos.java: Gerencia atalhos de teclado.
    - ConfigManager.java: Gerencia configurações do aplicativo.
    - EditarPlanilha.java: Facilita a edição de planilhas Excel (janela editar linha).
    - FlatLaf.properties: Configurações do tema FlatLaf.
    - GerarLaudoPDF.java: Responsável pela geração de laudos em PDF (janela gerar laudos em pdf).
    - LeitorURL.java: Lê dados de URLs para fazer download do modelo de laudo.
    - MensagemComLink.java: Gerencia mensagens com links.
    - ModeloTabela.java: Define o modelo da tabela.
    - Opcoes.java: Configura opções do aplicativo (janela de opções).
    - Principal.java: Classe principal do aplicativo (janela principal).
    - SplashAnimation.java: Exibe a animação de inicialização.
    - TextAreaSizeFilter.java: Filtra entradas em áreas de texto.
- main/resources: Contém recursos gráficos utilizados no aplicativo, como ícones e imagens.

4.2 CONFIGURAÇÃO MAVEN

Para garantir a consistência dos recursos e dependências, a configuração Maven inclui:

- Builds completas e builds simplificadas (para criação de executáveis).
- Caminhos específicos de recursos definidos no pom.xml para evitar problemas de acesso a imagens e outros recursos durante o build.

Além disso, a configuração de build utilizada no Maven inclui os seguintes goals:

- Build completa:
  - clean install: Remove arquivos gerados anteriormente e instala as dependências.
  - clean compile install verify: Remove arquivos anteriores, compila o projeto, instala as dependências e executa verificações.
- Build simples (somente JAR com dependências inclusas e execução automática da aplicação após a compilação):
  - clean compile assembly:single exec:java: Remove arquivos anteriores, compila o projeto, cria um JAR único com todas as dependências e executa a aplicação automaticamente.

Essa configuração é utilizada para criar o arquivo executável posteriormente em ferramentas como Launch4j, Inno Setup, JSmooth, Exe4j, LaunchAnywhere, install4j, Excelsior ou JexePack.

5 FLUXO DE EXECUÇÃO E INICIALIZAÇÃO DO PROGRAMA

Começaremos pelas animações de entrada (SplashAnimation.java), realizei essa animação simples (Figura 3) que realiza um carregamento de 0 a 100% proporcional ao tempo de leitura e gravação dos arquivos necessários.

Figura 3: SplashAnimation.java

Na etapa do 0-20% é realizado a verificação de arquivos e pastas dependentes, dentre os arquivos verificados, verifica se o arquivo de configuração existe e captura a primeira linha, se não existir ele vai ser criado. Depois verifica se o arquivo de modelo do laudo que contêm cabeçalhos personalizados de referência para o nosso código existe em nosso arquivo de configuração, caso positivo irá salvar o estado e prosseguir para o carregamento da SplashAnimation, caso negativo, irá acionar uma flag para realizar o download ou selecionar manualmente esse arquivo.

Na etapa dos 20-50%, caso o usuário esteja entrado na condição do pior caso (de ter que fazer o download) irá aparecer uma nova janela com algumas instruções conforme na imagem abaixo.

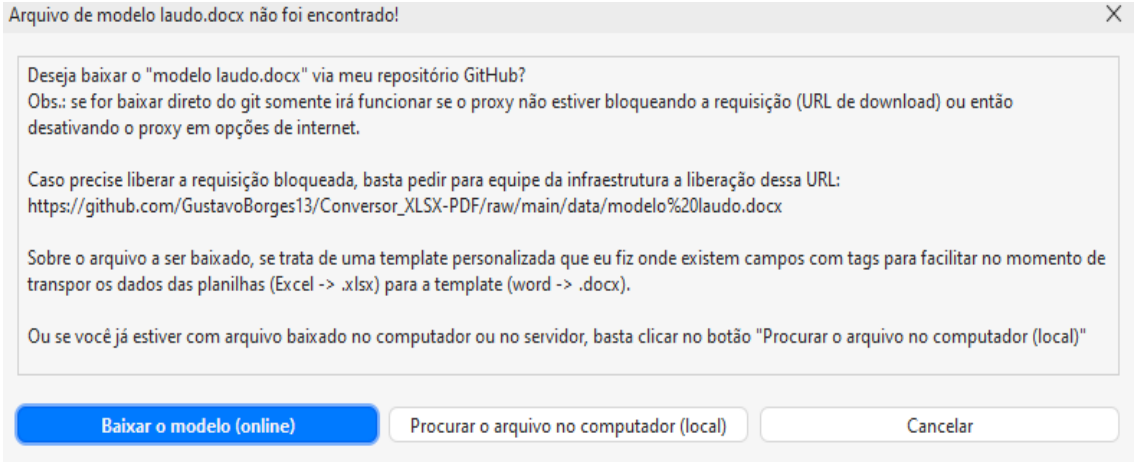

Figura 4: arquivo de modelo de laudo.docx.

Caso baixe o modelo online (pior caso), é feito o download do modelo de laudo técnico disponível no meu github caso tenha sido selecionado na etapa anterior, e após baixado é atualizado o local path desse modelo no arquivo de configuração, essa operação de download do arquivo necessário é realizada na classe LeitorURL.java.

Caso tenha escolhido a localização manual do mesmo (médio caso), foi feito uma restrição de que o arquivo template é único, logo, se eles alterarem algo no documento original irá dar uma exeception, pois o arquivo tem que estar entre os seus 27-40KB de tamanho para evitar quebrar o código caso não tenha sido usada as mesmas configurações de origem. Nesse tempo de escolhas o tempo da splash Animation é freezado.

Caso o arquivo já exista (melhor caso) e o path ter sido armazenado no arquivo de configuração) ele irá prosseguir para a próxima etapa. Vale ressaltar que, a criação e gerenciamento do arquivo de configuração é feito através da classe ConfigManager.java.

Na etapa 50-60% irá realizar outra verificação por garantia caso tenha ocorrido algum problema na condição anterior ou alguém tenha excluído acidentalmente no mesmo instante para conseguir carregar o modelo corretamente.

Na etapa 60-80% será feito o ajuste da UI tais como temas e configurações da interface gráfica feitas pelo usuário, mas que infelizmente não tive tempo de implementar essas funcionalidades, logo, é apenas um texto visual.

Na etapa 80-100% é feito a conexão com o sistema e a outra classe java chamada Principal.java responsável pela parte de interação de toda a nossa aplicação e dispensar a splash Animation.

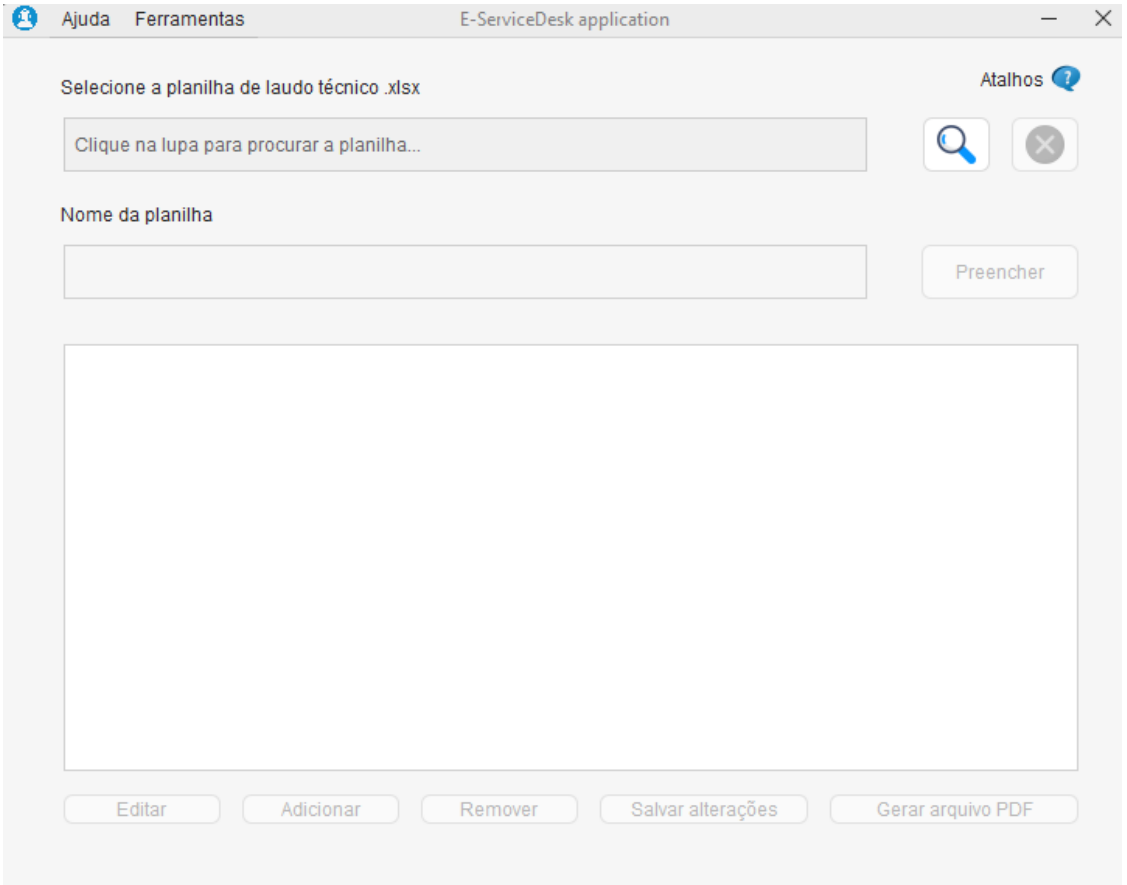

Figura 5: interface gráfica da janela principal (Principal.java).

Aqui nessa classe como pode ser visto, foi utilizado um JMenuBar para oferecer mais recursos: ajuda e ferramentas.

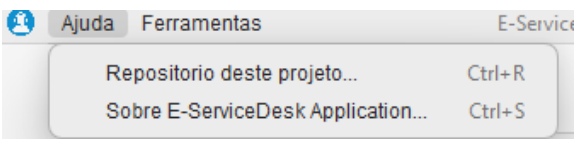

Figura 6: JMenu Ajuda itens.

Nessa mensagem sobre as informações adicionais (Figura 7), usei JDialog na mesma classe principal pois era algo bem específico, e foi utilizado uma classe MensagemComLink.java para conseguir indexar o link (fazer hyperlink) e reutilizar em outros lugares caso necessário.

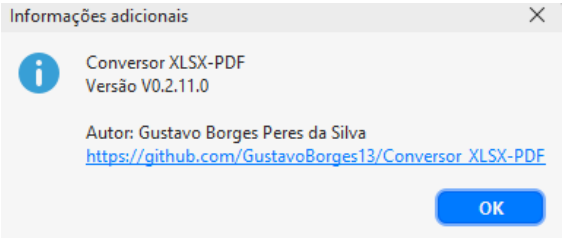

Figura 7: JMenuItem: repositorio deste projeto.

