1. OBJETIVO

Propor um algoritmo que possibilite encontrar correspondências entre bases de dados de medicamentos (IMS, SAMMED, ABCFARMA) para permitir a associação e o estudos dos preços destes medicamentos praticados por distribuidores, laboratórios, farmácias e hospitais privados/governo.

1. DESAFIOS PARA SOLUÇÃO
   1. **As tabelas não possuem relação explícita:** As tabelas não possuem atributos do tipo chave estrangeira, por isso o objetivo do trabalho é justamente inferir uma relação “de-para” (código FCC – código GGREM) de modo que se possa relacioná-las.
   2. **Não padronização da apresentação das informações:** Os atributos das tabelas, embora se refiram a informações similares, possuem apresentação bastante distintas e por essa razão dificultam a identificação de pares sem um processamento prévio das informações.
2. REQUISITOS DA SOLUÇÃO
   1. Eficácia
   2. Legibilidade
   3. Reusabilidade
   4. Eficiência
   5. Escalabilidade
3. ESTRATÉGIAS DE SOLUÇÃO
   1. Construção do Modelo de Dados
   2. Normalização da tabela PMB/IMS
   3. Pré-processamento das tabelas SAMMED
   4. Estrutura de diretórios e arquivos
   5. Identificação de pares de substâncias
4. RELATÓRIO PARCIAL DE IMPLEMENTAÇÃO
5. **CONSTRUÇÃO DO MODELO DE DADOS**

Esta primeira atividade compreendeu a análise das tabelas ABCFARMA, SAMMED, PMB/IMS e HMB/IMS para a construção de um modelo de dados que relaciona e descreve os atributos principais de cada tabela. Este modelo de dados pode ser encontrado no arquivo *IPEA - Modelo de dados - Tabelas Medicamentos.pdf.*

1. **NORMALIZAÇÃO DA TABELA PMB/IMS**

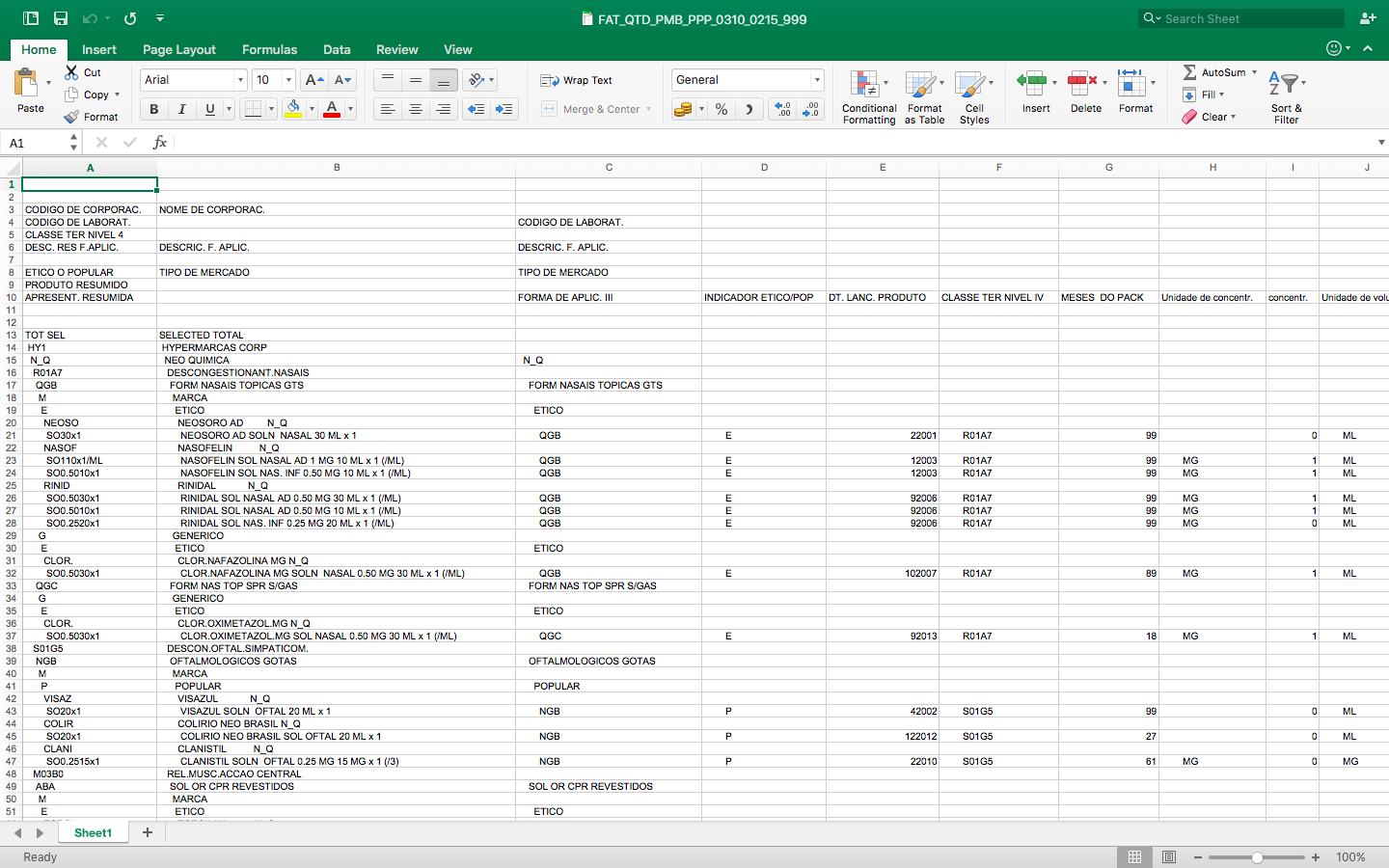
Quando se trabalha com um projeto de banco de dados relacional, é ideal que se tenha um projeto que armazena cada item de dados lógico em apenas um lugar no banco de dados. Essa prática é conhecida como normalização de dados, e garante consistência e economia de espaço de armazenamento. Algumas vezes, no entanto, é necessário usar a redundância controlada para melhorar o desempenho das consultas, pois colocando todos os dados juntos não precisamos pesquisar vários arquivos para coletar esses dados - isso é conhecido como desnormalização. [1]

