Documentação do MD Smart Console

transcrição das informações de cada script usado na criação do app.

Ricardo da Hora

2023

Índice de figuras

[Figura 1- Scripts do MD Smart Console. 3](#_Toc140040891)

[Figura 2- tela de início do app. 4](#_Toc140040892)

[Figura 3-login 5](#_Toc140040893)

[Figura 4-Planilha de acesso. 5](#_Toc140040894)

[Figura 5- Permissões do app domino. 6](#_Toc140040895)

[Figura 6-Diagrama de comunicação dos módulos. 6](#_Toc140040896)

[Figura 7-Página de doenças de milho. 8](#_Toc140040897)

[Figura 8-Salvar. 9](#_Toc140040898)

[Figura 9-Página do Corn note. 10](#_Toc140040899)

[Figura 10-Corn note Preenchida com atualização do iframe. 12](#_Toc140040900)

[Figura 11 - DREC\_NREC 13](#_Toc140040901)

[Figura 12-Página de Gestão de produtos. 14](#_Toc140040902)

Sumário

[**1.** **Objetivo** 3](#_Toc140041347)

[**2.** **MD Smart console componentes** 3](#_Toc140041348)

[**3.** **Script views** 4](#_Toc140041349)

[**4.** **Script doencas** 7](#_Toc140041357)

[4.1. Função editar. 9](#_Toc140041360)

[4.2. Função Salvar. 9](#_Toc140041361)

[4.3. Função atualiza(tabela) 9](#_Toc140041362)

[4.4. Função carrega\_valor(tab4) 9](#_Toc140041363)

[4.5. Função filtra (p, b, pi, m, tabela3) 10](#_Toc140041364)

[**5.** **Script note** 10](#_Toc140041365)

[5.1. Função page1 10](#_Toc140041366)

[5.2. Função Page3 11](#_Toc140041367)

[5.3. Função filtra1(p,b,pi,tabela13) 13](#_Toc140041368)

[5.4. Função atualiza1 13](#_Toc140041369)

[5.5. Função adicionaitem(cb4,story1, tabela10c, info1) 13](#_Toc140041370)

[**6.** **Funções da página gestão de produtos que estão nos scripts note e arquivo.** 14](#_Toc140041371)

[6.1. Envio do RM/Fenotype - post\_arquivo(mais o prefixo do país)1. 14](#_Toc140041372)

[6.2. Botão Salvar de dados RM/Fenotype - Função salvar (prefixo do país)4. 15](#_Toc140041373)

[6.3. Envio de dados da Gestão de Produtos -post\_arquivo(prefixo do país)2. 15](#_Toc140041374)

[6.4. Botão Salvar de dados da Gestão de Produtos -Função salvar (prefixo do País) 5. 15](#_Toc140041375)

[6.5. Envio de dados de Corp Monitoring - post\_arquivo(prefixo do País) 3. 16](#_Toc140041376)

[6.6. Botão Salvar de dados de Corp Monitoring -função Salvar(prefixo do País) 6 16](#_Toc140041377)

[6.7. Envio de dados de Corne note -post\_arquivo(prefixo do País) 4 16](#_Toc140041378)

[6.8. Botão Salvar de dados de Corne note -função Salvar (prefixo do País) 7. 17](#_Toc140041379)

[6.9. Planilhas de exemplo ou baixa de arquivo - Função get\_arquivo. 17](#_Toc140041380)

[**7.** **Atualiza\_banco.** 18](#_Toc140041381)

[7.1. função Stability. 18](#_Toc140041384)

[7.2. função m\_b. 19](#_Toc140041385)

[7.3. função e a phentype. 19](#_Toc140041386)

[7.4. função e a h2h 21](#_Toc140041387)

[7.5. funções fitradoencas , notadohibrido e nderc 21](#_Toc140041388)

[7.6. função agronomicrick 21](#_Toc140041389)

[7.7. função e a geraplanilha 22](#_Toc140041390)

[**8.** **Anexos** 23](#_Toc140041391)

1. **Objetivo**

O objetivo do MD Smart console é de ser uma ferramenta onde o usuário possa colocar os dados que necessitam de uma revisão previa, antes de serem entregues a o banco de dados (Lake) e tem o papel juntamente com o script de atualiza\_banco de filtrar e automatizar a entrega dos dados em uma única saída para ser consumido pelos aplicativos das marcas (que a Bayer representa em cada país).

O script de atualiza banco tem como papel principal filtrar, calcular e formatar os dados para serem entregues ao time de KT, por meio de acesso ao banco de dados como também em formato de planilha.

1. **MD Smart console componentes**

O MD smart console está dividido em scripts que rodam de forma agendada uma atualização dos dados de cada país (Conosur, Mexico em Norla e Brasil). Conta também com scripts com funções para cada página e pais. A função que é responsável por gerenciar todas as chamadas de página interconectar todas as funções dos está no script views.

Scripts que compõem hoje o MD smart console. Views, argdoencas, argarquivo, argnote, brdoencas, brarquivo, brnote, mxdoencas, mxarquivo e mxnote, na parte da interface ainda conta com os scripts atualiza\_bancoarg, atualiza\_bancobr e atualiza\_bancomx.



Figura - Scripts do MD Smart Console.

Os scripts de atualiza\_banco são responsáveis por extrair, filtrar e calcular os dados que vem do Lake e depois transformar os mesmo em dados para serem consumidos pelos aplicativos das marcas e gerar a planilha de final de MD (ela é padronizada para todos os países para terem os dados necessários para entrega para a Magoya).

Já os scripts doencas, arquivo e note tem a função de interagir com os códigos de páginas HTML´s que fazem a interface gráfica para o usuário juntamente com o views.

Descrevendo os scripts e páginas da interface para do usuário.

Começando pelo Script views ele e chamado por uma função que inicia a partir da plataforma Domino. Para criar esta inicialização utiliza se um script pré-definido pelo domino na página a [Publish Apps (dominodatalab.com)](https://docs.dominodatalab.com/en/5.3/user_guide/71635d/publish-apps/) , que tem todo o passo a passo como criar o script para rodar a base do app.

Após seguir o passo a passo acesse ao menu do domino item App, e pressione start.

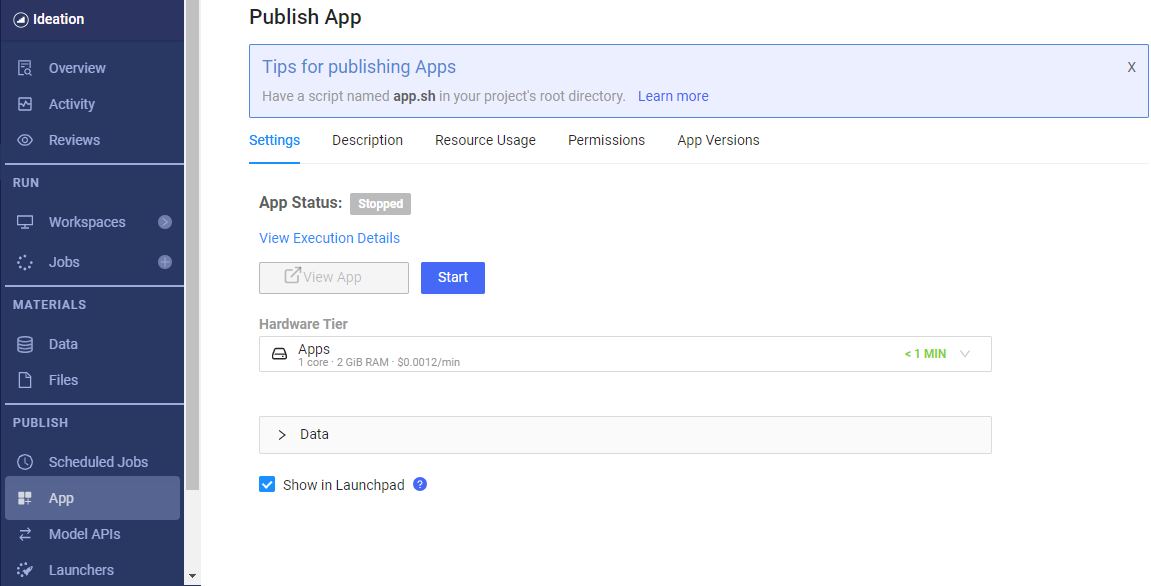


Figura - tela de início do app.