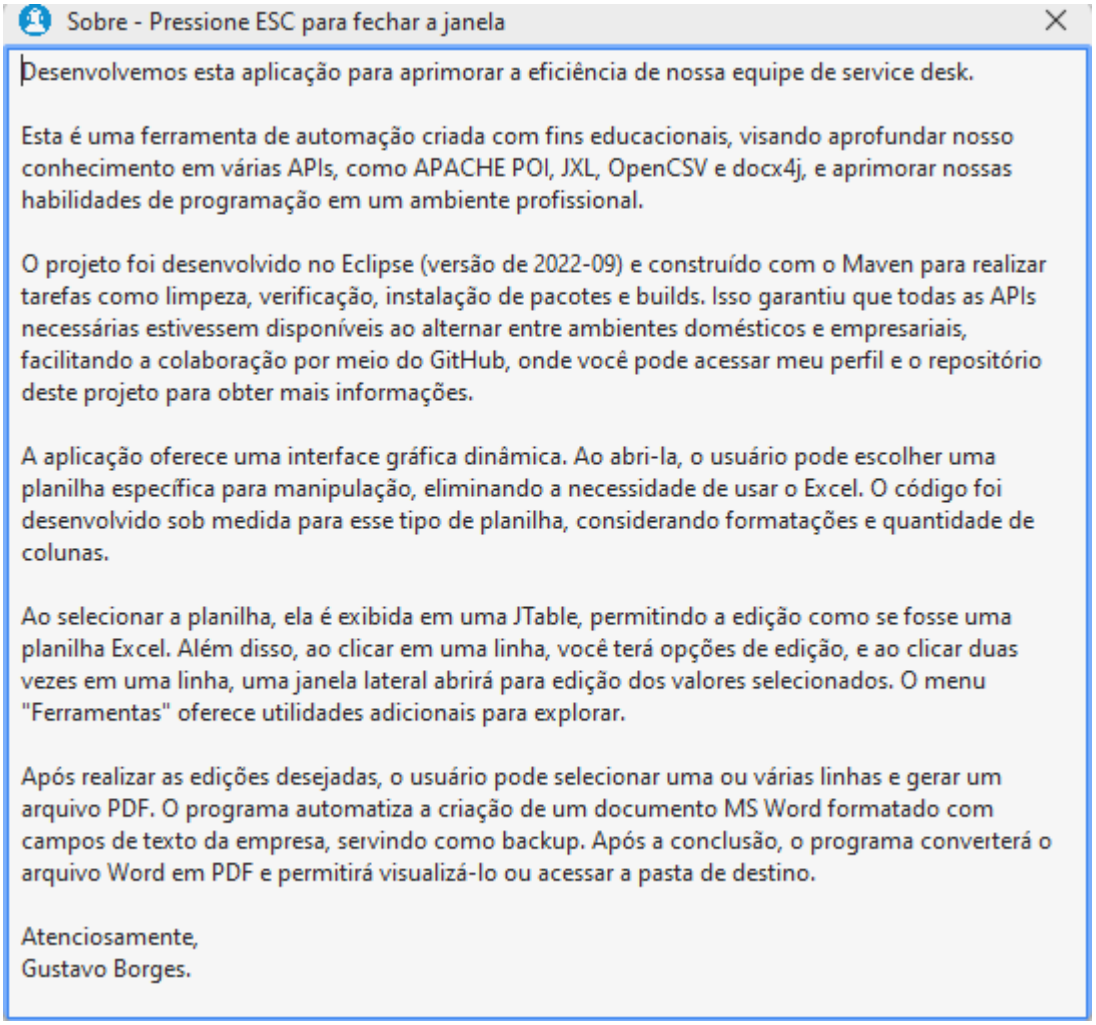

Figura 8: JMenuItem: sobre E-ServiceDesk Application.

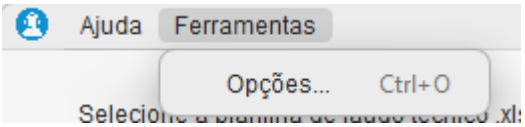

Figura 9: JMenu Ferramentas itens.

Nesse item, será carregado outra classe chamada Opcões.java que extende JDialog pois ao invés de usar JOptionPane

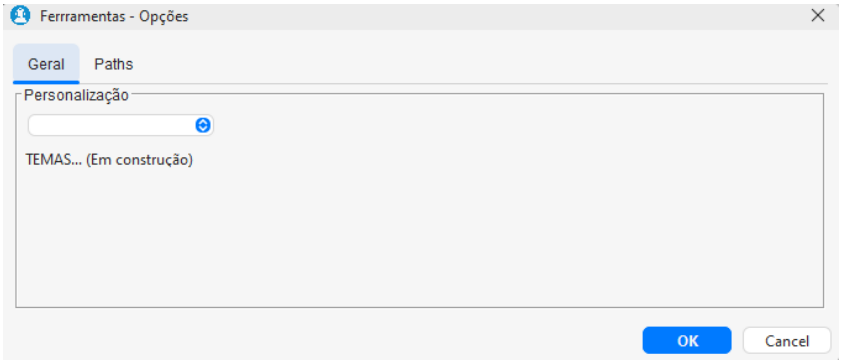

Figura 10: JMenuItem: Ferramentas – Geral (beta).

Abaixo temos os paths retirados do nosso arquivo de configuração gerado pela primeira vez ao rodar o programa.

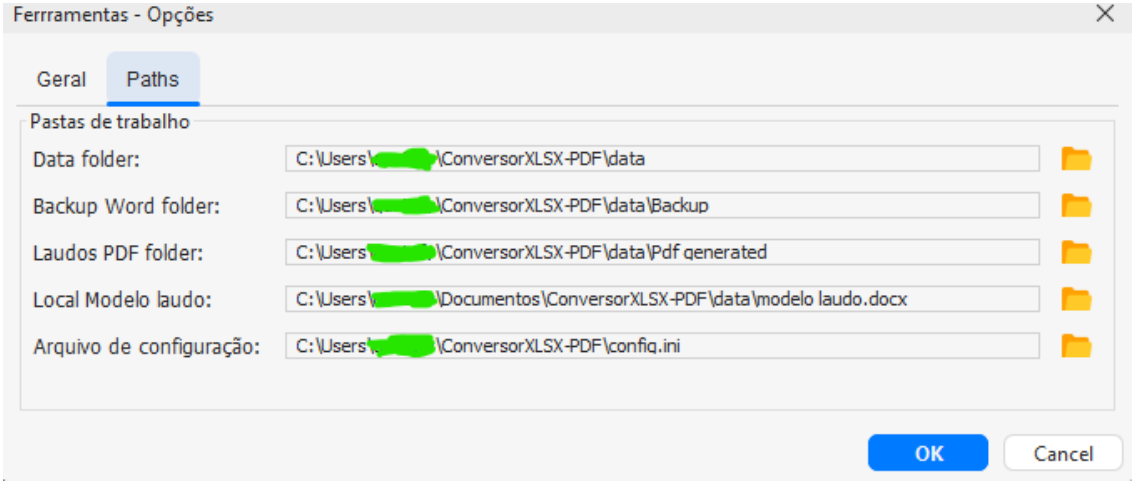

Figura 11: JMenuItem: Ferramentas – Paths.

Abaixo temos a Figura 12 referente aos atalhos direcionados a respectiva janela, ou seja, vamos ter outras janelas com outros atalhos específicos que não funciona em outras janelas, ao clicar no ícone irá ser aberto uma janela através da classe Atalho.java, mas que não tive tempo de implementá-la.

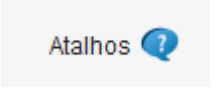

Figura 12: Atalhos referentes a janela atual.

Na seleção de arquivos de laudos técnicos usamos o JFileChooser com filtro de apenas arquivos xlsx para facilitar na busca, e como vemos aqui, conseguimos acessar arquivos

OneDrive e serviços clouds caso alguém vincule uma pasta online, assim podemos resolver o problema inicial que havíamos da migração do pacote office offline para o online.

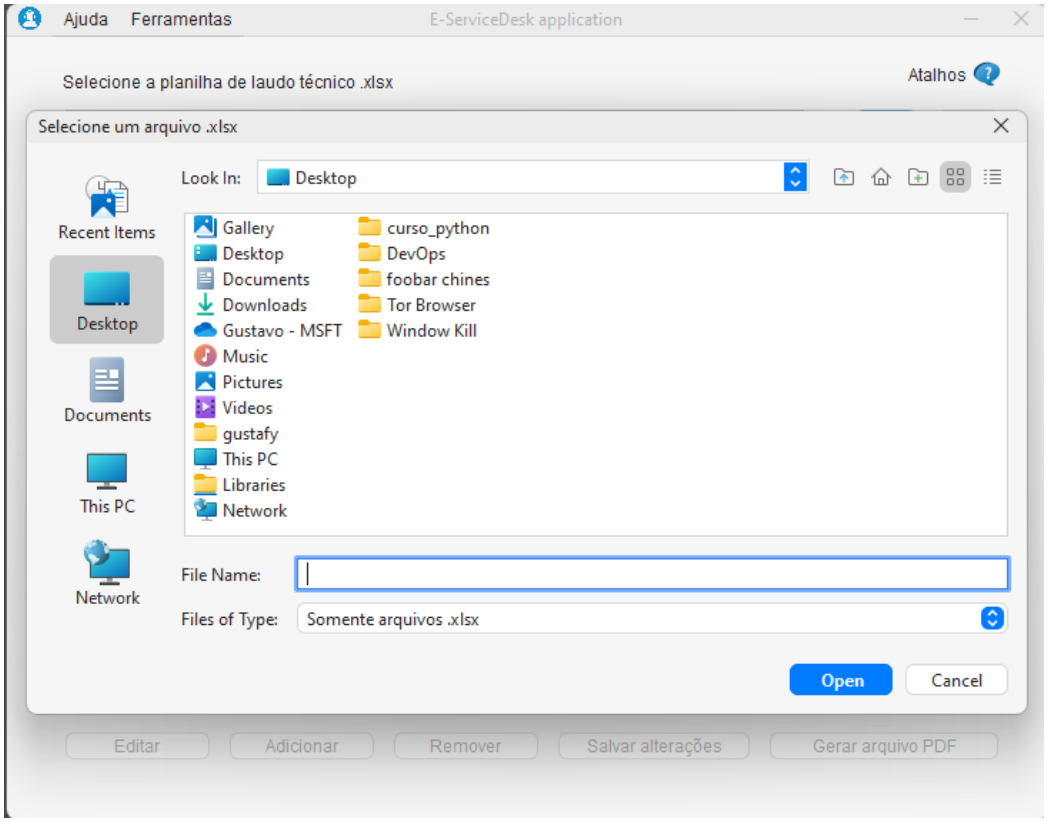

Figura 13: gerenciador de arquivos JFileChooser.

O interessante de usarmos um arquivo de configuração é que ele irá armazenar esse local do arquivo de laudo técnico no arquivo de configuração, assim conseguiremos otimizar o código e fazer resumo desses paths para facilitar na navegação. Por exemplo, na primeira vez que a gente for escolher o local onde o arquivo está salvo irá salvar esse path para que na próxima vez o JFileChooser abra no local raiz onde o ultimo arquivo foi aberto para tornar mais rápida a experiencia e evitar de salvar dados sensíveis no github pois caso tenha selecionado a pasta de um servidor o local do mesmo irá apenas ficar indexado no arquivo de configuração local criado em “./Documentos” ou “./Documents”.

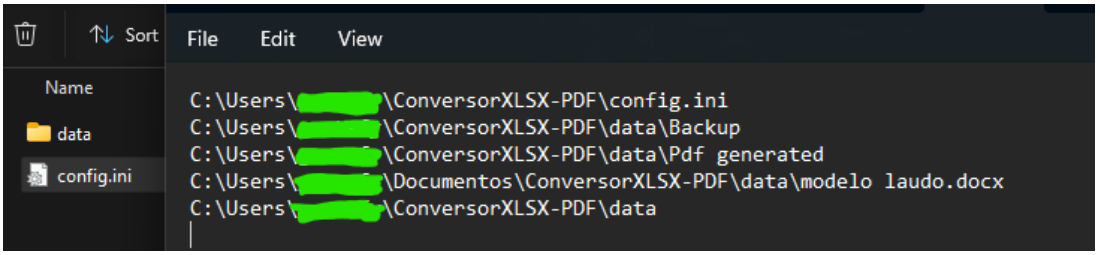

Figura 14: conteúdo do arquivo de configuração config.ini.

Como podemos observar, o único path que não está no mesmo diretório é o do modelo laudo pois ele é algo que temos que baixar ou selecionar manualmente. Além dele tem o path do local dos laudos técnicos que não foi mencionado para proteção de dados pois geralmente fica salvo no servidor da empresa esse arquivo.

Após abrir uma planilha adequada, o programa vai preencher uma JTable com a mesma formatação feita no excel (Figura 15), essa tabela é formatada no código da classe Principal.java e seu objeto é ModeloTabela.java para conseguirmos manipular valores entre outras classes. Basicamente teremos um espécime de excel aqui onde se dermos dois cliques na linha poderemos editar a linha selecionada.