No contexto deste projeto, não se está tratando com um banco de dados relacional em si, mas com um conjunto de bases de dados para as quais se deseja justamente encontrar suas relações, pois estas bases não apresentam atributos do tipo chave estrangeira que as relacione diretamente. O que se pretende, portanto, é utilizar técnicas de processamento de linguagem natural para inferir uma relação do tipo “de-para” entre os identificadores únicos das tabelas; para tanto, as diversas tabelas do conjunto de dados serão acessadas e consultadas através de scripts Python.

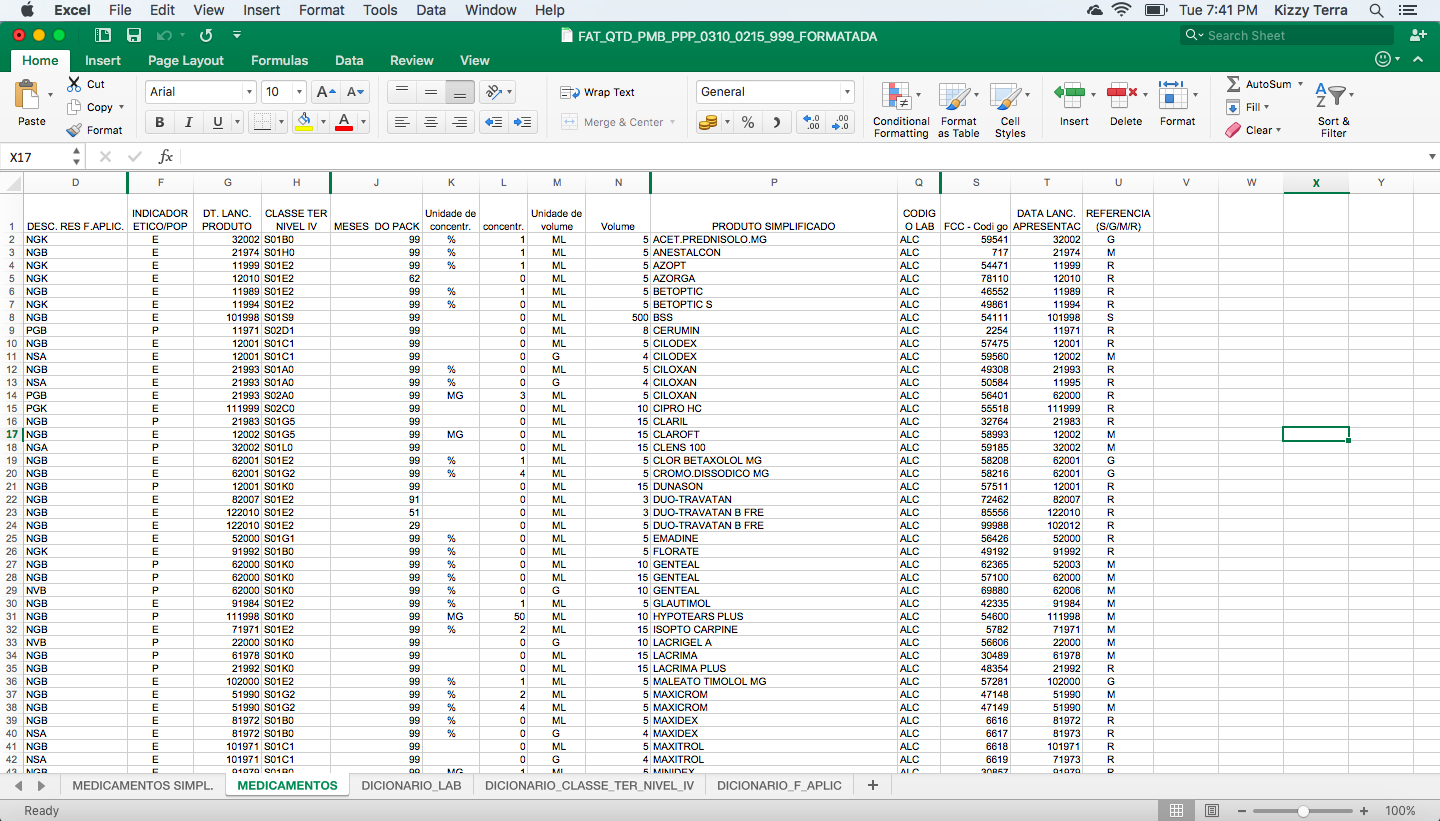
No entanto, ainda que não se esteja lidando com um banco de dados relacional na sua forma tradicional, os processos de normalização e desnormalização são igualmente convenientes e possibilitam um aumento na eficiência das implementações, uma vez que com eles é possível eliminar redundâncias desnecessárias e preservar ou inserir redundâncias úteis.