1. **Script views**

O script views e responsável por conectar os outros scripts além de gerenciar o acesso dos usuários ao aplicativo e suas permissões de acesso a cada página.

Como primeiro passo o script carrega as bibliotecas necessárias ,instancias , scripts e funções do MD Smart console.

Após a carga dos itens mencionados o sistema carrega a página **iniciar1.html** figura abaixo:

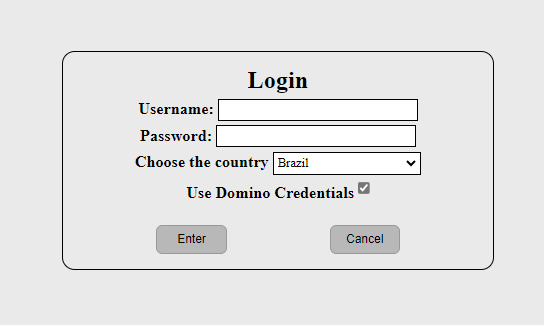


Figura -login

Ela composta por um quadro onde o usuário escolhe se colocara seus dados ou utilizará as credenciais do Domino, qual país irá fazer as alterações, lembrando que o acesso a página do país e condicionado ao usuário ter acesso registrado no arquivo acesso.xlsx do sistema que é verificado pela função iniciar(script views), se usuário caso tenha acesso será direcionado para a pagina que o mesmo pode fazer as alterações ou consultas necessárias, caso não tenha acesso será mostrada uma mensagem para que ele verifique se o acesso lhe é permitido, caso não seja e necessário entrar em contato com o responsável pelo Console.

Segue o exemplo de como deve ser colocas as permissões para cada usuário, lembrando também que será necessário que se de permissão no domino também. Como no exemplo da figura 4.

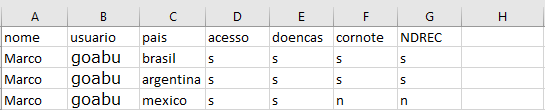


Figura -Planilha de acesso.

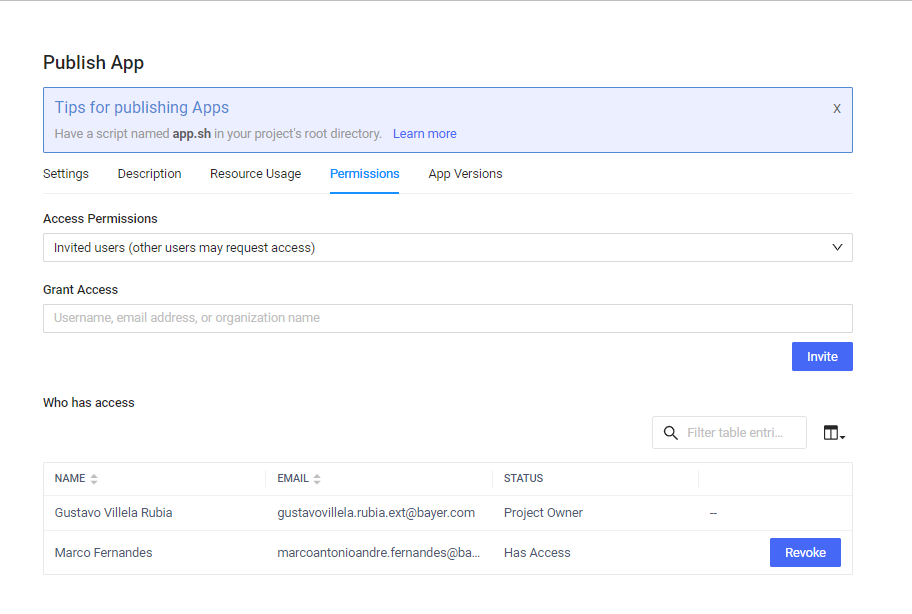


Figura - Permissões do app domino.

A partir da validação do usuário ele e enviado para a página inicial que pode ser a de doenças de milho no país que ele tem acesso ou a página de Nitrogênio e densidade.

Na figura 6 temos qual é o papel do script Views que funciona como um gateway para direcionar os Posts e Gets de cada uma das páginas que compõem o MD Smart console.

Página de doenças de milho

Página de Corn note

Views

Gestão de Produtos

DERC\_NREC

Planilhas de saída e exemplo

Atualiza banco

Lake

Figura -Diagrama de comunicação dos módulos.

1. **Script doencas**

Brdoncas, Argdoencas e mxdoencas são scripts que direcionam o acesso as páginas de notas de doenças de cada país(ver figura 7), onde o usuário pode verificar as doenças de milho e alterar a nota para cada tipo de doença, selecionando o híbrido por marca, biotecnologia(há casos como no Mexico que não se usa nos produtos a biotecnologia), nome do produto e Pipeline. A página que gera a interface para o usuário e a homepage seguida pelo prefixo do país com exceção do Brasil, ar (argentina) ou m (Mexico). E que faz as solicitações dos seus dados para a função que tem o mesmo nome com o prefixo no final br (Brasil), arg (Argentina) ou mx (Mexico).

OBS: todos os scripts ficam instalados na pasta Flask\_APP.

Como todas as funções têm o mesmo princípio de funcionamento e para não ser repetitivo vou explicar uma pois todas seguem o mesmo padrão.

Primeiramente são carregadas as bibliotecas necessárias para a inicialização do script e direciona o sistema para a pasta que contém os templates e arquivos que configuram as páginas HTML os mesmos carregam os dados que iniciam as variáveis que serão carregadas com informação de data e hora, tabelas e variáveis de inicialização global.

Carregados os componentes inicializa se as funções especificas para a tela de doenças e manipulação destes dados.

As funções como os scripts seguem com o prefixo do país(ex: homepagebr) são : Homepage, editar, salvar. E outras que tem o mesmo nome em qualquer um dos scripts de doenças são as : atualiza(tabela), carrega\_valor(tab4) e filtra(p, b, pi, tabela3).

Detalhando as funções especificas dentro do script:

Homepage e a responsável por carregar a página inicial de doenças para o MD Smart Console.

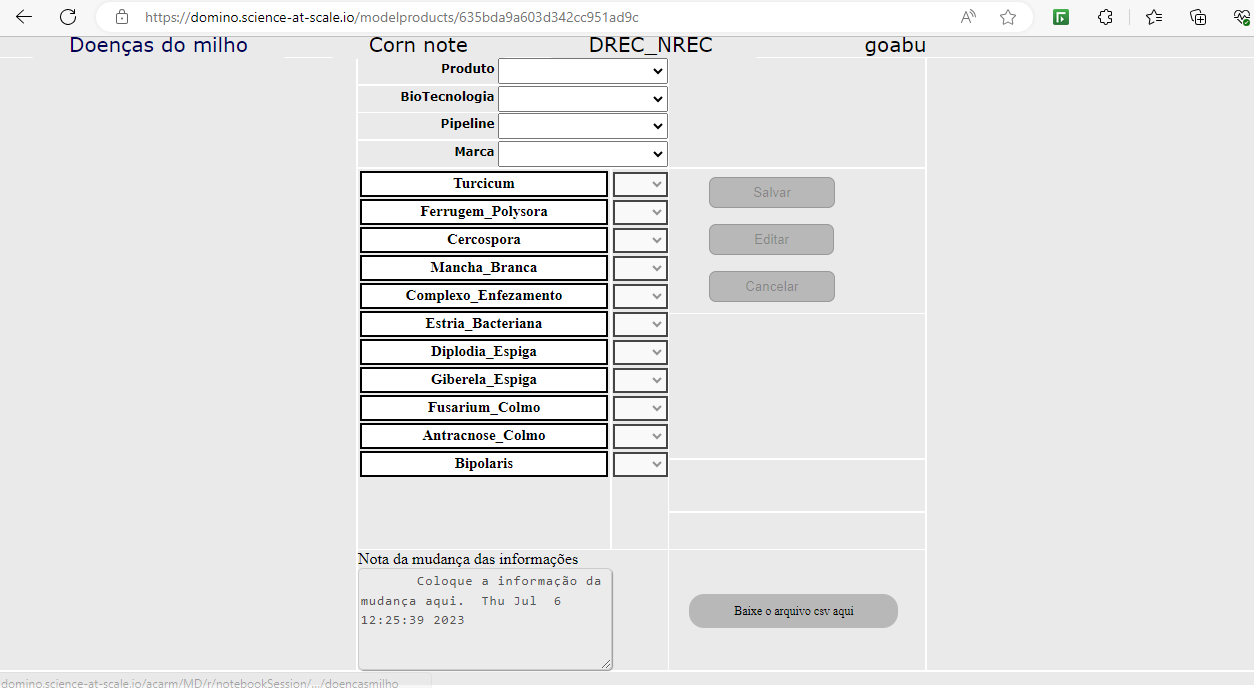


Figura -Página de doenças de milho.