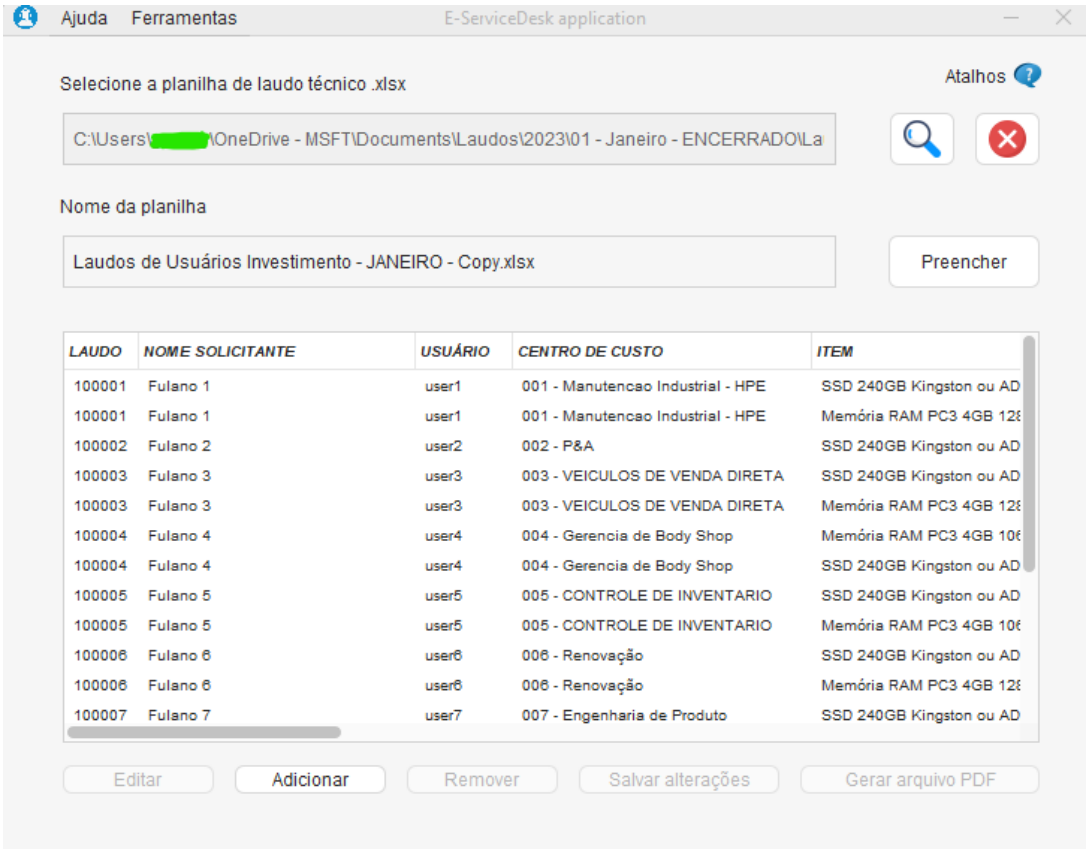

Figura 15: JTable preenchida.

Ao selecionar uma determinada linha, ela ficará azul (Figura 16) e irá habilitar novos botões (funcionalidades). E se dermos 2 cliques entraremos no painel de edição (EditarPlanilha.java) conforme na Figura 17.

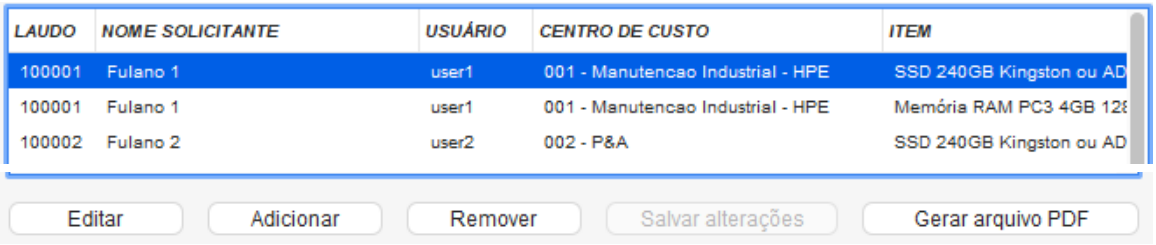

Figura 16: linha selecionada.

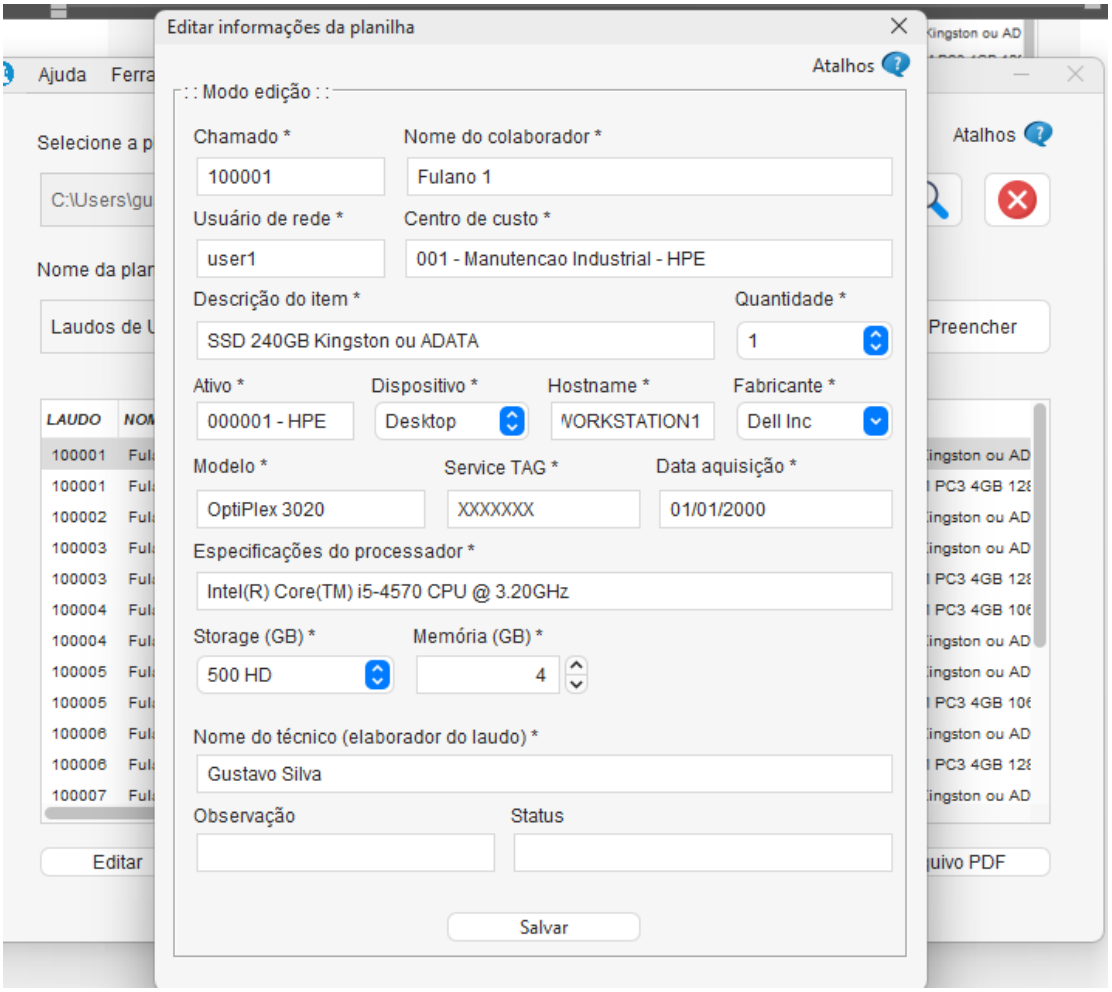

Figura 17: painel de edição.

Essa é uma das partes em que me orgulho bastante, deu bastante trabalho de implementar por conta da enorme quantidade de exceptions no código em geral. Os comboBox são interativos, é feita diversas comparações para verificar se o valor da planilha é adequado na nossa lista de comboBox e assim por diante, é feito um cálculo para memória para ver se o valor digitado está entre um range adequado e seleciona o valor mais próximo. E todos os campos com * são obrigatórios e irá apresentar diversos avisos caso esteja algum ausente como por exemplo textos em vermelho e JOptionPanes de diálogo para interagir. Além disso temos diversos atalhos implementados como por exemplo Ctrl+C e Ctrl+V para deixar mais rápido e TAB para mudança de campos.

Abaixo está alguns exemplos dessas funcionalidades ditas.

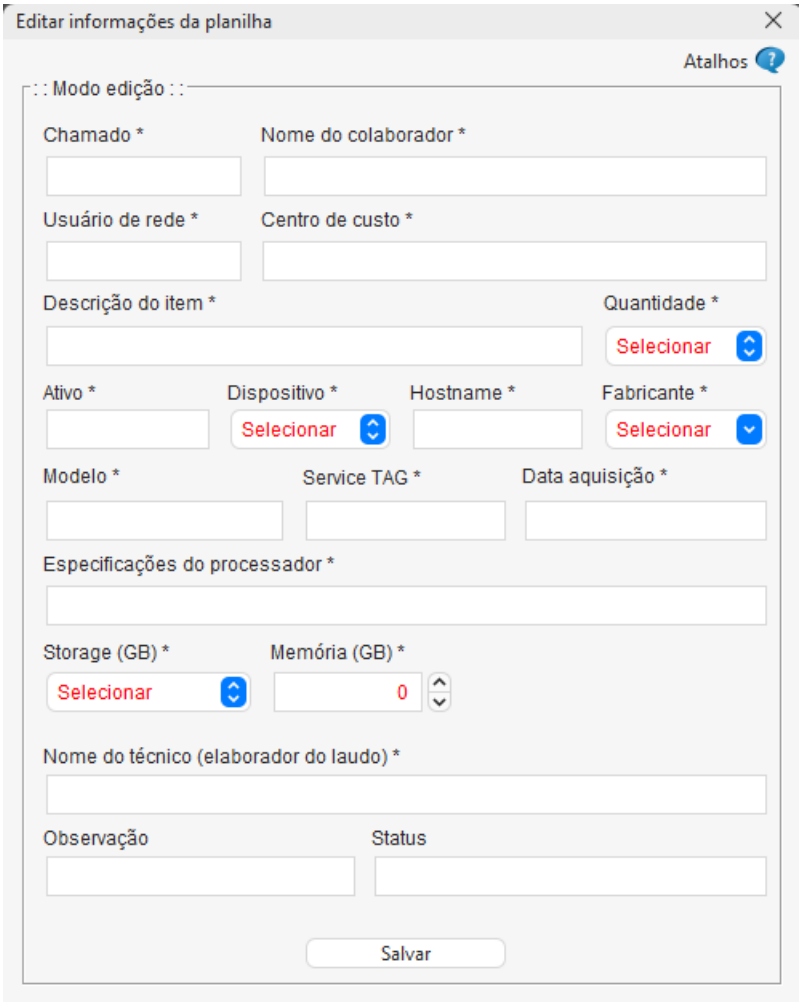

Figura 18: painel de edição – Adicionando uma linha.

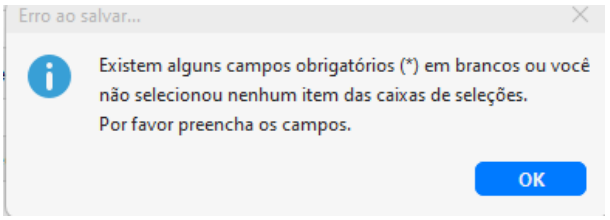

Figura 19: painel de edição – Adicionando uma linha – Exemplo de exception.

E claro, isso foi apenas um exemplo de exception, existem diversas outras que até me fogem a quantidade exata.

Mas outra coisa interessante é que na tabela (Figura 15) existem linhas com mesmo número do laudo, solicitante, usuário, ativo e hostname pois para cada item do solicitante tem que ser separado por linhas para evitar problema na hora de realizar a compra por um funcionário especializado nessa área, tendo isso em vista, nós podemos selecionar duas linhas para geração do PDF que irei abordar posteriormente para gerar um arquivo word e gerar seu pdf com os itens

necessários de forma pratica, mas caso queira editar os dois ao mesmo tempo teremos uma exception como essa da Figura 20.

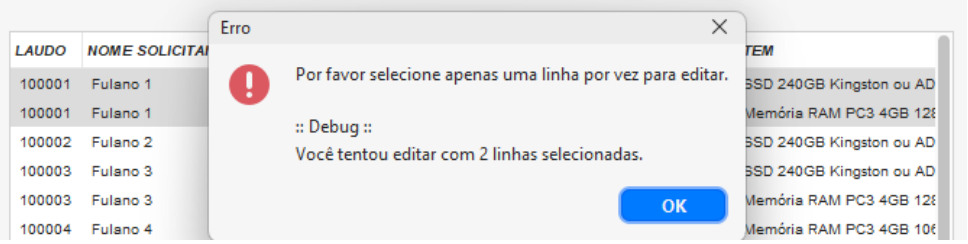

Figura 20: painel de edição – Tentativa de editar 2 linhas – Exemplo de exception.

Agora que vimos as funcionalidades importantes vamos para a cereja do bolo, que é a parte que irá realizar todo o trabalho pesado para gente (automatização). Essa funcionalidade é a gerar arquivo PDF fazendo parte de outra classe chamada GerarLaudoPDF.java que será ativada quando pressionar o botão de gerar sendo que para que ocorra é necessário ter uma linha ou mais linhas referentes ao mesmo laudo selecionadas.

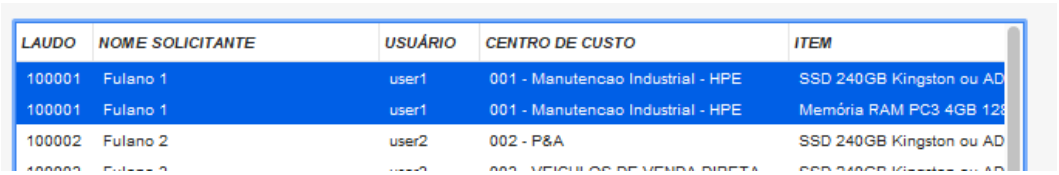

Figura 21: gerar um PDF com 2 itens – 2 linhas.