O processo de normalização realizado nos dados é diretamente dependente da disposição destes dados e por esse motivo, para cada tabela que se desejar normalizar é necessário fazer modificações específicas. Uma vez que as tabelas fornecidas se encontravam no formato de planilhas do Microsoft Excel, a normalização foi feita utilizando as próprias ferramentas de filtro e funções de agregação do programa.

Na tabela PMB/IMS as principais mudanças feitas a serem destacadas foram na disposição das informações que estavam em linhas e não e nas colunas, por terem sido extraídas neste formato, bem como a remoção do código do laboratório da coluna de descrição das substâncias.



**Figura 1 – Tabela PMB original**



**Figura 2 – Tabela PMB normalizada**

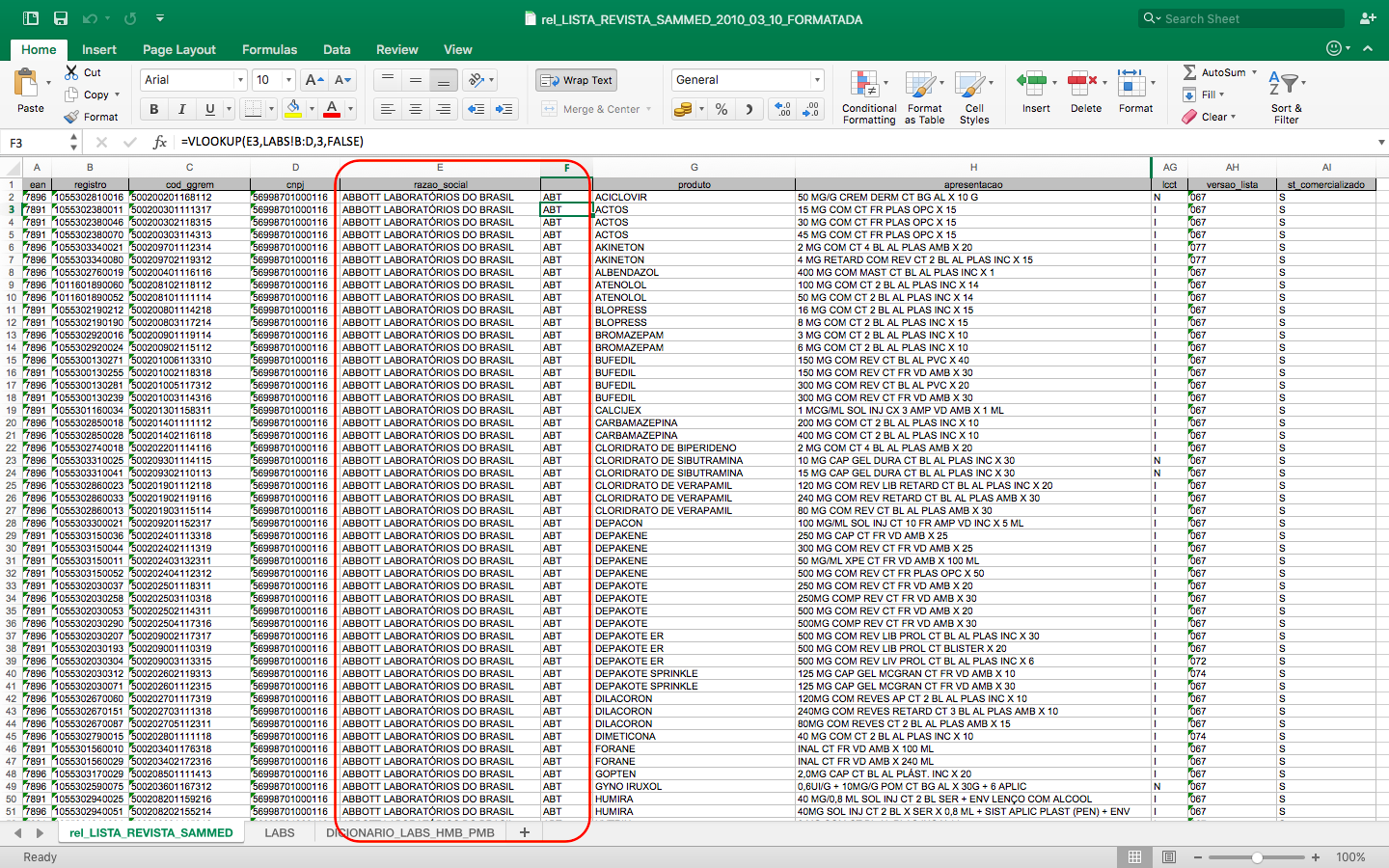
1. **PRÉ-PROCESSAMENTO DAS TABELAS SAMMED**

Nas tabelas da SAMMED foi realizado um pré-processamento simples na coluna de laboratórios para facilitar a comparação com as tabelas da IMS (PMB e HMB). As tabelas PMB e HMB possuem ambas, além um atributo com o nome do laboratório, um atributo com um código para o laboratório o qual é uma *string* de exatamente três letras.