Inicia se carregando a variável de data e hora com os dados do sistema, logo depois são carregados os dados como o nome da usuário que está logado e que será mostrado na página de HTML doencasmilho (mais prefixo do país), carrega-se também variáveis com os dados (que serão usados na página) como as tabelas que contém as base das informações e outras que terão os dados alterados . É realizado um filtro dos dados que não são mais validos mas que compõem um histórico do que foi alterado classificados como de Status old. Coloca-se os dados em ordem (por nomes de produtos ) separa-se os nomes das colunas que contêm os nomes das doenças das que tem outros dados( os mesmo servirão para preencher a tabela dentro da html como na figura 7).

O sistema verifica se a página está solicitando dados(GET) ou enviado dados (POST), no caso de solicitação ou a primeira vez que a tela e iniciada, são apenas carregados os dados default como células vazias , também e feito um pré-filtro usados as funções de atualiza e carrega\_valor dos dados como também de variáveis que desabilitam botões e demais funções da página.

Caso a página a esteja enviado dados(POST), estes passam por outra verificação, se é para atualizar os combox ou se eles já estão preenchidos(habilitando assim o botão de editar). Este processo retorna as alterações até que os combos estejam todos preenchidos com dados.

Caso estes estejam preenchidos e liberado o botão de editar e carregados os dados dos combos de notas, com as notada de doenças do milho( estas podendo ser números ou letras a depender do País). As informações que serão mostradas na tela são do arquivo em csv que fica no sistema e são acessados pelo MD Smart Console.

Se o usuário pressionar o botão de Editar o sistema inicia a função de mesmo nome que tema como função desabilitar os combos de dados do produto e habilitar os combos de notas de doenças. O script fica aguardando um novo post com as alterações das notas das doenças, sendo alteradas as informações são atualizadas, até que seja pressionado o botão de salvar que irá iniciar um outro tratamento dos dados(como veremos no item específico da função) . No script são geradas duas variáveis uma com os dados atuais(sem alteração) e outra com os dados alterados (novos).

2. 1. Função editar.

A função trata apenas os dados que forme enviado por meio de Post (da página). Basicamente ela e uma função auxiliar para função homepage e serve apenas para carregar os dados e dar as liberações para edição das notas de doenças.

* 1. Função Salvar.

Carrega a tabela que ainda não está atualizada para ser trabalhada com os dados que virão da função Homepage, primeiramente a função realiza uma filtragem do dado no índice que vem da página homepage, gera uma nova variável ( data frame ) que terá sua a coluna Status alterada para old e em outra variável(data frame) e salva a tabela inteira com a exceção desta linha. Realiza-se a inclusão dos dados alterados em uma nova linha e depois são inseridos os dados da variável que foi alterada para old.

Ordena-se os dados pelo nome do produto e depois salva-se um arquivo .csv que fica como backup do sistema e é enviado sem os dados old´s a nova tabela para o Lake com o nome de Disiases\_table.

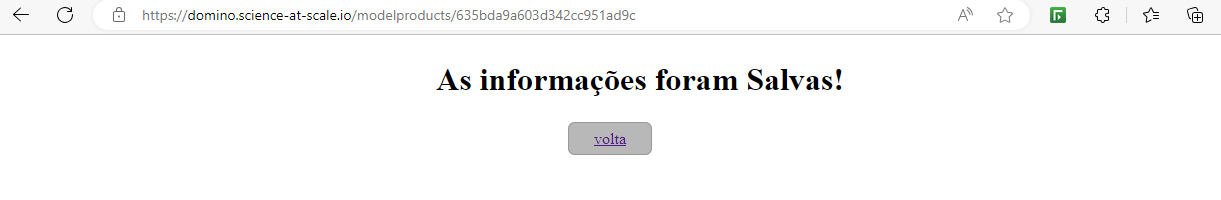


Figura -Salvar.

* 1. Função atualiza(tabela)

Tem como função fazer um filtro de dados para atualizar as informações de saída para a página homepage de produto, marca, pipeline e biotecnologia (caso tenha no país).

* 1. Função carrega\_valor(tab4)

Atualiza os dados das notas de doenças que estão na tabela do arquivo de doenças e faz o filtro pelos dados passados das variáveis que vem da função atualiza ou filtra. E carrega os dados para as variáveis da página de doencasmilho.

* 1. Função filtra (p, b, pi, m, tabela3)

Como o próprio nome diz tem a função de filtrar os dados de uma tabela, a partir das variáveis de entrada p, b, pi, m e tabela3, que são respectivamente produto, biotecnologia, pipeline, marca e a variável que contêm as informações do arquivo de doenças.

1. **Script note**

Brnote, Arnote e mxnote são scripts para acesso as notas de características dos híbridos, dados dos refúgios que são indicados e as notas de recomendações de manejo de cada país. Nesta página e possível escolher qual híbrido, pipeline e biotecnologia (exceto no Mexico) e na área lateral aparecera todas as informações do produto em forma de um documento impresso. Como também campos de alteração de suas informações com a exceção do nome, da região e biotecnologia(dados estes que só podem ser alterados no menu de gestão de produto).

* 1. Função page1

No script denominando page1(mais prefixo do país) primeiramente são carregados os locais onde se encontram os templates , arquivos de configuração da página e seus componentes, depois são carregadas as bibliotecas que são necessárias para o funcionamento das funções, bem como as variáveis que receberão os dados das planilhas e demais itens que serão alterados.



Figura -Página do Corn note.

São carregadas duas planilhas para a manipulação dos dados que são: notedescrip1( com o prefixo do país) e descricao1( que segue a mesma regra). A primeira tem os dados de características que foram cadastradas, e o segundo são exemplos de notas já usadas e que podem servir para uma maior agilidade na inserção de uma nota.

Depois os dados são filtrados e são retirados os de status olds ( que são notas alteradas e deixadas em um backup para registro). Também são carregados os dados defaults para as variáveis de página.

A primeira função e a page1(mais o prefixo do país) ela e a responsável por todas as interações na página Corn note. Primeiramente ela carrega a variável de data e hora para registro das informações, carrega também o nome do usuário e as tabelas que são filtradas e ordenadas pelo nome do híbrido.

E ela fica aguardando que venha uma requisição da página, seja ela um GET ou um POST, sendo um GET(uma requisição de carga das informações) ela inicia a carga default das variáveis da página , variáveis de manipulação, carrega os dados das colunas que serão usados nas alterações, utiliza-se das funções atualiza1 e filtra1 para carregar as informações no combos da página, carrega os dados da planilha que são mostrados no documento lateral, informações de itens desabilitados dentro da página e informações iniciais dos campos de notas.

Quando a página recebe um POST ( um envio de informação da página) o script carrega as variáveis globais e inicia a verificação de cada uma das variáveis que vem da página até que se tenha todas as variáveis(combos preenchidos) ela vai fazendo a filtragem das informações e carregando na página. Quando todos os dados são preenchidos e únicos(não pode haver dados iguais dentro das planilhas ) e mostrada na tela lateral as informações do híbrido e é liberada o botão editar (que inicia um outro script chamado de editar2(seguido do prefixo do país).

Ao se acionar o botão de Editar (e iniciada a função editar2) que traz para a página os dados alterados e habilitada o botão Salvar(que inicia uma função chamada de Salvar2(mais o prefixo do país), o botão continuar adicionando e são desabilitados os combos de escolha dos dados e habilitados os combos de alteração e nota.