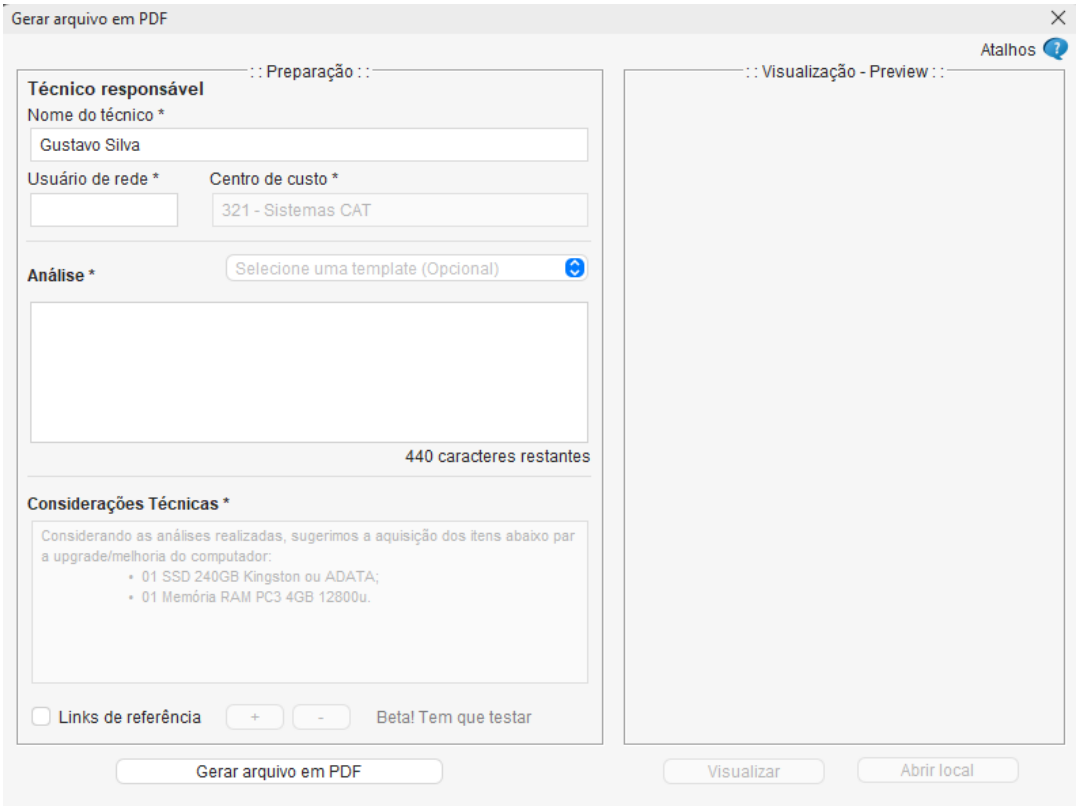

Figura 22: painel de gerar PDF – 2 linhas.

Conforme podemos observar na Figura 22, temos alguns campos para preencher, ele inclusive preenche automático o nome do técnico e o centro de custo para economizar o tempo, o centro de custo é trancado pois apenas o TI faz o laudo, e o usuário de rede é necessário escrever pois pode ser outro colaborador fazendo no lugar dessa pessoa caso tenha esquecido.

Na análise tem um textArea com limite de 440 caracteres esse sistema foi embutido com outra classe chamada TextAreaSizeFilter.java onde existem algumas templates de análises inclusas para acelerar o tempo.

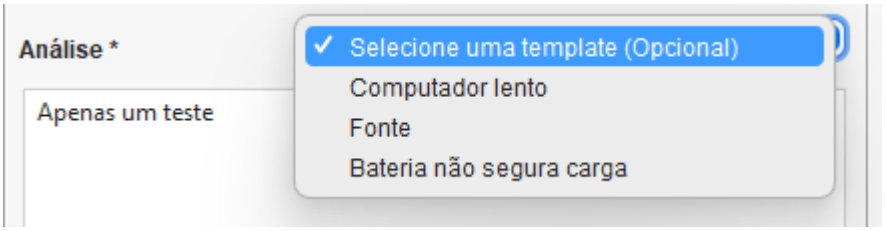

Figura 23: textArea - templates.

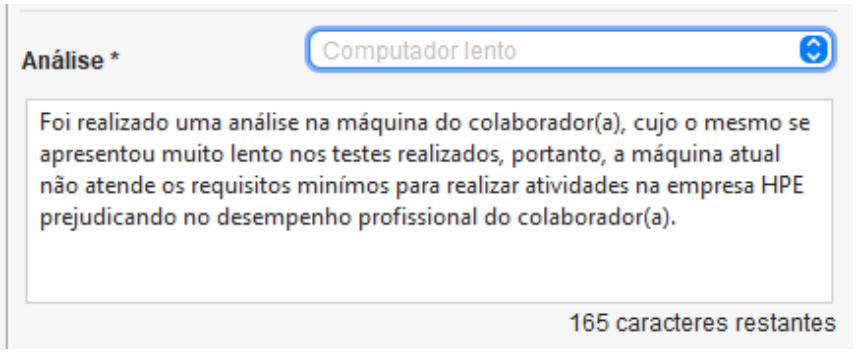

Figura 24: textArea – template: computador lento.

Além disso, temos as considerações técnicas que puxa os itens referentes as linhas selecionadas automaticamente e formata automaticamente mostrando como ficaria no final do documento, e caso opte por inserir mais informações tem a opção de links de referência para anexar algum link externo para alguma loja de compra por exemplo conforme podemos ver nas figuras posteriores.

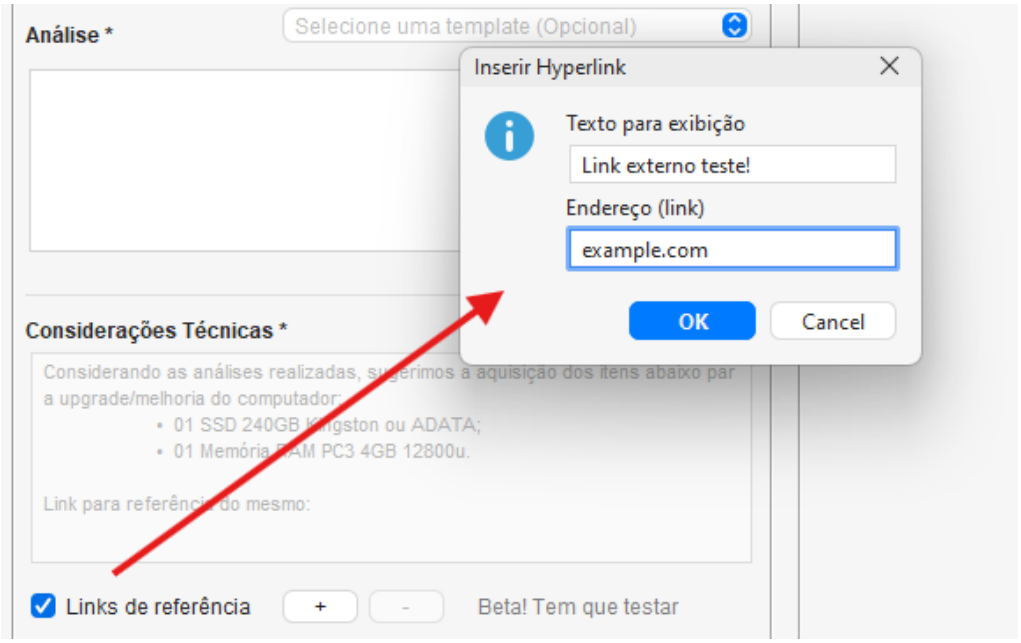

Figura 25: gerar pdf – links de referência.

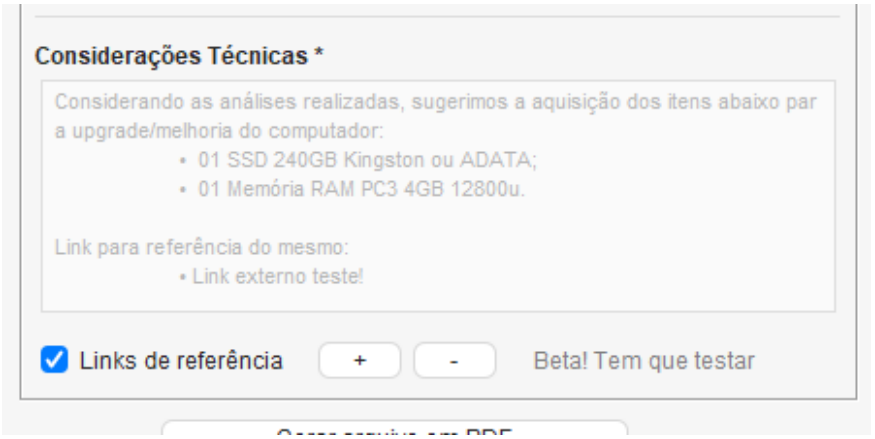

Figura 26: gerar pdf – links de referência adicionado com sucesso.

E caso deseja remover existe os dois botões de + e -, onde o - vai remover o último link inserido.

E no final, ao clicar no botão Gerar arquivo em PDF a mágica ocorre conforme podemos observar na Figura 27.

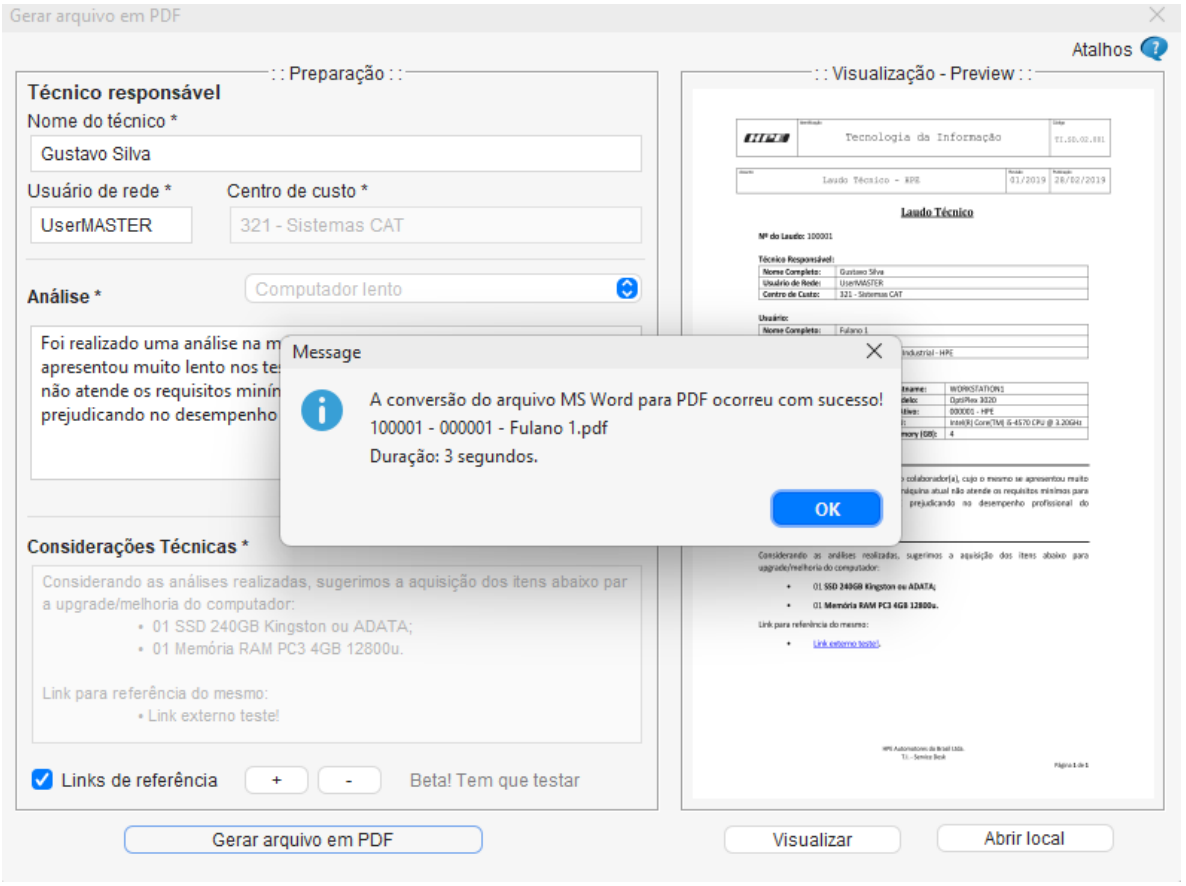

Figura 27: gerar pdf – geração finalizada.