Tendo identificado isto, foi possível gerar uma tabela com nomes e códigos de laboratórios únicos – totalizando 368 laboratórios - (figura 3) e utilizá-la para associar códigos aos respectivos laboratórios nas tabelas da SAMMED.

As tabelas da SAMMED relacionam aproximadamente 270 laboratórios e foi possível associar códigos a 182 destes (cerca de 67%). A listagem de laboratórios encontra-se no arquivo *laboratorios.xlsx* .

Este processamento contribui para a organização dos dados na estrutura de arquivos e diretórios cuja descrição será detalhada em seguida.



**Figura 3 – Tabela SAMMED com códigos de laboratório**

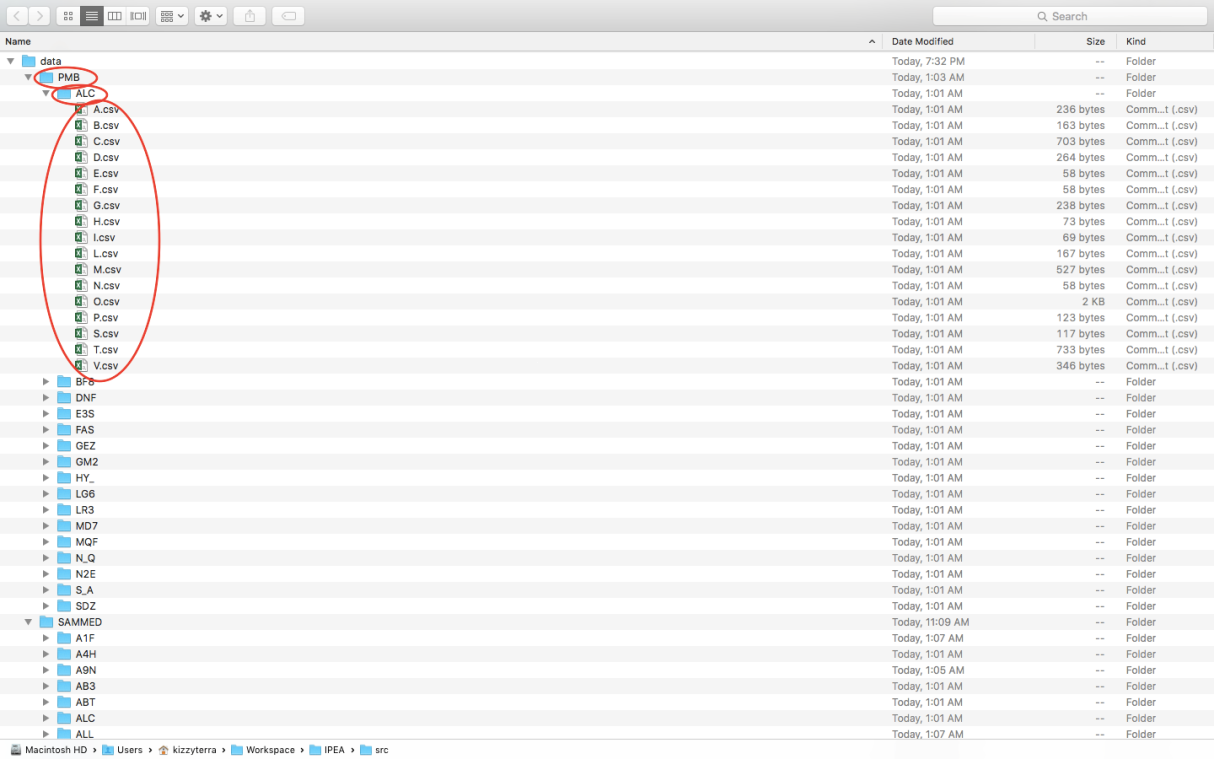
1. **ESTRUTURA DE DIRETÓRIOS E ARQUIVOS**

A estruturação de um conjunto de dados em diretórios e arquivos que facilitem o acesso e busca de informações é uma técnica bastante utilizada na área de recuperação de informação, a exemplo dos [arquivos invertidos](http://dns.uls.cl/~ej/daa_08/Algoritmos/books/book5/chap03.htm) que são empregados nos casos em que se almeja fazer consultas por termos ou expressões em grandes documentos e arquivos de textos.

Neste projeto, por sua vez, as informações são acessadas, em cada tabela, por atributo (coluna da tabela) através da estrutura de dados *DataFrame* disponível no pacote Pandas do Python e por consequência, a disposição dos dados em arquivos invertidos não é a mais apropriada. Porém, é possível organizar os dados em uma estrutura de diretórios e arquivos mais pertinente para o contexto.

A hierarquia de diretórios e arquivos elaborada pode ser facilmente compreendida observando-se a figura 4. Nela é possível notar que existe um diretório raiz denominado *data* que contém diretórios correspondentes às diversas fontes de bases de dados de medicamentos (PMB/IMS e SAMMED, no caso da figura). Mais internamente, no último nível da hierarquia encontram-se arquivos do formato *.csv* denominados com as letras do alfabeto. Estes arquivos armazenam as informações de medicamentos cuja substância se inicia com a letra que dá nome ao arquivo, facilitando a busca e a comparação entre possíveis pares de medicamentos.

O código-fonte da implementação deste processamento pode ser encontrado no arquivo *preprocessing.py* que se encontrano diretório *src*.