Caso seja alterada alguma informação e acionar o botão continuar adicionando, as informações são alteradas na área lateral e mostradas para que ao final possam ser salvas na planilha e carregadas em definitivo no sistema. Detalhe todas estas informações alteradas, juntamente com as notas são salvas em variáveis para serem usadas na função Salvar2.

* 1. Função Page3

Função Page3(seguida do prefixo do país) receber as informações que vem da função page1, e mostrar os dados em forma de documento para a Página de Corn Note.

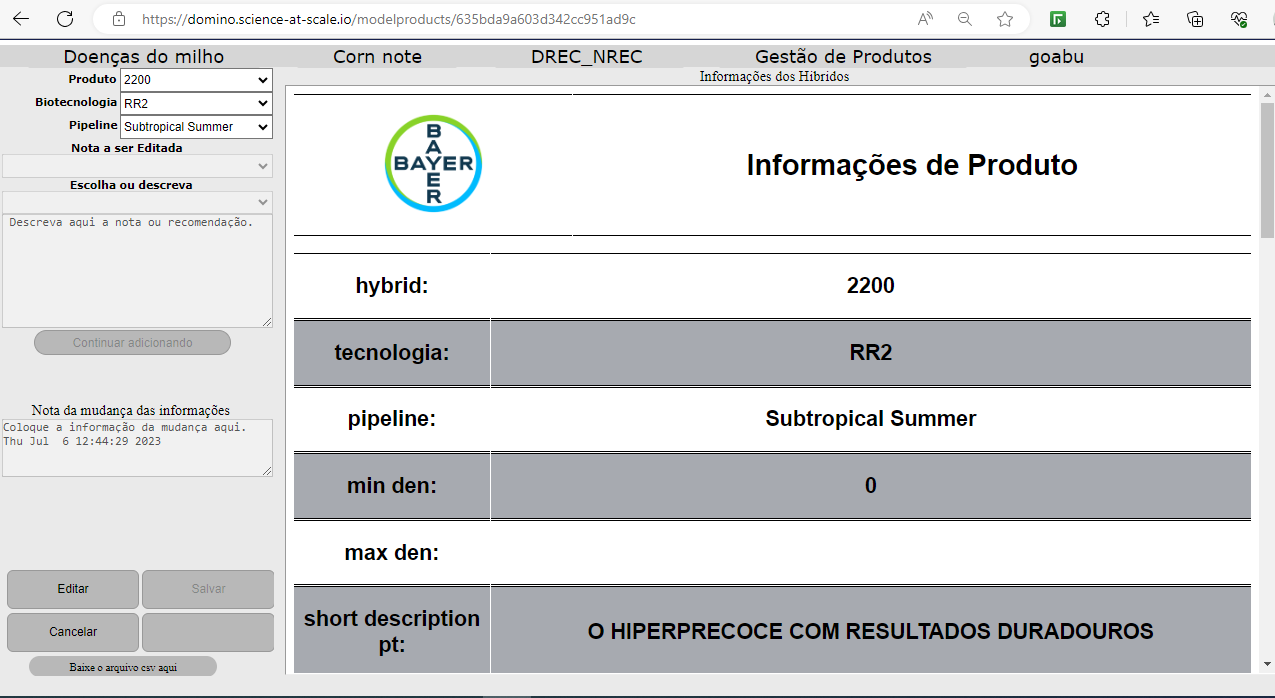


Figura -Corn note Preenchida com atualização do iframe.

Função editar2(mais prefixo do país) e iniciada a partir de uma solicitação(POST) que vem por meio do acionamento do botão Editar, e que verifica dos dados de todas as variáveis estão completas(combos preenchidos) e habilita os botões da página Corn note para edição das informações.

Função salvar2(mais prefixo do país) e iniciada a partir do acionamento do botão salvar da página Corn note. Que inicialmente carrega uma variável (data frame) com as informações da planilha corrente para gera um backup das informações da linha, que será alterada e coloca a informação no Status em old.

São Salvos os dados na linha correspondente a informação alterada pelo usuário e realiza-se uma junção das informações em uma única planilha, ordena se os dados pela coluna Hybrid , salvando as informações na planilha definitiva que será atualizada no Lake. Após o fim da execução do armazenamento e mostra uma mensagem que os mesmos forma alterados .

Na função page2(mais o prefixo do país ) ela monta a página onde e mostrada uma previa dos dados de nitrogênio e densidade, como também a possibilidade de baixar um exemplo com os dados atualizados e de inserir uma nova planilha com as alterações necessárias.

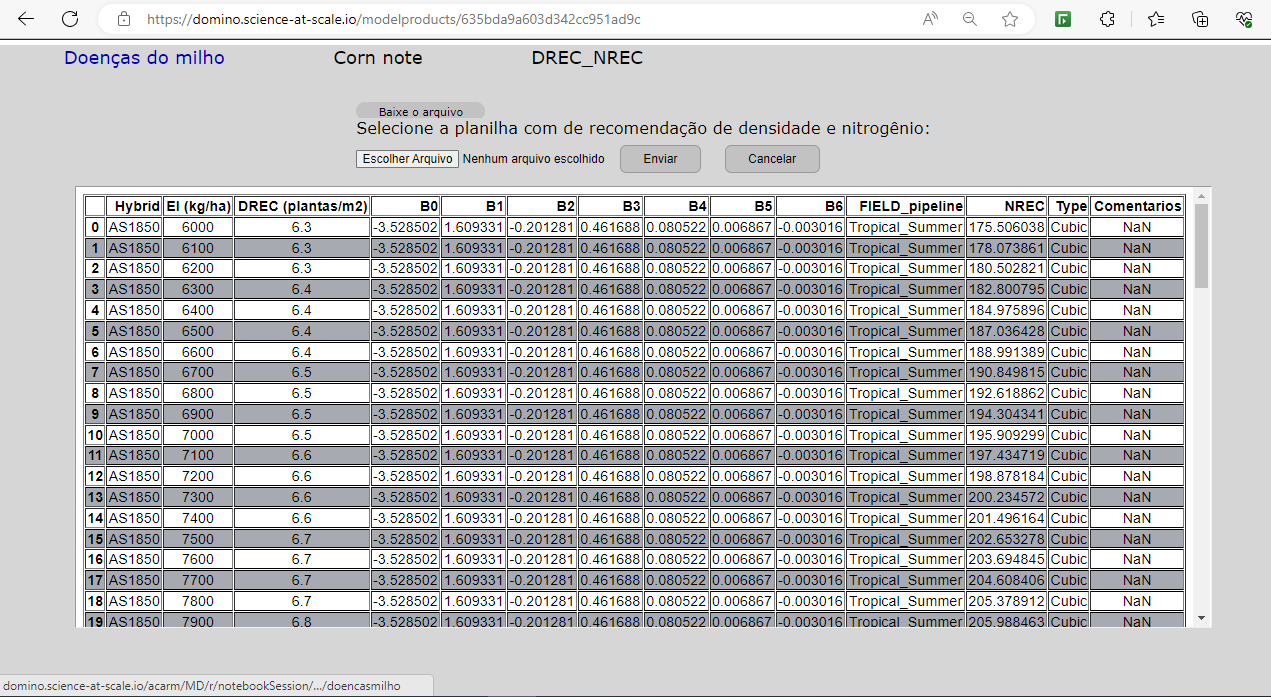


Figura - DREC\_NREC

* 1. Função filtra1(p,b,pi,tabela13)

Função filtra1(p,b,pi,tabela13) realiza a filtragem das informações, para mostrar as informações na área lateral como um documento. Ela recebe as informações como híbrido, biotecnologia(quando houver), Pipeline e a tabela, e a partir destes itens ela gera uma variável com os itens filtrados até que haja apenas uma linha que será carregada no script page3.

* 1. Função atualiza1

Função atualiza1(tabela10a) e usada em conjunto com a função filtra1 para gerar os dados que serão carregados no combo da página.

* 1. Função adicionaitem(cb4,story1, tabela10c, info1)

Função adicionaitem(cb4,story1, tabela10c, info1) e responsável por guardar as informações adicionadas pelo botão continuar adicionando e gera a partir das informações do combo cb4 e story1 as informações da variável tabela10c que é uma variável usada na inserção de novos valores a tabela original.