Abaixo está o trecho do código responsável pela velocidade da conversão, aqui optei por usar mais threads e uma pool maior para deixar mais rápido levando até 3 segundos a conversão.

```
IConverter converter = LocalConverter.builder()
    .baseFolder(new File(pathTemp))
    .workerPool(50, 200, 5, TimeUnit.SECONDS)  // Aumenta o número de threads e o tamanho do pool
    .processTimeout(3, TimeUnit.SECONDS)       // Reduz o tempo limite do processo
    .build();
```

Código 1: trecho do código GerarLaudoPDF.java.

E ao final da conversão temos uma visualização/preview dela para verificarmos possíveis bugs antes de abrir, e caso queria abrir tem a opção visualizar, e caso queira navegar no local tem a outra opção de abrir localmente.

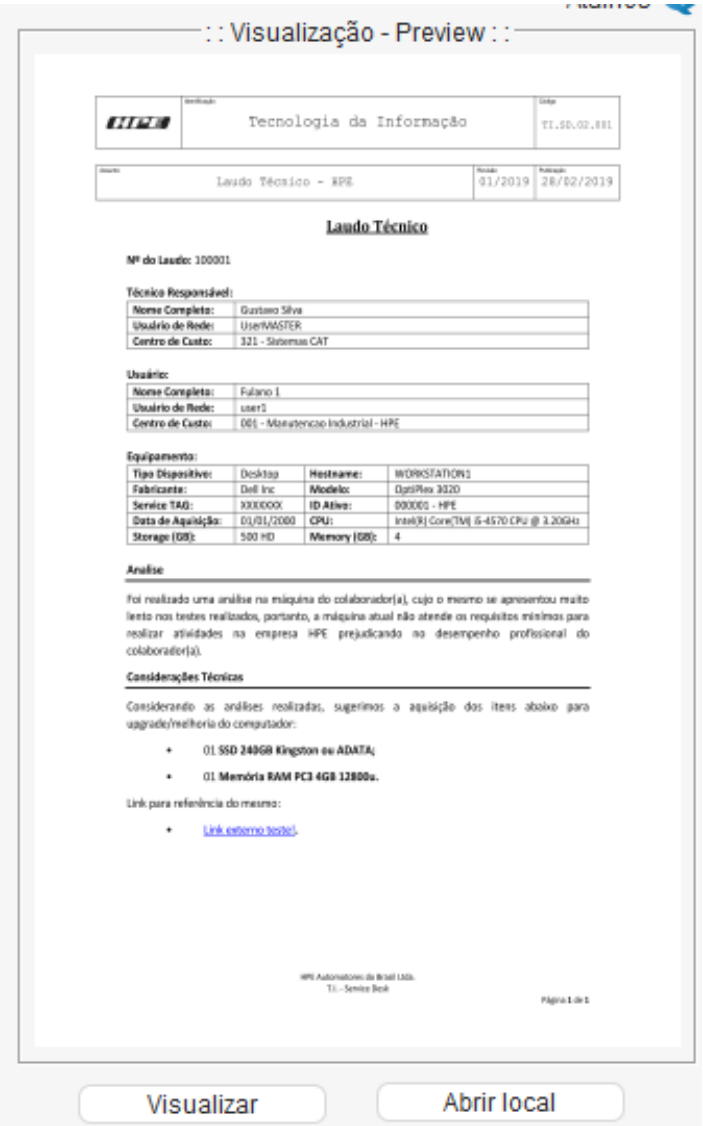

Figura 28: gerar pdf – visualização preview.

E sobre a organização dos arquivos gerados, o mesmo serão encontrados aqui na pasta do programa que é criada ou dentro da pasta “./user” ou “./Documentos” ou “./Documents”. Será gerado o arquivo word (docx) primeiramente e posteriormente o arquivo PDF a partir desse word.

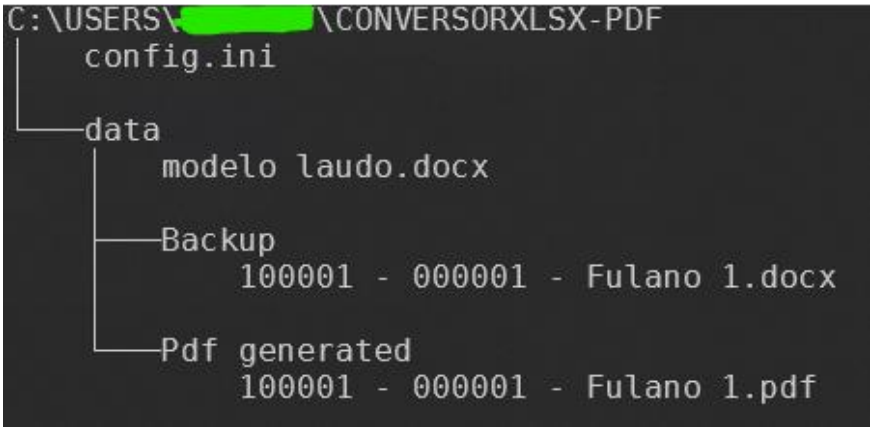

Figura 29: diretório de arquivos gerados.

6. MANUAL DO USUÁRIO

6.1 PRIMEIROS PASSOS

Instalar JDK 20+ e configurar o IDE para usar a versão correta para evitar esse erro abaixo.

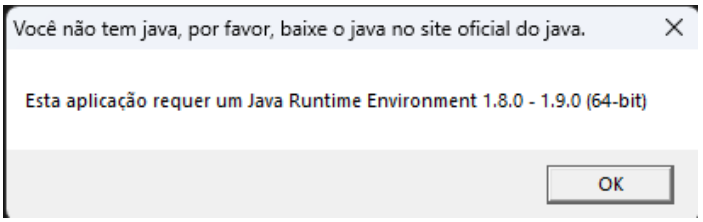

Figura 29: exception inicial.

Após rodar o programa pela primeira vez, irá solicitar ao usuário se deseja baixar o modelo de laudo online pelo GitHub, ou procurar o arquivo no computador (localmente).

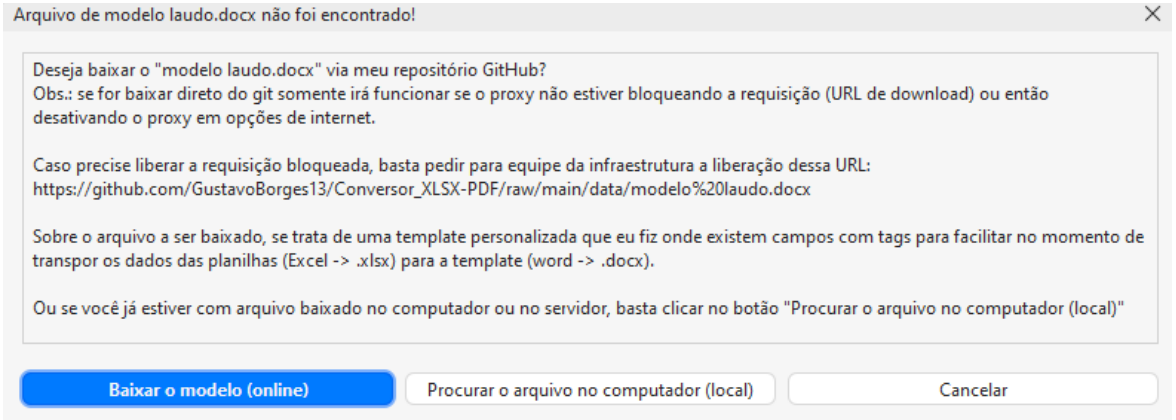

Figura 30: exception secundaria.

Para mais praticidade, iremos realizar o download do modelo online.

Após realizar o download o programa automaticamente irá carregar a janela de interface principal.

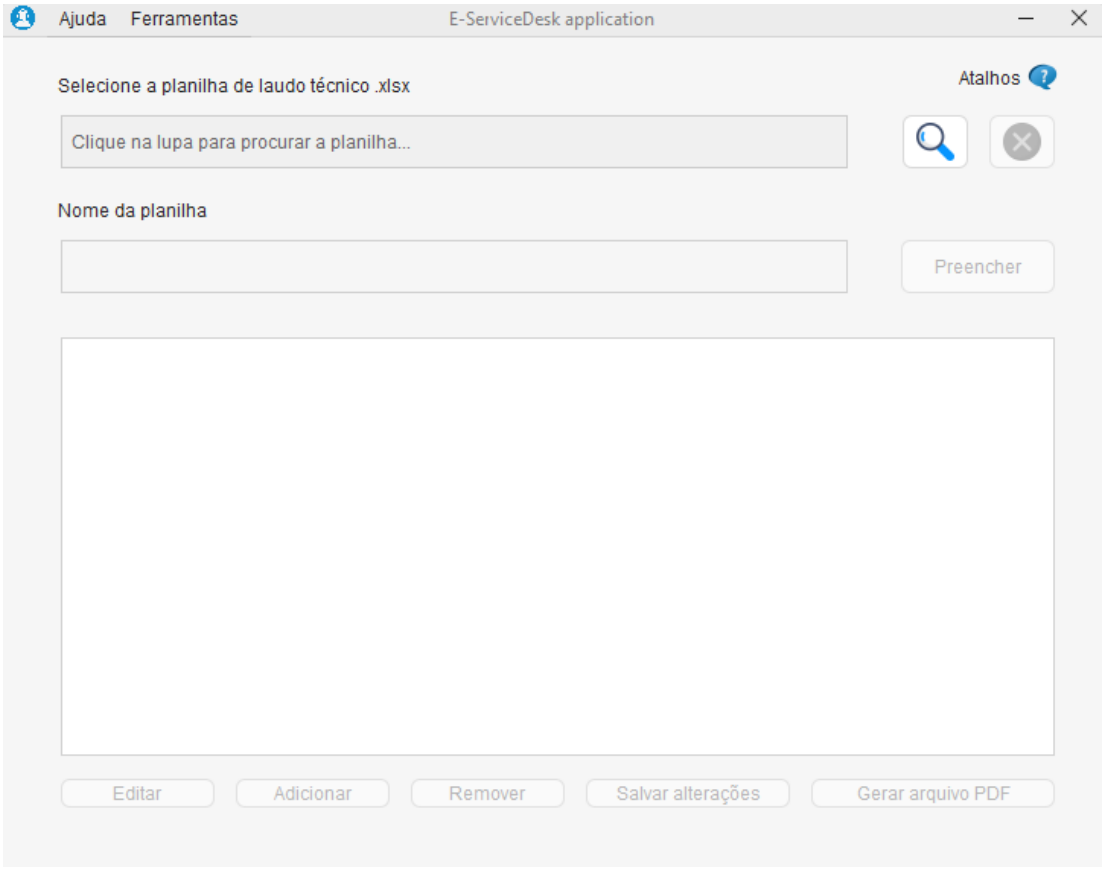

Figura 31: carregamento da interface gráfica.

6.2 USANDO A INTERFACE GRÁFICA

Para utilizarmos a interface, precisamos primeiramente procurar uma planilha compatível, nesse caso iremos utilizar um exemplo para testes, então primeiramente clique na lupa para procurar.

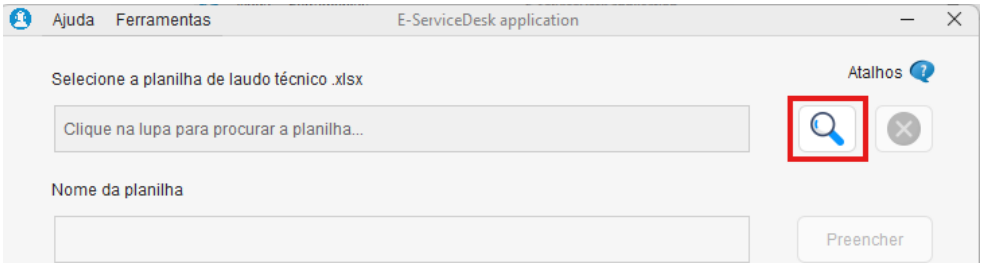

Figura 32: procurar planilha.