**Figura 4 – Estrutura de Diretórios e Arquivos**

1. **IDENTIFICAÇÃO DE PARES DE SUBSTÂNCIAS**

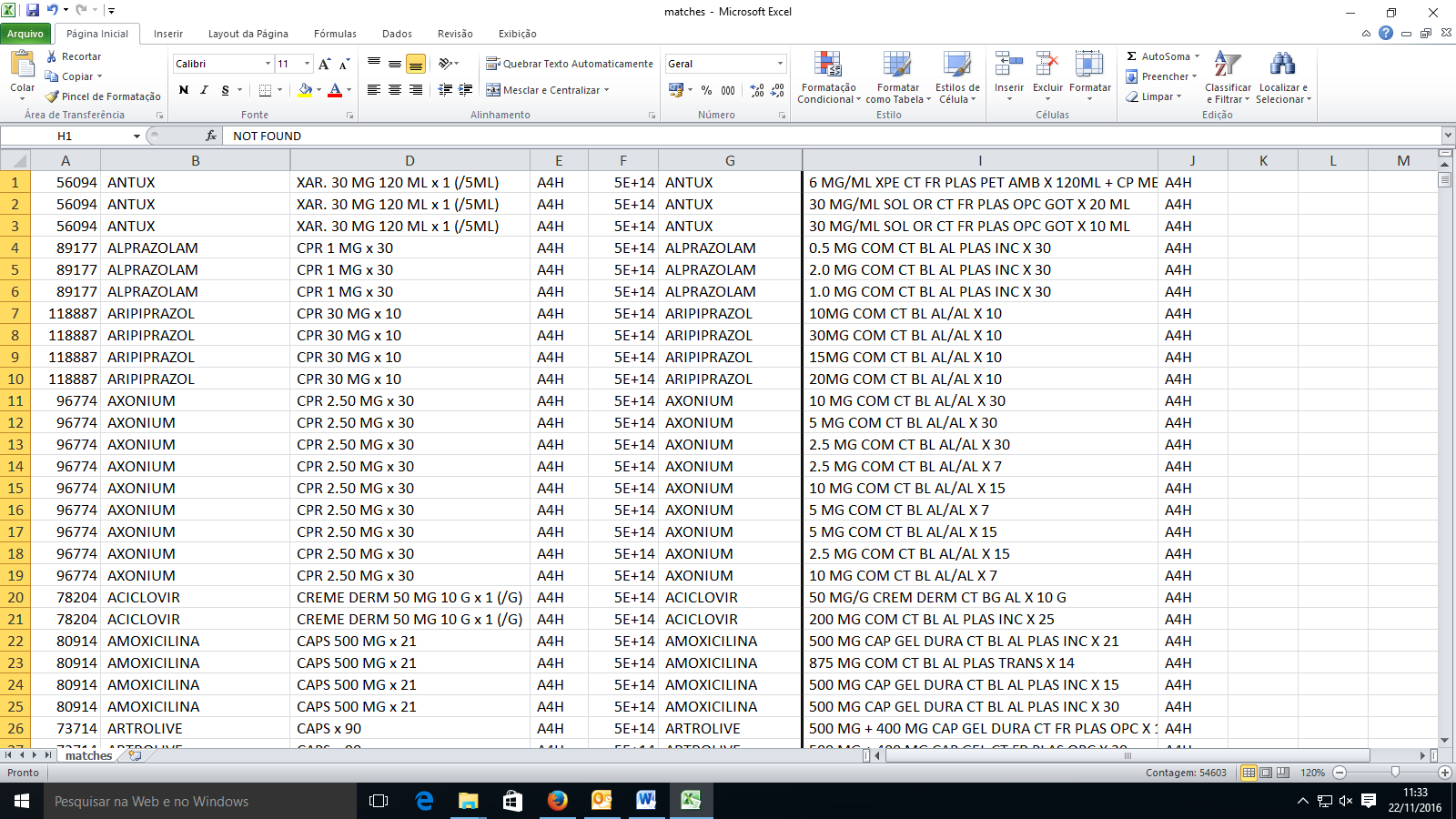
O processo de identificação de pares será implementado por etapas. Iniciando com a identificação dos laboratórios semelhantes utilizando a estrutura de diretórios para identificar e comparar os medicamentos correspondentes de cada laboratório iniciados com a mesma letra do alfabeto.

Seguindo esta etapa, serão comparados entre si os atributos *produto* e *apresentação* das tabelas, respectivamente. Isto é, através do pareamento do atributo *produto* serão identificados os medicamentos candidatos a pares e então, finalmente, encontrar os pares finais comparando o atributo *apresentação*. Para tanto, será necessário a implementação de métodos específicos para o pareamento de cada um destes atributos.