1. **Funções da página gestão de produtos que estão nos scripts note e arquivo.**

Na página Corn note também há uma opção no menu, que foi desenvolvida apenas para ela, que é o menu de gestão de Produtos, que tem a função como o próprio nome diz de gerenciar as informações dos produtos tanto de híbridos que estão habilitados para uso, como outras informações, que são inseridas por meio de planilhas vindas externamente (planilhas em Excel), com dados que não estão no Lake ou dariam muitas horas para serem inseridos manualmente. Na página e possível fazer download dos exemplos e da planilha com os dados finais, que pode ser entregue diretamente para o setor de KT. Para isso foram criadas funções especificas para cada inserção.

Ao se acionar o menu com o nome Gestão de Produtos, o sistema inicia a função gestaoprodutos(seguido do prefixo do país) que carrega a página de arquivoprodutos .

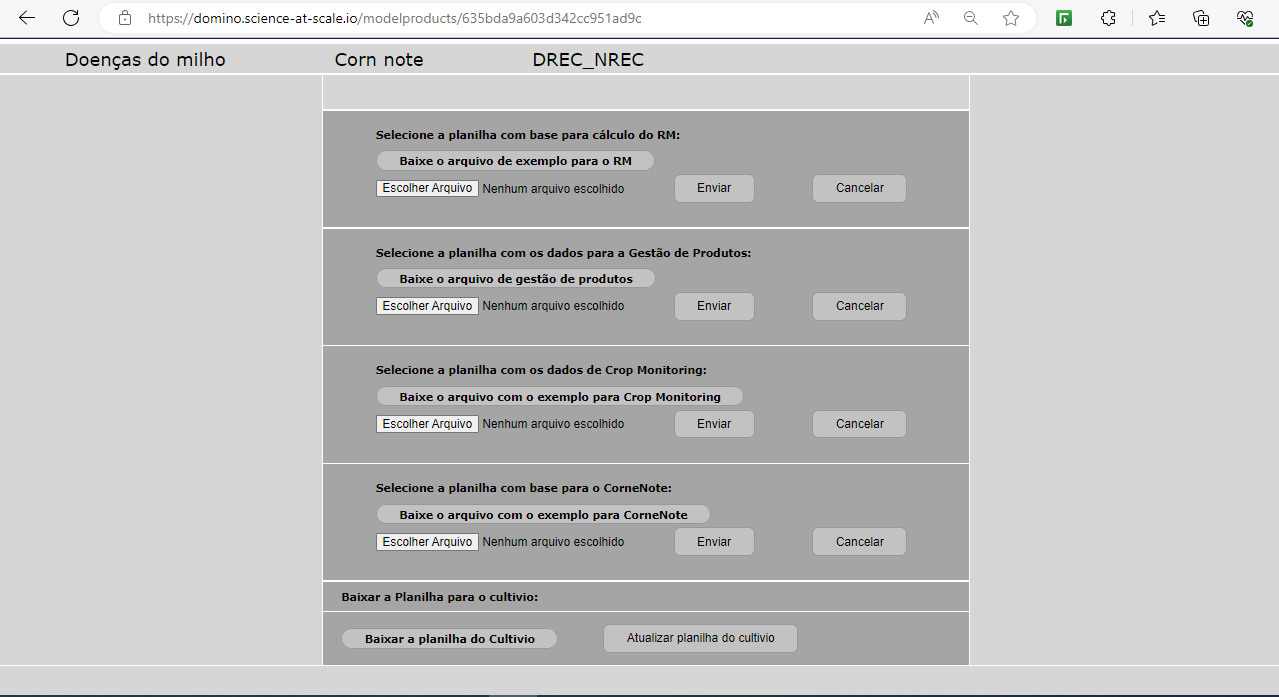


Figura -Página de Gestão de produtos.

Dentre as opções de envio temos:

* 1. Envio do RM/Fenotype - post\_arquivo(mais o prefixo do país)1.

É uma área destinada para inserir a planilha do RM no caso do brasil e Fenotype no caso de Argentina e México, o usuário escolhe a planilha com os dados, e pressionar o botão de enviar da área de RM(ou fenotype) e o sistema inicia a post\_arquivo(mais o prefixo do país)1 que faz um upload do arquivo para uma variável interna chamada “arquivo” e salva o seu conteúdo em arquivo temporário do sistema com a extensão xlsx (se existir um outro arquivo temporário o mesmo e substituído por este novo). O novo arquivo será usado na função page5(prefixo do País) que o coloca em uma variável e carrega uma outra variável com os valores que estão no sistema, para poder comparara-las e mostrar quais alterações foram feitas descriminando o que é novo e o que está no sistema atualmente na base( na última coluna da tabela)

* 1. Botão Salvar de dados RM/Fenotype - Função salvar (prefixo do país)4.

A partir de uma confirmação do usuário pressionando o botão Salvar, a página envia um Post para a função Salvar(prefixo do País) 4 que faz o tratamento das informações de data e hora do sistema, para gravar o nome do arquivo de backup. É feita a salva dos dados novos na planilha do sistema, bem como os dados que estavam na base são salvos(em uma pasta de backup) com o nome da tabela, mais o nome do usuário que fez a alteração e a data e hora atual do sistema.

* 1. Envio de dados da Gestão de Produtos -post\_arquivo(prefixo do país)2.

Outra opção e a de envio dos dados para a gestão de produtos que estão ativos, lançamentos e desativados no sistema(que também e usada para filtrar os dados que serão entregues na planilha final e enviados para o Lake). Ela inicia a partir da escolha da planilha e o pressionar o botão de envio. Iniciando a função post\_arquivo(prefixo do País) 2 que recebe o arquivo enviado pelo usuário e converte em dois arquivos um sendo o temporário em xlsx e outro em csv. Inicia-se também a página dadosdefiltro.html. Ao iniciar a página ela solicita a inicialização da função page6(prefixo do País) que recebe o arquivo temporário e o coloca em uma variável, também coloca os valores que estão na base em outro para poder comparar as informações e mostrar quais serão as alterações foram feitas descriminando o que é novo e o que está no sistema atualmente na base( na última coluna da tabela)

* 1. Botão Salvar de dados da Gestão de Produtos -Função salvar (prefixo do País) 5.

A partir de uma confirmação do usuário pressionando o botão Salvar, a página envia um Post para a função Salvar(prefixo do País) 5 que faz o tratamento das informações de data e hora do sistema, para gravar o nome do arquivo de backup. É feita a salva dos dados novos na planilha do sistema, bem como os dados que estavam na base são salvos(em uma pasta de backup) com o nome da tabela, mais o nome do usuário que fez a alteração e a data e hora atual do sistema.

Além de salvar os dados as função também atualiza as informações nas planilhas de doenças e notas , dados como por exemplo os novos produtos, status, pipeline e biotecnologia(caso tenha no país) é realizada uma verificação se há dados faltantes no filtro, pois caso haja, o status dos produtos que não aparecem são mudados para desativados. E após a verificação são atualizadas as planilhas de doenças e notas.

* 1. Envio de dados de Corp Monitoring - post\_arquivo(prefixo do País) 3.

Outra opção de entrada e a dos dados de Corp Monitoring que é uma nova implementação com os dados de tempo de crescimento do milho e suas fases.

A partir da escolha da planilha no escolher arquivo, e pressionar o botão de enviar da planilha escolhida o sistema inicia a função post\_arquivobr3 que faz um upload do arquivo para uma variável interna chamada “arquivo” e salva o seu conteúdo em arquivo temporário do sistema com a extensão xlsx (se existir um outro arquivo temporário o mesmo e substituído por este novo). O novo arquivo será usado na função page7(prefixo do País) que o coloca em uma variável e carrega uma outra variável com os valores que estão no sistema, para poder comparara-las e mostrar quais alterações foram feitas descriminando o que é novo e o que está no sistema atualmente na base( na última coluna da tabela)

* 1. Botão Salvar de dados de Corp Monitoring -função Salvar(prefixo do País) 6

A partir de uma confirmação do usuário pressionando o botão Salvar, a página envia um Post para a função Salvar(prefixo do País) 6 que faz o tratamento das informações de data e hora do sistema, para gravar o nome do arquivo de backup. É feita a salva dos dados novos na planilha do sistema, bem como os dados que estavam na base são salvos(em uma pasta de backup) com o nome da tabela, mais o nome do usuário que fez a alteração e a data e hora atual do sistema.