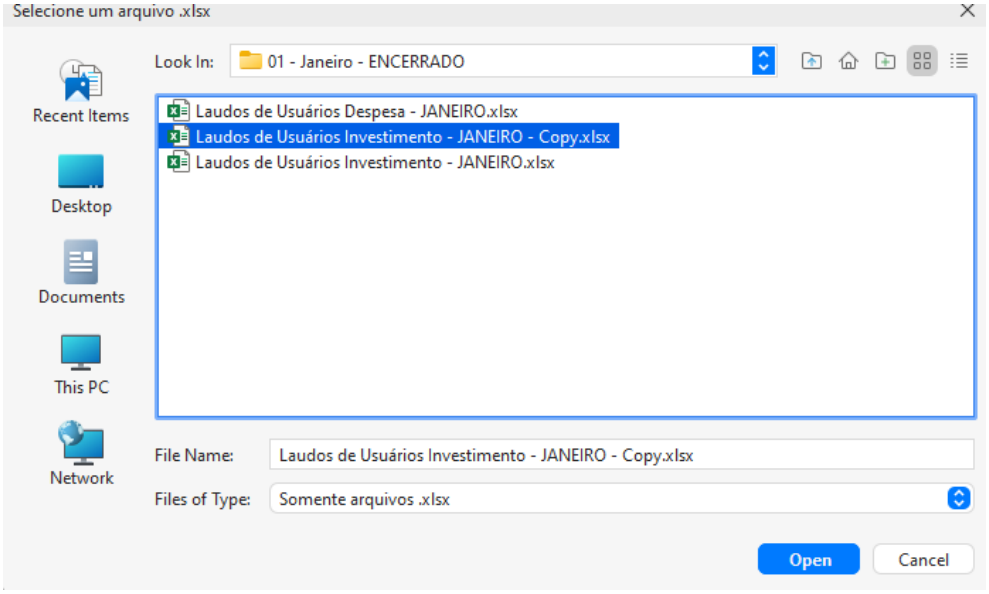

Figura 33: selecionar planilha.xlsx.

Agora que selecionamos, nada ainda aconteceu, pois precisamos preencher a nossa JTable (tabela) com os dados da planilha, para isso, basta clicar em “Preencher”.

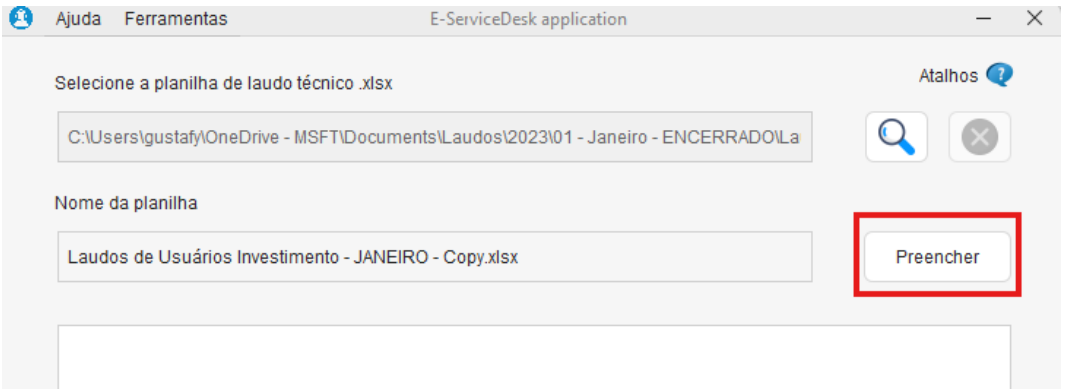

Figura 34: preencher tabela com dados da planilha.

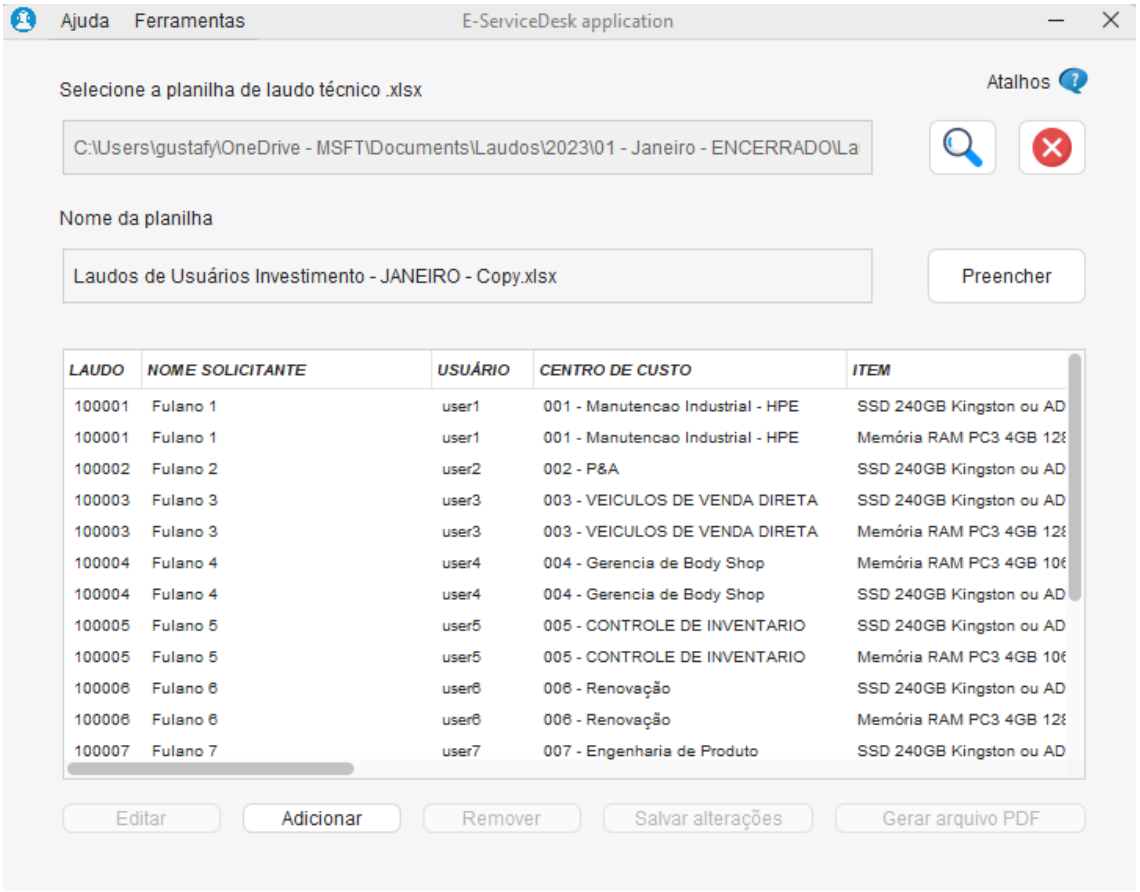

Figura 35: visão geral.

Agora iremos nos atentar na parte inferior onde estão localizados 5 botões, onde apenas 1 está desbloqueado, pois nenhuma linha ainda foi selecionada na tabela.

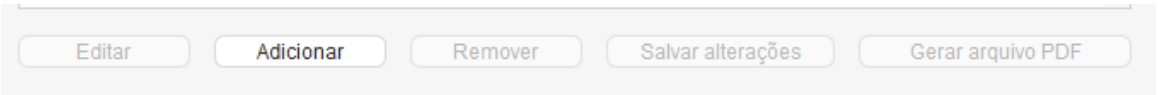

Figura 36: visão geral dos botões.

Ou seja, para desbloquear outras funções, ou a gente cria uma linha nova, ou seleciona alguma linha existente para editar, remover ou gerar arquivo pdf com base no conteúdo dessa linha selecionada ou conjunto de linhas com o mesmo laudo para combiná-las.

Para esse primeiro teste, iremos adicionar uma linha nova.

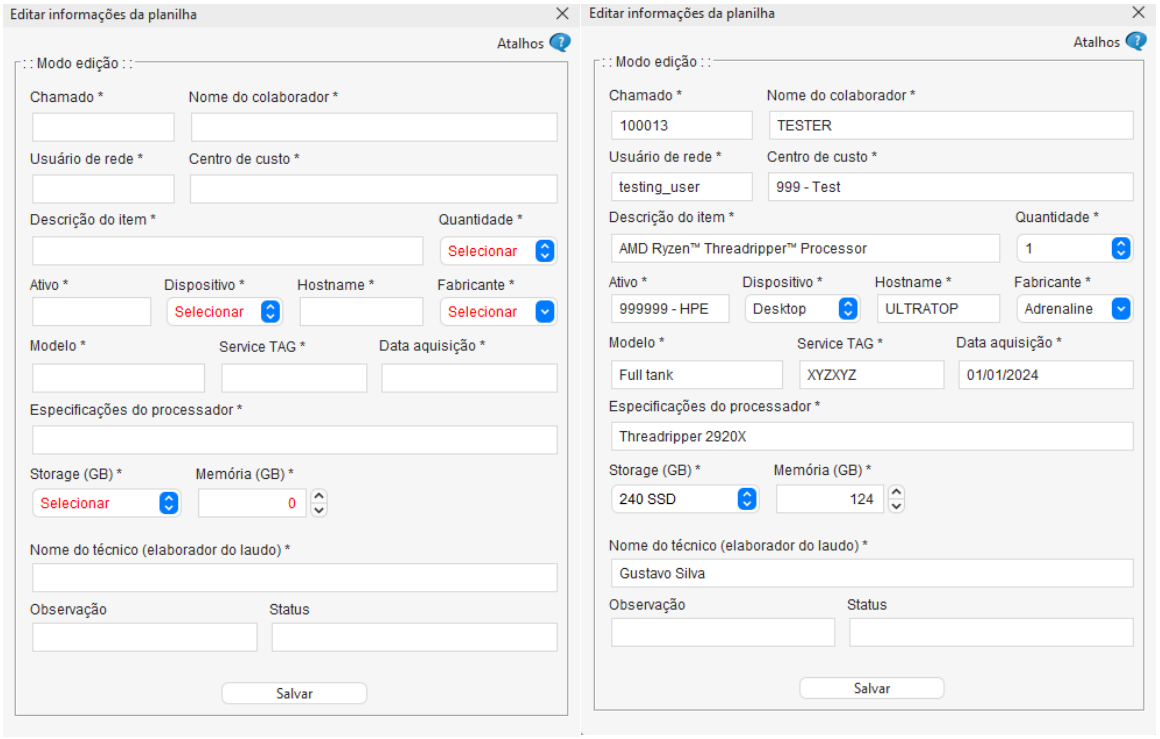

Figura 37: adicionando uma linha nova.

Pronto, agora desbloqueamos novos acessos.

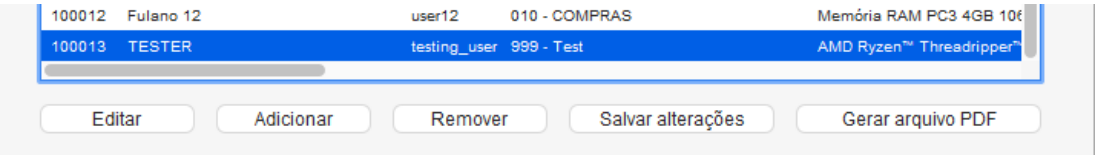

Figura 38: desbloqueio de novas funcionalidades.

Vamos editar então! Preciso mudar o nome do solicitante de “TESTER” para “Usuario” e do usuário de rede “testing_user” para “SuperUser”.

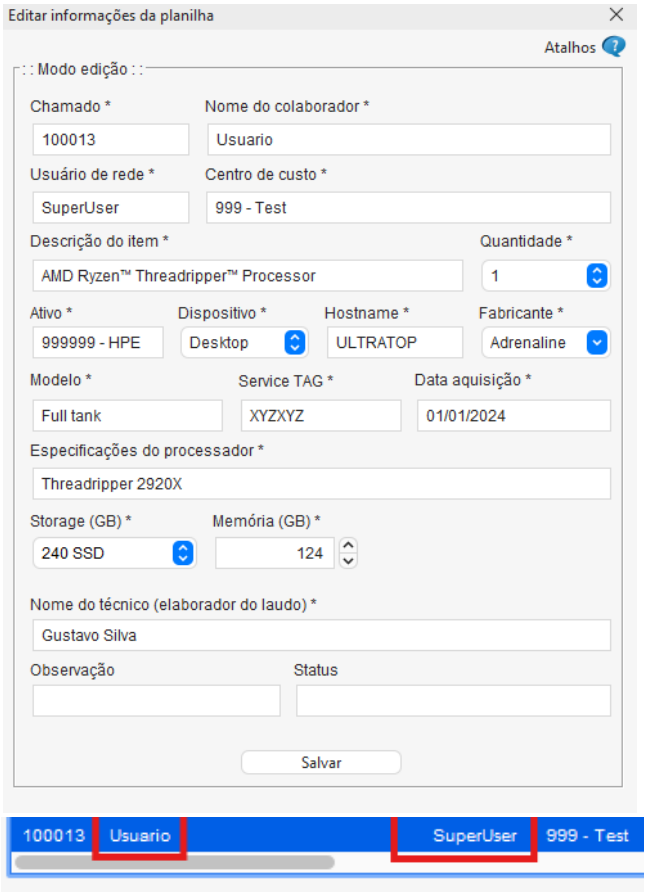

Figura 39: editando linha.