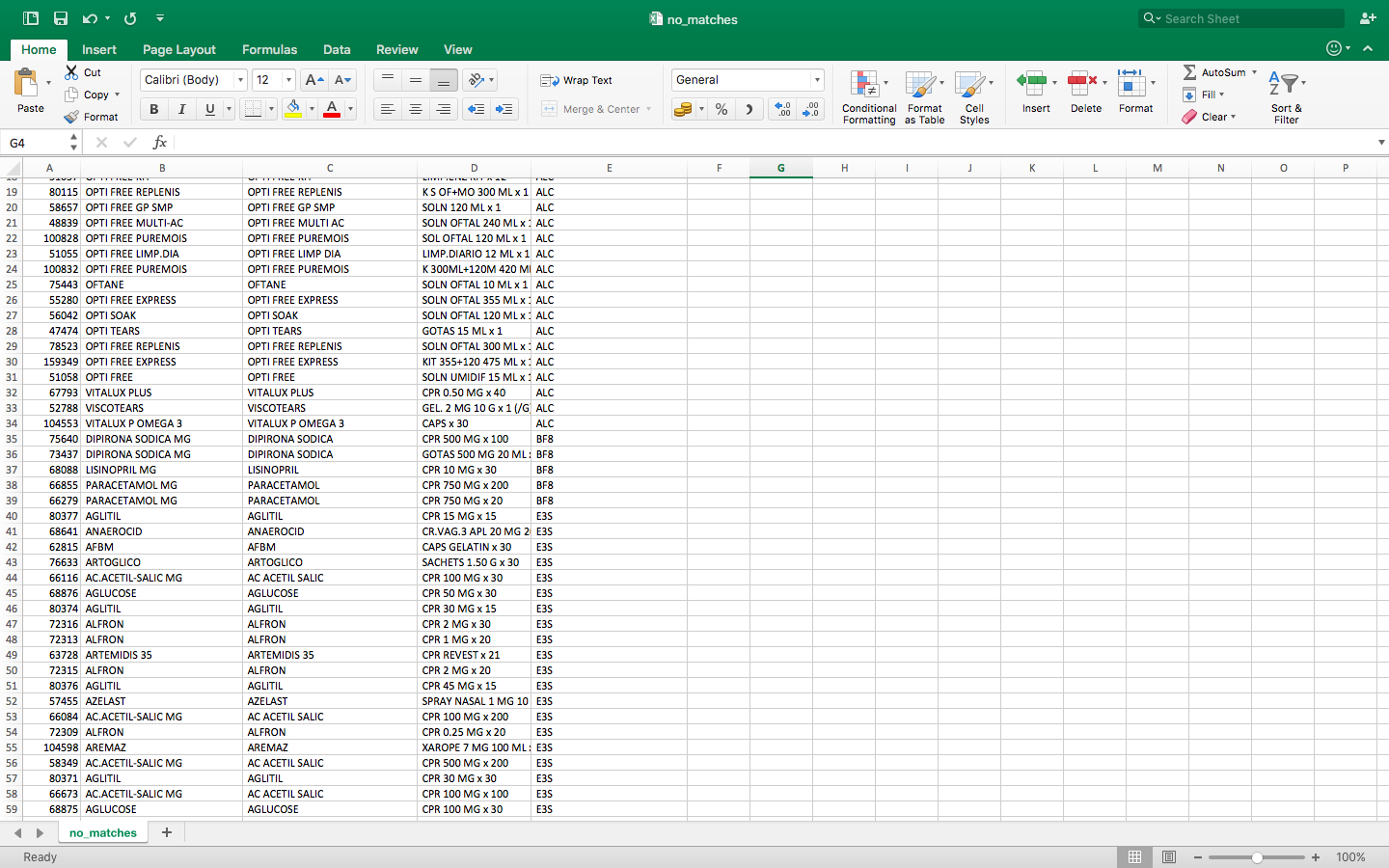
* 1. Pareando pelo atributo *produto*

Este pareamento é feito através da comparação das substâncias presentes em cada arquivo *.csv* de cada diretório de laboratório. Faz-se a leitura dos arquivos linha a linha; em seguida recupera-se o campo correspondente ao atributo *produto* e utiliza-se um [método de identificação de abreviações](http://stackoverflow.com/questions/7331462/check-if-a-string-is-a-possible-abbrevation-for-a-name) (código-fonte no arquivo *utils.py*) para identificar se os medicamentos são pares prováveis. Os detalhes podem ser analisados no método *product\_matches* do arquivo *classifier.py* que se encontrano diretório *src*. É interessante ressaltar que para uma melhor acurácia do processo é feita uma remoção de *stopwords* do nome das substâncias, para então fazer a identificação de possíveis pares.

O pareamento pelo atributo *produto* gera como saída arquivos de *matches* (figura 5)e *no\_matches* (figura 6)os quais indicam os medicamentos com e sem pares candidatos. No primeiro teste do algoritmo implementado foram encontrados matches para 3476 substâncias e não foram encontrados matches para 1787 substâncias.