* 1. Envio de dados de Corne note -post\_arquivo(prefixo do País) 4

E a última opção e a de fazer uma atualização de toda a planilha de notas e recomendações que aparecem na tela de Corn note. A partir da escolha da planilha no escolher arquivo, e pressionar o botão de enviar da planilha escolhida o sistema inicia a função post\_arquivobr4 que faz um upload do arquivo para uma variável interna chamada “arquivo” e salva o seu conteúdo em arquivo temporário do sistema com a extensão xlsx (se existir um outro arquivo temporário o mesmo e substituído por este novo). O novo arquivo será usado na função page8(prefixo do País) que o coloca em uma variável e carrega uma outra variável com os valores que estão no sistema, para poder comparara-las e mostrar quais alterações foram feitas descriminando o que é novo e o que está no sistema atualmente na base( na última coluna da tabela).

* 1. Botão Salvar de dados de Corne note -função Salvar (prefixo do País) 7.

A partir de uma confirmação do usuário pressionando o botão Salvar, a página envia um Post para a função Salvar(prefixo do País) 7 que faz o tratamento das informações de data e hora do sistema, para gravar o nome do arquivo de backup. É feita a salva dos dados novos na planilha do sistema, bem como os dados que estavam na base são salvos(em uma pasta de backup) com o nome da tabela, mais o nome do usuário que fez a alteração e a data e hora atual do sistema.

* 1. Planilhas de exemplo ou baixa de arquivo - Função get\_arquivo.

Na página também há botões para baixar exemplos e arquivos para os usuários poderem usar para entrada de dados, bem como verificações do que está no sistema. E eles ativam as funções abaixo que fazem o download dos arquivos para o usuário:

Função get\_arquivo(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em csv de doenças de milho do país que o usuário está logado

Função get\_arquivo2(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX da planilha com os dados finais para KT do país que o usuário está logado.

Função get\_arquivo3(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX de exemplo dos dados de notas e recomendações para milho do país que o usuário está logado.

Função get\_arquivo4(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX de exemplo dos dados e formato das informações da gestão de produto do país que o usuário está logado.

Função get\_arquivo5(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX de exemplo dos dados e formato das informações de densidade e nitrogênio e do país que o usuário está logado.

Função get\_arquivo6(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX de exemplo dos dados e formato das informações de fenologia ou RM (rendimento médio) do país que o usuário está logado.

Função get\_arquivo7(prefixo do País) envia para o usuário a partir do acionamento do botão a planilha em XLSX de exemplo dos dados e formato das informações de corp monitoring do país que o usuário está logado.

1. **Atualiza\_banco.**

Outro script muito importante para a atualização dos dados que não tem a ligação direta com os scripts do console e o script de atualização dos dados no sistema, que gera a planilha final para ser consumida pelos aplicativos da marca, como planilha que pode ser entregue, como também como fonte dos dados do Lake, que fica disponível para os aplicativos da marca. Este script fica na área de aplicações scheduled do Domino para ser executada no horário noturno ou pela madrugada.

Atualiza\_banco(prefixo do país) – e responsável por filtrar com os dados da planilha da gestão de produtos, que são as informações que podem ser disponibilizadas para os aplicativos da marca, sem restrições.

O script começa com o carregamento das bibliotecas necessárias para a conexão com o banco de dados do google cloud, bibliotecas de cálculo e manipulação dos dados.

Depois inicia-se a função para estabelecer uma comunicação com a base de dados do Lake(função crida anteriormente pelo grupo de Field Trialing).

Começamos pela função comparativo que grava em uma aba todos os dados da planilha de gestão de produtos. Ela contém os híbridos ativos ou lançamentos que serão disponibilizados para os aplicativos e servem para um filtro de dados, e nas próximas versões terão quais os dados que estão faltando em cada uma das abas (planilhas que integra os dados proveniente dadas outras planilhas cada uma em uma aba específica).

Na função carrega-se os dados da planilha de gestão de produtos em uma variável, depois realiza-se um filtro somente com os dados de híbridos lançamento ou ativos, que serão gravados em um arquivo chamado compartivo.csv (e este arquivo irá compor a aba comparativo no arquivo final).

2. 1. função Stability.

Próxima função e a stability, que se conecta ao Lake para baixar os dados e após fazer os cálculos e filtrar as informações necessárias retorna as informações para uma nova planilha no Lake.

Primeiramente e definido qual planilha que será acessada dentro do Lake, usando a função que estabelece a conexão com o banco carrega-se os dados uma variável, realiza-se a filtragem dos dados mediante as colunas. Na coluna Variable filtra-se os dados de YLD e na coluna Protocol filtra-se os dados com FTN, retira-se as colunas que não serão usadas como também as duplicadas. É multiplicado o valor de Head\_mean e EI por 100. São carregados os dados do filtro(dados inseridos pelo console na gestão de produtos) e normalizados os dados da coluna Situação para separa as informações apenas de ativos e lançamentos.

E realizada a junção dos nomes de produto com a biotecnologia, para poder filtrar os dados que vem do banco de dados, realiza-se também a normalização dos dados de Pipeline para que atendam ao que está no banco de dados, e se faz a separação dos dados de Pipeline, Marca e nome do produto para as marca do portifólio da Bayer, separando também os concorrentes cria-se variáveis que serão responsáveis pela filtragem dos dados, como também e feita a alteração do nome de colunas, para os dados que vem da variável do banco de dados.

Remove e renomeia-se as colunas para que atendam a um formato que foi pré-estabelecidos por MD para a entrega para os aplicativos, realiza-se a retirada dos valores vazios e se grava-se um arquivo para backup no sistema e grava-se os dados no repositório do Lake seguindo o padrão de prefixo do país com 3 dígitos mais a letra T(de total) e o nome da aba( planilha csv) de saída para os aplicativos da marca.

E finalmente e retornado os dados da variável filtrada para ser usada em outra função.

* 1. função m\_b.

A próxima função e a de m\_b que utiliza os dados da função stability para gerar os valores para o ranking de produtos.

Ela inicia carregando os dados que foram obtidos da função citada acima e cria-se uma variável com apenas os dados de Protocol e Head, depois estes dados são unidos para a criação de uma nova coluna, que serão agrupados em dados únicos e contados a partir dos dados de Protocol e head.

Os Dados desta nova coluna serão separados para que os dados com mais de 4 campos possam ser calculados para a planilha final( há necessidade deste mínimo para que os cálculos sejam validos) e estes dados serão filtrados da primeira variável separando apenas os dados que atende esta condição.

E criada uma variável com as colunas da planilha final de MB e com dados vazios que serão preenchidos pelo laço de repetição, que filtra os dados e aplica o modelo linear.

Após serem preenchidos os dados e feita a retirada de possíveis dados repetidos e dados vazios. Grava-se um arquivo para backup no sistema e grava-se os dados no repositório do Lake seguindo o padrão de prefixo do país com 3 dígitos mais a letra T(de total) e o nome da aba( planilha csv) de saída para os aplicativos da marca.

* 1. função e a phentype.

A próxima função e a phentype (que é apenas usado pelo Brasil, Mexico e Argentina não usam estes dados, usam os dados fixo inseridos pela planilha no console), nela se une os dados vindos de Breding com os dados do cultivo inseridos no Lake.

Primeiramente são geradas as variáveis com os dados de correntes e histórico, para se fazer uma comparação e retirar os dados fenológicos e de rendimento dos híbridos. É realizada a retirada das células vazias e colunas que não são necessárias para a comparação das duas variáveis(planilhas), são alterados os nomes de alguns produtos, mas são registrados de formas diferentes. Outro item a ser feito e sapara os código que serão usados para as colunas da planilha final e filtrados em ambas as variáveis.

E realizada a comparação e fusão das duas variáveis em uma só variável, e depois são normalizados os dados de pipeline como nas funções anteriores e realizado o carregamento dos dados da planilha de filtro (gestão de produtos) e adequação dos dados para serem filtrados, e separados os híbridos e biotecnologia que serão vendidos e retirando os obsoletos ou que iram sair do mercado. E realizada a correção dos nomes do pipeline para a posterior filtragem dos dados. Filtra-se os dados de híbridos ativos ou lançamento e cria-se variáveis para posterior filtragem.