Ótimo, agora vamos remover o solicitante “Fulano 12”.

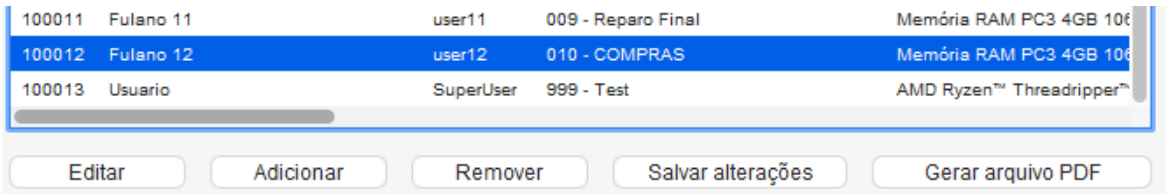

Figura 40: antes de remover linha.

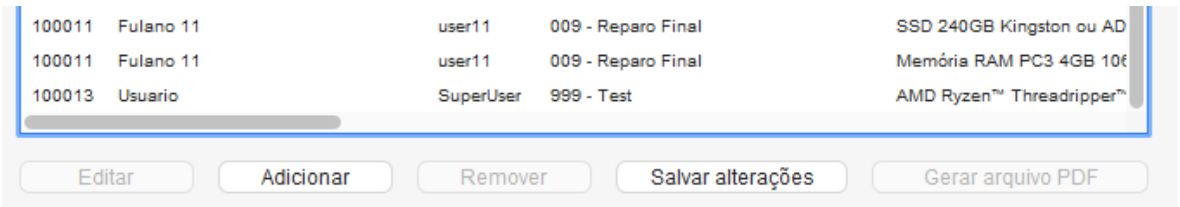

Figura 41: após de remover linha.

Perfeito, como já realizamos diversas modificações já conseguimos perceber que nosso botão de salvar alterações foi desbloqueado desde a primeira operação que realizamos (editar), logo, caso queira manter as operações clique em salvar alterações, caso contrário, clique em preencher a planilha novamente para dar um refresh (F5) na mesma e você irá perder as alterações feitas.

Neste caso iremos salvar depois para demonstrar uma exception interessante.

Agora vamos selecionar a nossa última linha e duplicar ela com Ctrl+C e Ctrl+V para modificarmos o item.

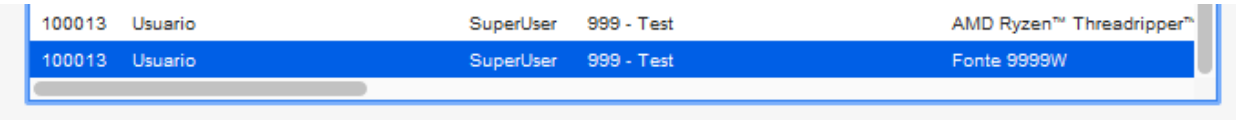

Figura 42: linha duplicada e modificada com outro item.

Agora vamos selecionar as duas linhas para combiná-las em nosso arquivo final do laudo pois as duas linhas pertencem aos mesmos números de laudo, e nessa combinação os valores da linha serão transpostos para o nosso modelo de laudo que foi feito em word e será convertido para PDF e posteriormente será enviado aos colaboradores via e-mail.

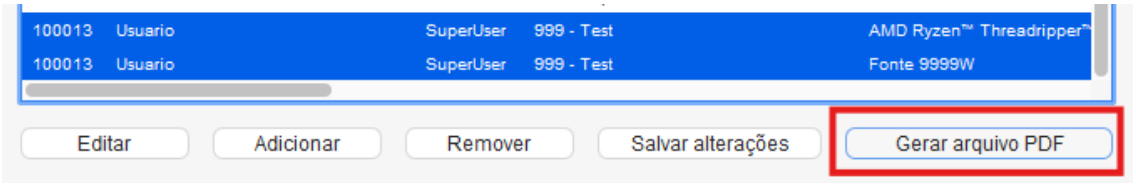

Figura 43: seleção de 2 linhas e preparação para gerar pdf.

Ao clicar em gerar arquivo pdf, iremos obter um aviso dizendo que ainda não salvamos as alterações realizadas na planilha, ou seja, o programa possui diversas correções para facilitar ao usuário a buscar a resolver os próprios problemas que encontra no programa.

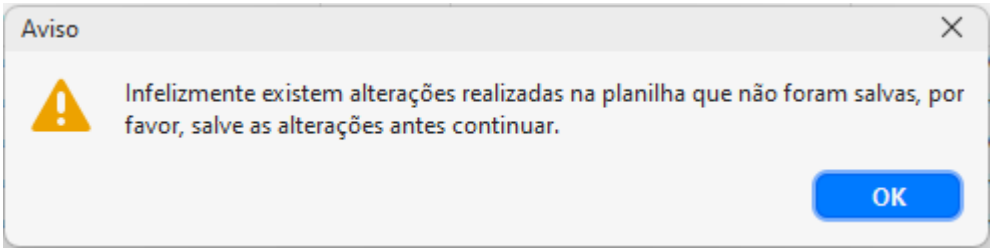

Figura 44: aviso ao tentar gerar pdf sem salvar alterações.

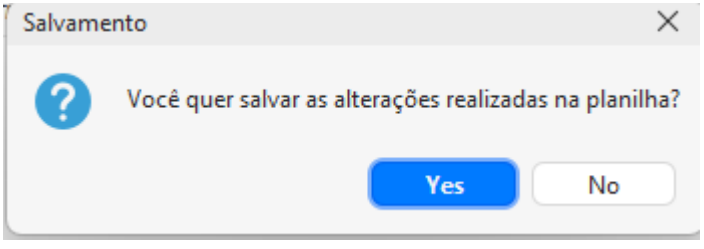

Figura 45: aviso ao salvar.

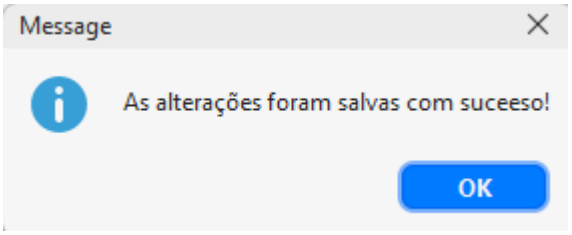

Figura 46: aviso que o salvamento foi um sucesso.

Abaixo veremos a janela referente a geração do arquivo em pdf. Aqui precisamos apenas preencher os campos de usuário de rede pois pode ser que outra pessoa tenha feito para o colega caso ele tenha esquecido. O centro de custo é intocável pois toda a equipe é do mesmo setor. A análise existe algumas templates para escolher e economizar tempo para digitar. E por último os links de referência caso queira referenciar alguma pagina em especifica com links externos.

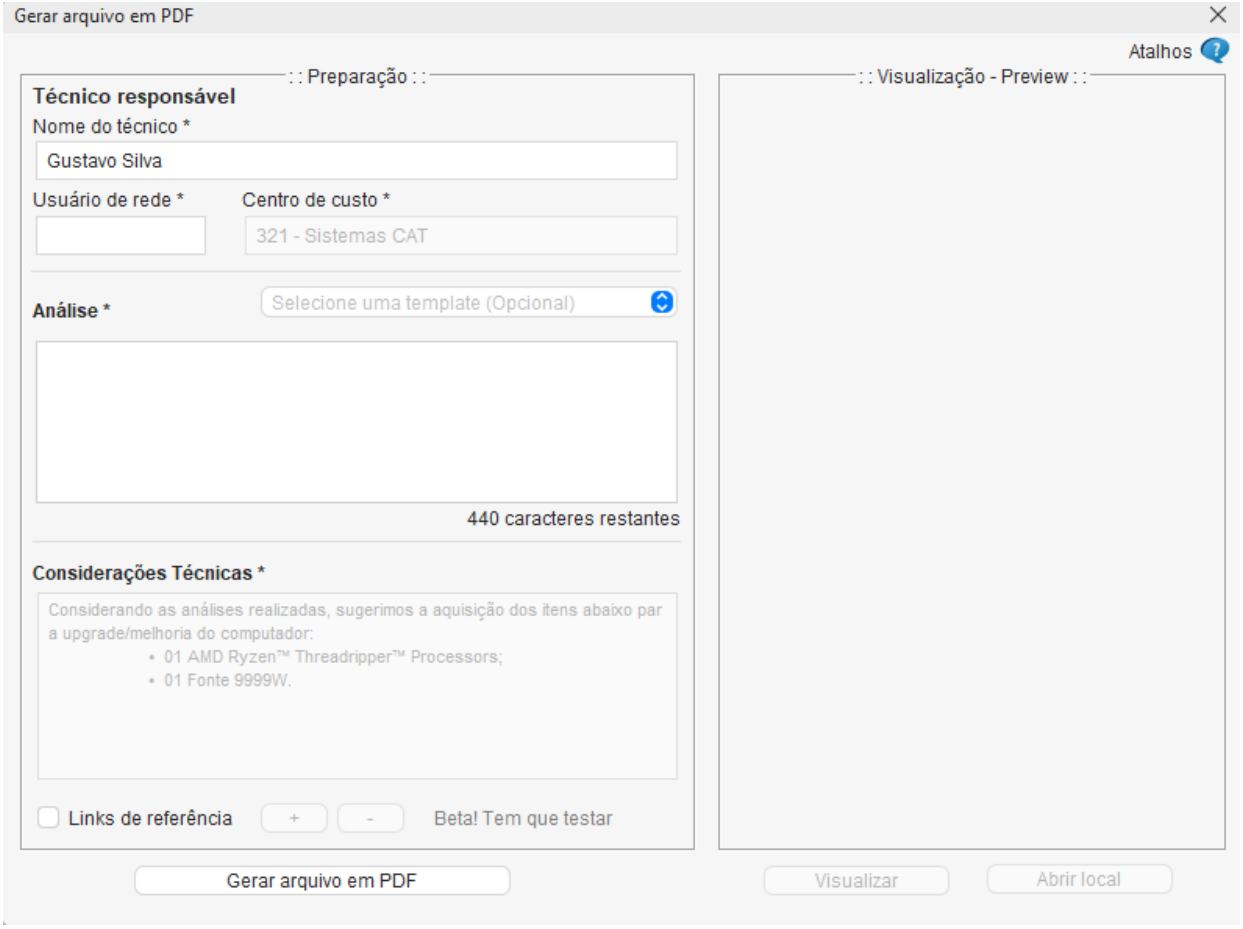

Figura 47: janela de gerar arquivo em pdf.

Das templates disponíveis temos 3, iremos utilizar o exemplo de computador lento.

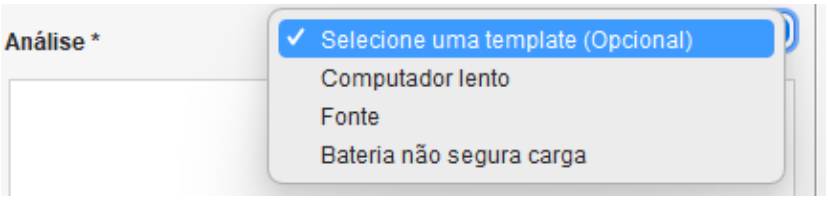

Figura 48: templates de análise.

Agora vamos adicionar links de referência para as nossas considerações finais, por exemplo, escolhemos duas linhas, a primeira linha se trata do item de um processador AMD, e a segunda linha fala de uma fonte para o computador, nesse sentido, poderíamos referenciar links de produtos associados a esses respectivos itens. No nosso exemplo vamos usar links genéricos para demonstrar.

Figura 49: considerações técnicas: links de referência.

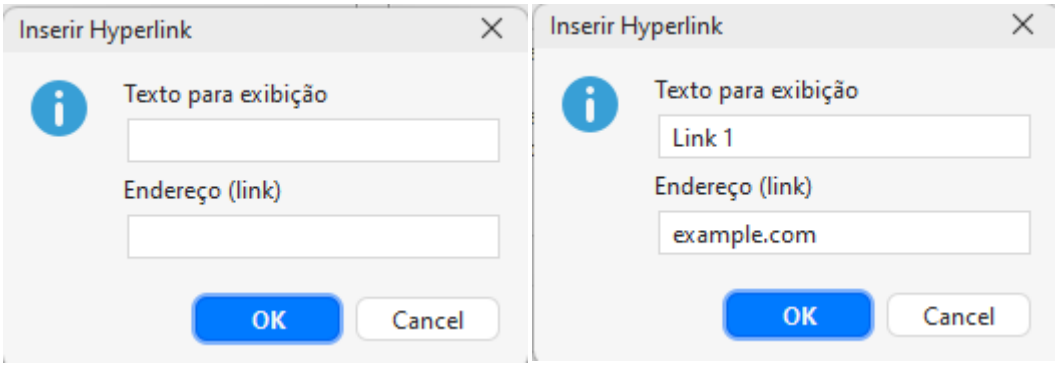

Figura 50: considerações técnicas: inserir hyperlink.