**Figura 5 – “Matches” - Pareamento pelo atributo *produto***

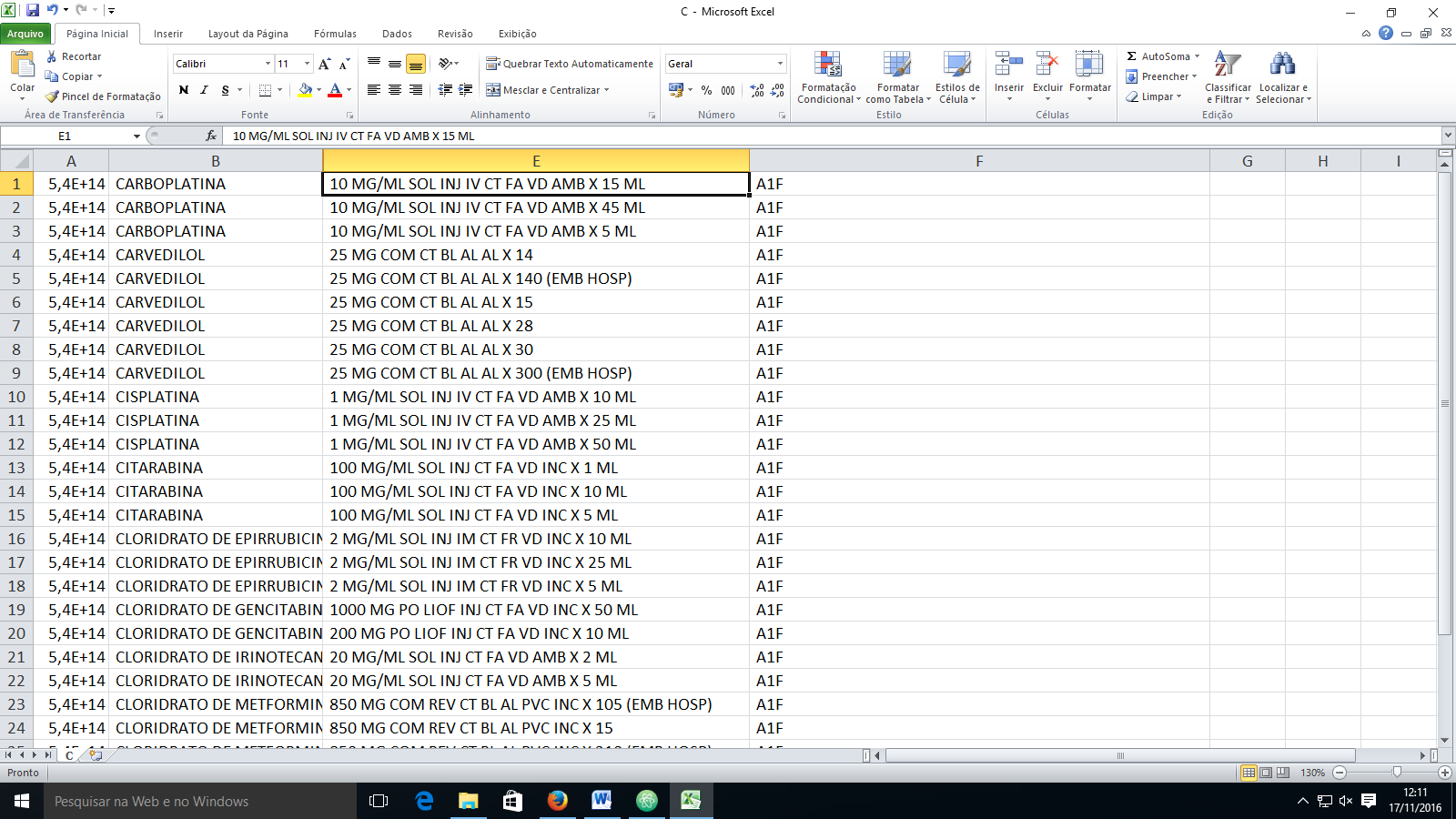


**Figura 6 – “No Matches” - Pareamento pelo atributo *produto***

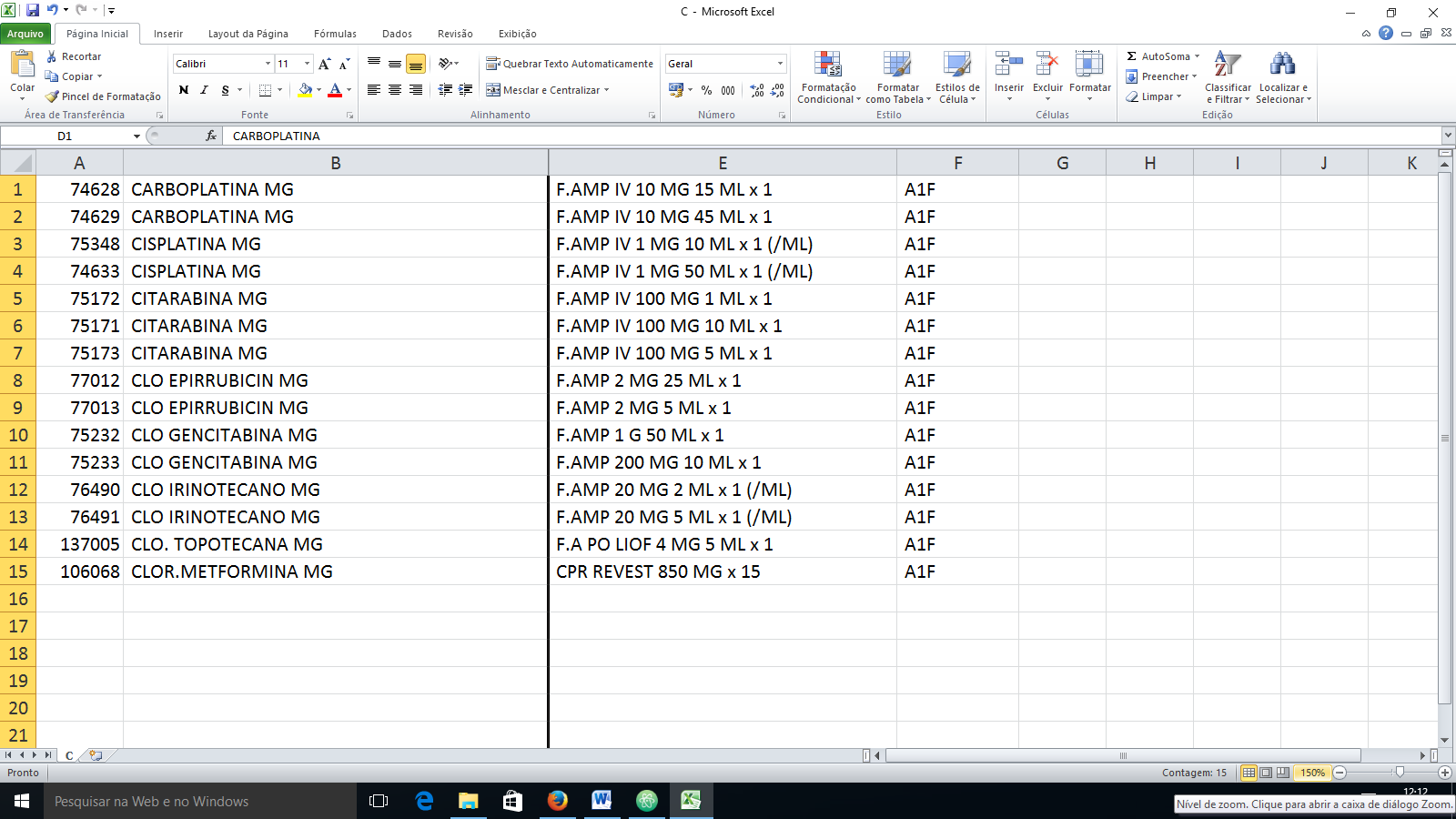
* 1. Pareando pelo atributo *apresentação*