Separa-se as colunas necessárias e que irão para o sistema, corrige-se os nomes da coluna OBS\_code retirando o final N/A dos dados. Realiza-se o agrupamento das informações para o pipeline, produto, comercial name e OBS\_code e faz a média dos dados da coluna OBS\_numValue e em seguida e realizada uma transposição de linha para coluna dos dados da coluna OBS\_code e OBS\_numValue e mantem se os demais dados.

E realizada as correções de nomes dos pipeline e retirada das colunas desnecessárias, retira-se também os dados repetidos e vazios. Retira-se o prefixo das biotecnologias e grava-se uma arquivo de backup que será usado para se juntar a outras informações vidas do console e o cálculo de RM.

Inicia-se o processo de cálculo do RM com base dos dados de plantio, com a carga dos dados do arquivo de valores de RM que foram criados pelo console. Separa se os dados MST e YLD para o cálculo de rendimento e os valores de base de alguns RM fornecidos por breeding.

É realizado um loop para gerar os valores calculados de RM fazendo uma filtragem por Pipeline e dados de MST e YLD. Após esta filtragem e feita uma nova por Seção no caso as localizações de cada plantio, eles são separados e contados para saber se temos no mínimo 4 campos plantados para poder gerar o cálculo. E finalmente separamos por nome do campo (Field name) que serão usados para cálculo do RM, é realizada a separação dos campos que não tem dados e os que não tem o valor default de RM e finalmente passamos para o modelo linear para obtermos o valor final de RM(o RM calculado).

Com os valores calculados e realizado um agrupamento dos dados por new\_pipeline, Field\_planting\_Season e Field\_name tendo os valores de medias de rsquare e p\_value médios(valores obtidos no modelo linear) realiza-se a verificação se o valor dos grupos de p-valore e maior que 0,15 (os menores e iguais a 0,15 ficaram fora da planilha final).

Realiza-se a correção dos nomes dos Pipelines para os usados normalmente na planilha final, realiza-se o agrupamento e cálculo do RM para a coluna RM\_Calculated sendo estes valores médios em relação ao grupo de pipeline e híbrido.

Após calculados os valores e realizada a união dos dados obtidos com os dados da planilha de RM (vinda do console), realiza-se a transformação dos números de Float em inteiros e finalmente se separa as colunas que serão entregues na aba(planilha do Lake) fenotype.

Grava-se um arquivo para backup no sistema e grava-se os dados no repositório do Lake seguindo o padrão de prefixo do país com 3 dígitos mais a letra T(de total) e o nome da aba( planilha csv) de saída para os aplicativos da marca.

* 1. função e a h2h

A próxima função e a h2h que tem como objetivo fazer uma comparação entre híbridos.

A função inicia com a conexão com o banco de dados do Lake e cria uma variável como os dados totais das comparações de h2h, realiza-se uma filtragem dos dados de Variable, Macroregion, Year e Protocol , e realiza-se a multiplicação por 100 dos valores de Head\_mean e diffmean. É carregada a planilha para o filtro(planilha que e carregada na gestão de Produtos), realiza-se a mudança do nome das colunas a serem comparadas e de Pipeline.

Realiza-se a união das duas variáveis(variável com os dados obtidos do Lake com a que contêm os dados de filtro) para se obter os dados em comum das duas, como um filtro que integra os mesmos a uma única variável. Cria-se uma mova coluna com o dados para head e a marca para depois filtrar os dados no final, e gera-se uma variável com os dados de head filtrados e depois se faz a filtragem do Other, na sequência são separadas as marcas para não haver comparação entre marcas diferentes, por fim são retirados os dados duplicados e ordenados pelo head.

E por fim grava-se um arquivo para backup no sistema e grava-se os dados no repositório do Lake seguindo o padrão de prefixo do país com 3 dígitos mais a letra T(de total) e o nome da aba( planilha csv) de saída para os aplicativos da marca.

* 1. funções fitradoencas , notadohibrido e nderc

As funções fitradoencas , notadohibrido e nderc fazem apenas a filtragem dos dados que vem das planilhas do console retirando os dados old´s e que não sejam ativos ou lançamento. Também se retira as colunas desnecessárias para a planilha final e o Lake. Gravando os dados como nas funções anteriores.

* 1. função agronomicrick

A função agronomicrick realiza a geração dos dados para classificação do híbrido na plataforma do aplicativo da marca.

Primeiramente são geradas as variáveis com os dados de correntes e histórico, para se fazer uma comparação, retirar os dados fenológicos e de rendimento dos híbridos. Retira-se as células vazias e colunas que não são necessárias para a comparação das duas variáveis(planilhas), são alterados os nomes de alguns produtos que foram registrados de forma diferente (mas são iguais). Separa-se os código que serão usados para as colunas da planilha final e que foram filtrados em ambas as variáveis.

E realizada a comparação e fusão das duas variáveis em uma só variável, e depois são normalizados os dados de pipeline, como nas funções anteriores e realizado o carregamento dos dados da planilha de filtro (gestão de produtos) e adequação dos dados para serem filtrados, separa-se os híbridos e biotecnologia que serão vendidos, retirando os obsoletos ou que iram sair do mercado. E realizada a correção dos nomes do pipeline para a posterior filtragem dos dados. Filtra-se os dados de híbridos ativos ou lançamento, criando variáveis para posterior filtragem.

Gera-se as variáveis que conterão os dados de pipeline e comercial name , mantem-se os dados que contêm YLD.N/A , inicia-se o loop filtrando os dados primeiramente dos pipelines, retira-se os valores em branco da coluna EIRange. Agrupa-se os dados de Product,new\_pipeline, FIELD\_plantingSeason, UniqueName , FIELD\_name e é feita a média dos valores de OBS\_numValue, reseta-se do índice, é criada uma variável com os dados de produto e pipeline, cria-se uma coluna com o nome yld\_calc , que tem como conteúdo a subtração dos valores de OBS\_numValue dos valores de EI, depois separa-se os dados positivos dos negativos, depois realiza-se a contagem dos valores positivos e gera-se uma coluna com os dados do número de produtos totais do cálculo , gera-se o valor percentual de valores de vitória (valores maiores que 0), realiza-se o arredondamento para 2 casas decimais, realiza-se o agrupamento de produtos por EI e realiza-se a média dos valores.

Separa-se as colunas necessárias, corrige-se os nomes dos dados de pipeline, retira-se os duplicados e valores vazios.

Finalmente grava-se um arquivo para backup no sistema e grava-se os dados no repositório do Lake seguindo o padrão de prefixo do país com 3 dígitos mais a letra T(de total) e o nome da aba( planilha csv) de saída para os aplicativos da marca.

* 1. função e a geraplanilha

E a última função que e a que gera os dados para a planilha final que futuramente será integrada ao console para que o usuário possa atualizar a planilha a qualquer hora com o dados do banco de dados.

Esta função e a geraplanilha que após serem rodados todas as outras funções gera os dados atualizados para uma planilha que poderá ser baixada dentro do console e que pode ser entregue para o time de KT.

Esta função cria uma lista com todos os nomes que serão das abas e cria uma planilha vazia em XLSX depois e criado um loop para conectar ao banco de dados e gera cada aba com o nome especificado na lista e salva os dados que são obtidos do banco nesta aba da planilha final e grava também os dados do comparativo para poder ser feitos os filtros pelo KT.