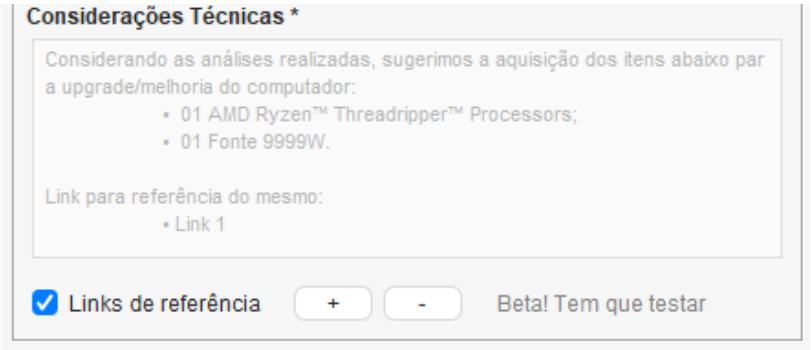

Figura 51: considerações técnicas: visão geral.

Caso optemos por excluir as referencias basta desmarcar a opção de links de referência, caso queira remover o último link, clique em “-”, caso quisesse adicionar mais links no final da lista clique em “+”. Nesse caso essa quantidade está Ok, vamos prosseguir.

Agora vamos clicar no botão “Gerar arquivo em PDF” e aguardar o processamento.

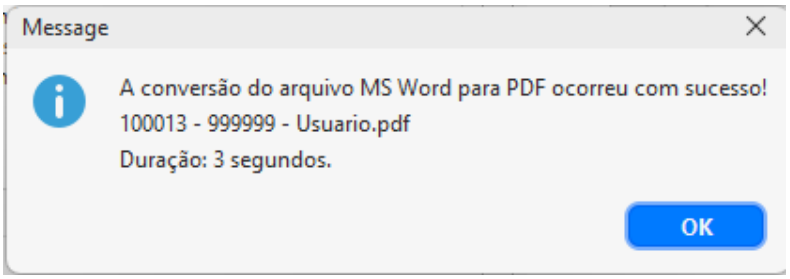

Figura 52: gerar arquivo pdf: conversão ocorreu com sucesso.

Como podemos observar na Figura 52, a conversão ocorreu bem rápido e gerou um arquivo de backup em word na pasta de configuração, e também o arquivo em pdf, ambos com o nome formatado da seguinte forma: “numero_laudo - ID_ativo - nome_solicitante.pdf ”.

E por fim temos ao lado direito a visualização/preview do nosso arquivo em pdf para que possamos editar sem precisar ficar abrindo.

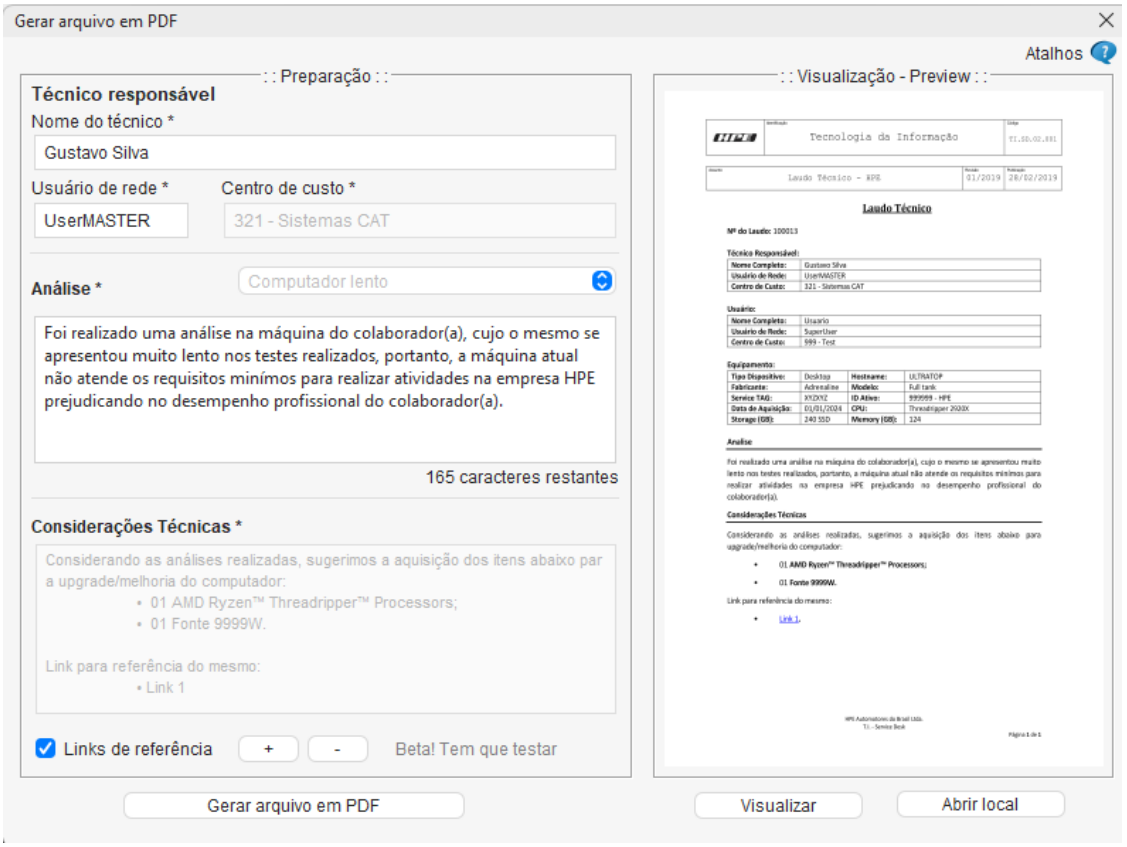

Figura 53: gerar arquivo pdf: visão geral e preview.

	Identificação	Código
HPE	Tecnologia da Informação	TI.SD.02.001

Assunto	Revisão	Publicação
Laudo Técnico - HPE	01/2019	28/02/2019

Laudo Técnico

Nº do Laudo: 100013

Técnico Responsável:

Nome Completo:	Gustavo Silva
Usuário de Rede:	UserMASTER
Centro de Custo:	321 - Sistemas CAT

Usuário:

Nome Completo:	Usuario
Usuário de Rede:	SuperUser
Centro de Custo:	999 - Test

Equipamento:

Tipo Dispositivo:	Desktop	**Hostname:**	ULTRATOP
Fabricante:	Adrenaline	**Modelo:**	Full tank
Service TAG:	XYZXYZ	**ID Ativo:**	999999 - HPE
Data de Aquisição:	01/01/2024	**CPU:**	Threadripper 2920X
Storage (GB):	240 SSD	**Memory (GB):**	124

Analise

Foi realizado uma análise na máquina do colaborador(a), cujo o mesmo se apresentou muito lento nos testes realizados, portanto, a máquina atual não atende os requisitos minímos para realizar atividades na empresa HPE prejudicando no desempenho profissional do colaborador(a).

Considerações Técnicas

Considerando as análises realizadas, sugerimos a aquisição dos itens abaixo para upgrade/melhoria do computador:

- 01 **AMD Ryzen™ Threadripper™ Processors;**
- 01 **Fonte 9999W.**

Link para referência do mesmo:

- Link 1.

HPE Automotores do Brasil Ltda.
T.I. - Service Desk

Figura 54: conteúdo do arquivo gerado em pdf.

Agora vamos comparar com o nosso modelo de laudo.

	Identificação	Código
HPE	Tecnologia da Informação	TI.SD.02.001

Assunto	Revisão	Publicação
Laudo Técnico – HPE	01/2019	28/02/2019

Laudo Técnico

Nº do Laudo: laudoz

Técnico Responsável:

Nome Completo:	
Usuário de Rede:	
Centro de Custo:	

Usuário:

Nome Completo:	
Usuário de Rede:	
Centro de Custo:	

Equipamento:

Tipo Dispositivo:		**Hostname:**	
Fabricante:		**Modelo:**	
Service TAG:		**ID Ativo:**	
Data de Aquisição:		**CPU:**	
Storage (GB):		**Memory (GB):**	

Analise

analisez

Considerações Técnicas

Consideracoez

Figura 55: conteúdo do arquivo modelo laudo em word.

Como podemos observar, temos a presença de Text Form Fields, eles são ferramentas disponíveis no word em modo developer que podem ser criados através desse menu (Figura 56).

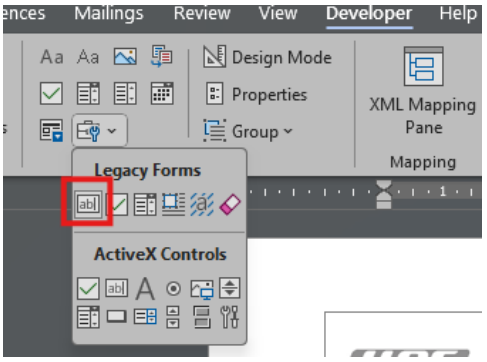

Figura 56: localização text form field.

Para melhor identificação coloquei textos fáceis de identificar para que no código eu pudesse manipular dados em campos específicos com mais facilidade. Nas tabelas não precisam pois existem métodos nas bibliotecas utilizadas que conseguem identificar a presença de tabelas. E para uma melhor compreensão abaixo está a configuração usada no text form field a diferença dos demais é somente o texto do nome.

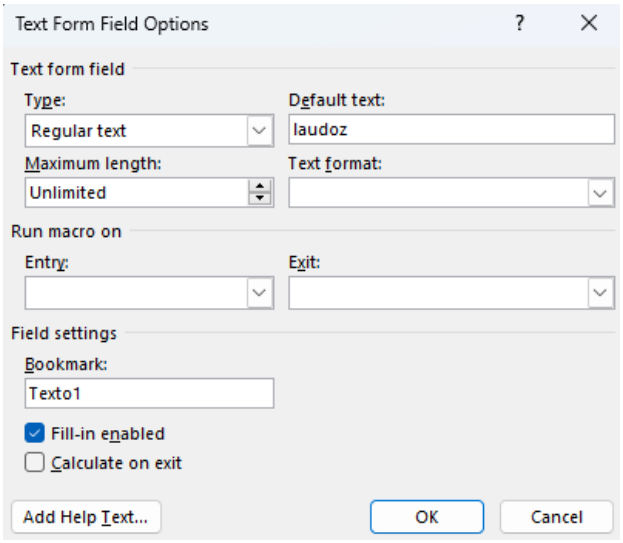

Figura 57: text form field options.

7. CONSIDERAÇÕES FINAIS

A implementação deste projeto foi um sucesso em atender aos objetivos propostos de forma eficaz. A automação da transferência e manipulação de dados entre planilhas do Excel e documentos Word/PDF resultou em uma melhoria significativa no fluxo de trabalho. Utilizando as bibliotecas Java, como apache POI e Documents4j, simplificamos o processo de geração de laudos técnicos, eliminando a necessidade de operações manuais repetitivas, como copiar e colar.

A nova ferramenta proporcionou uma experiência mais eficiente para os colaboradores, permitindo que se concentrassem em tarefas mais estratégicas em vez de se perderem em processos administrativos. Graças à automação, conseguimos reduzir o tempo necessário para a elaboração de laudos técnicos de 10-15 minutos para apenas cerca de 5 minutos. Essa otimização não apenas aliviou a carga de trabalho, mas também minimizou o uso do teclado, uma vez que os usuários agora se preocupam apenas em digitar as informações nas linhas da planilha, utilizando uma interface prática e compacta. Essa interface permite visualizar todos os campos em uma única janela, ao contrário do Excel, que, devido à quantidade de colunas, exigia o uso do scroll para navegar, tornando o processo mais cansativo.

Além disso, o programa executa automaticamente as etapas tediosas de copiar as informações e colar no modelo de laudo no Word, seguido pela geração do PDF, tornando o trabalho muito mais ágil e menos suscetível a erros.

A documentação do processo e a aplicação prática da solução destacaram a eficácia da automação na otimização do suporte técnico e na produção de documentos oficiais. Este projeto não apenas atendeu às expectativas iniciais, mas também abriu portas para futuras melhorias e inovações no fluxo de trabalho, garantindo um ambiente de trabalho mais ágil e eficiente. Agradeço novamente a todos os envolvidos que contribuíram para o sucesso deste projeto, cujo impacto positivo será sentido por muito tempo.

REFERÊNCIAS

[1] Gustavo Borges Peres da Silva (2022), “Conversor_XLSX-PDF”, https://github.com/GustavoBorges13/Conversor_XLSX-PDF.