Nesta etapa de pareamento de medicamentos pelo atributo *apresentação* filtra-se do arquivo *matches.csv* (figura 5)que possui os possíveis pares *-* obtido a partir do pareamento pelo nome das substâncias – aqueles que possuem apresentação similar.

Cabe ressaltar que o atributo apresentação dos medicamentos contém, em geral, as seguintes informações: forma farmacêutica, concentração, quantidade, volume. No entanto, cada fonte de dados (IMS, SAMMED) dispõe estas informações de uma maneira diferente. É possível encontrar diferentes abreviações para as mesmas formas farmacêuticas, bem como a ordem em que as informações estão no campo também pode variar.



**Figura 7 – Exemplo de atributo apresentação de medicamento na tabela SAMMED**



**Figura 8 – Exemplo de atributo apresentação de medicamento na tabela PMB**

Em virtude destas variações, expressões regulares foram utilizadas para auxiliar na classificação dos medicamentos, identificando possíveis padrões do atributo *apresentação* para cada fonte de dados e assim recuperar as informações necessárias e compará-las para identificar pares de substâncias similares.

**Lista das expressões regulares identificadas**

Uma expressão regular é uma notação para representar **padrões** em sentenças e serve para validar entradas de dados ou fazer busca e extração de informações em textos. [2]As expressões regulares mapeadas neste contexto buscam, principalmente, identificar dígitos e expressões que contém palavras-chave próximas a dígitos.

1. Filtrar número inteiro seguido de G ou MG

r"\s?(\d+)\s?M?G"

1. Filtrar número inteiro ou fracionário seguido de MG/G ou MG/L ou ML/L

r"\s?(\d+\.\d\*)\s?MG?L?/G?L?"

1. Filtrar número fracionário seguido de G ou MG

r"\s?(\d+\.\d\*)\s?M?G"

1. Filtrar número inteiro antecedido de x ou X e seguido de G ou MG

r"x?X?\s?(\d+)\s?M?G"

1. Filtrar número inteiro antecedido de x ou X e seguido de L ou ML

r"x?X?\s?(\d+)\s?M?L"

1. Filtrar número inteiro antecedido de CT ou C/

r"CT\s?C?/?\s?(\d+)"

1. Filtrar número fracionário seguido de BLT

r"\s?(\d+)\s?BLT"

1. Filtrar número inteiro antecedido de C/

r"\s?C/\s?(\d+)"

1. Filtrar número inteiro maior do que 2 antecedido de x ou X no final da expressão

r"X\s?([1-9]\d+|[2-9]$)"

1. Filtrar número 1 antecedido de x ou X no final da expressão

r"x?X?\s?1$"

1. Filtrar número 1 antecedido de x ou X e seguido de espaço

r"x?X?\s?1\s"

1. Filtrar expressão composta de um sinal + e um número

r"[+]\s?\d+"

1. Filtrar número

r"\s?(\d+)"

**REFERÊNCIA BIBLIOGRÁFICA**

[1] ELMASRI, RAMEZ. Sistemas de Banco de Dados/ Ramez Elmasri e Shamkant B. Navathe; tradução Daniel Vieira; 6. Ed. – São Paulo: Pearson Addison Wesley, 2011.

[2] RAMALHO, LUCIANO. Expressões regulares: introdução, 2012. Disponível em: http://turing.com.br/material/regex/introducao.html

[3] PYTHEX. Ferramenta de teste de expressões regulares Python. Disponível em: <http://pythex.org/>

**APÊNDICE**

1. **Construção do dicionário de laboratório**
2. **Construção de um interpretador para o campo de apresentação dos medicamentos**