1. **Anexos**

views.py

1. from flask\_app import app
2. from flask\_app import templates
3. from flask\_app import static
4. from flask\_app.brarquivo import \*
5. from flask\_app.brdoencas import \*
6. from flask\_app.brnote import \*
7. from flask\_app.argdoencas import \*
8. from flask\_app.argarquivo import \*
9. from flask\_app.argnote import \*
10. from flask\_app.mxdoencas import \*
11. from flask\_app.mxarquivo import \*
12. from flask\_app.mxnote import \*
13. import pandas as pd
14. from flask import Flask, render\_template, redirect, url\_for
15. from flask.globals import request
16. import time
17. daaltera=time.ctime()
18. @app.route("/")
19. @app.route("/iniciar",methods=['GET', 'POST'])
20. def iniciar():
21. if request.method == 'POST':
22. nome = request.headers.get("domino-username")
23. pais=request.form['cb1']
24. acesso=pd.read\_excel("/domino/datasets/local/MD/acesso.xlsx")
25. t = request.form.get('tipo')
26. pais=pais.lower()
27. nome=nome.lower()
28. acesso =acesso[acesso['pais']== pais]
29. acesso =acesso[acesso['usuario']== nome]
30. if not(acesso.empty) and acesso['acesso'].values[0]== "s" and t=="on":
31. desabilita2 = 'disabled'
32. desabilita1 = 'disabled'
33. desabilita3 = ''
34. invisivel=''
35. if pais == "brasil" :
36. return redirect(url\_for("homepage"))
37. else:
38. if pais == "argentina":
39. return redirect(url\_for("homepagear"))
40. else:
41. return redirect(url\_for("homepagem"))
42. else:
43. return render\_template("iniciar1.html", invisivel='')
44. else:
45. invisivel='hidden'
46. return render\_template("iniciar1.html", invisivel=invisivel)
48. @app.route("/editar2",methods=['POST'])
49. def editar2(methods=['POST']):
50. return editar2br()
51. @app.route("/editar",methods=['POST'])
52. def editar(methods=['POST']):
53. return editarbr()
55. @app.route("/salvar",methods=['POST'])
56. def salvar(methods=['POST']):
57. return salvarbr()
59. @app.route("/salvar2",methods=['POST'])
60. def salvar2(methods=['POST']):
61. return salvar2br()
63. @app.route("/doencasmilho",methods=['GET', 'POST'])
64. def homepage(methods=['GET', 'POST']):
65. return homepagebr()
66. @app.route("/gestaoprodutos",methods=['GET', 'POST'])
67. def gestaoprodutos(methods=['GET', 'POST']):
68. return gestaoprodutosbr()
69. @app.route('/cornote',methods=['GET', 'POST'])
70. def page1(methods=['GET', 'POST']):
71. return page1br()
72. @app.route('/DREC\_NREC')
73. def page2(methods=['GET', 'POST']):
74. return page2br()
75. @app.route('/pagina\_de\_descricao')
76. def page3(methods=['GET', 'POST']):
77. return page3br()
78. @app.route('/DREC\_NREC salvar')
79. def page4(methods=['GET', 'POST']):
80. return page4br()
81. @app.route('/DREC\_NREC inicio')
82. def page4a(methods=['GET', 'POST']):
83. return page4abr()
84. @app.route('/rm\_valoresbr')
85. def page5(methods=['GET', 'POST']):
86. return page5br()
87. @app.route('/filtrobr')
88. def page6(methods=['GET', 'POST']):
89. return page6br()
90. @app.route('/corpmonitorbr')
91. def page7(methods=['GET', 'POST']):
92. return page7br()
93. @app.route('/descricaobr')
94. def page8(methods=['GET', 'POST']):
95. return page8br()
96. @app.route("/arquivos1", methods=["GET"])
97. def get\_arquivo(methods=["GET"]):
98. return get\_arquivobr()
100. @app.route("/arquivos2", methods=["GET"])
101. def get\_arquivo2(methods=["GET"]):
102. return get\_arquivo2br()
103. @app.route("/arquivos3", methods=["GET"])
104. def get\_arquivo3(methods=["GET"]):
105. return get\_arquivo3br()
106. @app.route("/arquivos4", methods=["GET"])
107. def get\_arquivo4(methods=["GET"]):
108. return get\_arquivo4br()
109. @app.route("/arquivos5", methods=["GET"])
110. def get\_arquivo5(methods=["GET"]):
111. return get\_arquivo5br()
112. @app.route("/arquivos6", methods=["GET"])
113. def get\_arquivo6(methods=["GET"]):
114. return get\_arquivo6br()
116. @app.route("/arquivos", methods=["POST"])
117. def post\_arquivo(methods=["POST"]):
118. return post\_arquivobr()
120. @app.route("/arquivos1", methods=["POST"])
121. def post\_arquivo1(methods=["POST"]):
122. return post\_arquivobr1()
124. @app.route("/arquivos2", methods=["POST"])
125. def post\_arquivo2(methods=["POST"]):
126. return post\_arquivobr2()
128. @app.route("/arquivos3", methods=["POST"])
129. def post\_arquivo3(methods=["POST"]):
130. return post\_arquivobr3()
132. @app.route("/arquivos4", methods=["POST"])
133. def post\_arquivo4(methods=["POST"]):
134. return post\_arquivobr4()
136. @app.route("/salvo")
137. def salvo():
138. return render\_template("salvo.html")
139. @app.route("/salvo3")
140. def salvo3():
141. return render\_template("salvo3.html")
142. @app.route("/salvo1", methods=["POST"])
143. def salvo1():
144. return salvarbr1()
145. @app.route("/salvo4", methods=["POST"])
146. def salvo4():
147. return salvarbr4()
148. @app.route("/salvo5", methods=["POST"])
149. def salvo5():
150. return salvarbr5()
151. @app.route("/salvo6", methods=["POST"])
152. def salvo6():
153. return salvarbr6()
154. @app.route("/salvo7", methods=["POST"])
155. def salvo7():
156. return salvarbr7()
158. @app.route("/salvo2")
159. def salvo2():
160. return render\_template("salvo2.html")
162. @app.route("/editar2ar",methods=['POST'])
163. def editar2ar(methods=['POST']):
164. return editar2arg()
165. @app.route("/editarar",methods=['POST'])
166. def editarar(methods=['POST']):
167. return editararg()
168. @app.route("/doencasmilhoar",methods=['GET', 'POST'])
169. def homepagear(methods=['GET', 'POST']):
170. return homepagearg()
171. @app.route("/gestaoprodutosar",methods=['GET', 'POST'])
172. def gestaoprodutosar(methods=['GET', 'POST']):
173. return gestaoprodutosarg()
174. @app.route('/cornotear',methods=['GET', 'POST'])
175. def page1ar(methods=['GET', 'POST']):
176. return page1arg()
177. @app.route('/DREC\_NRECar')
178. def page2ar(methods=['GET', 'POST']):
179. return page2arg()
180. @app.route('/pagina\_de\_descricaoar')
181. def page3ar(methods=['GET', 'POST']):
182. return page3arg()
183. @app.route('/DREC\_NREC salvarar')
184. def page4ar(methods=['GET', 'POST']):
185. return page4arg()
186. @app.route('/DREC\_NREC inicioar')
187. def page4aar(methods=['GET', 'POST']):
188. return page4aarg()
189. @app.route('/rm\_valoresar')
190. def page5ar(methods=['GET', 'POST']):
191. return page5arg()
192. @app.route('/filtroar')
193. def page6ar(methods=['GET', 'POST']):
194. return page6arg()
195. @app.route('/corpmonitorar')
196. def page7ar(methods=['GET', 'POST']):
197. return page7arg()
198. @app.route('/descricaoar')
199. def page8ar(methods=['GET', 'POST']):
200. return page8arg()
201. @app.route("/arquivos1ar", methods=["GET"])
202. def get\_arquivoar(methods=["GET"]):
203. return get\_arquivoarg()
204. @app.route("/arquivos2ar", methods=["GET"])
205. def get\_arquivo2ar(methods=["GET"]):
206. return get\_arquivo2arg()
207. @app.route("/arquivos3ar", methods=["GET"])
208. def get\_arquivo3ar(methods=["GET"]):
209. return get\_arquivo3arg()
210. @app.route("/arquivos4ar", methods=["GET"])
211. def get\_arquivo4ar(methods=["GET"]):
212. return get\_arquivo4arg()
213. @app.route("/arquivos5ar", methods=["GET"])
214. def get\_arquivo5ar(methods=["GET"]):
215. return get\_arquivo5arg()
216. @app.route("/arquivos6ar", methods=["GET"])
217. def get\_arquivo6ar(methods=["GET"]):
218. return get\_arquivo6arg()
220. @app.route("/arquivosar", methods=["POST"])
221. def post\_arquivoar(methods=["POST"]):
222. return post\_arquivoarg()
223. @app.route("/arquivos1ar", methods=["POST"])
224. def post\_arquivo1ar(methods=["POST"]):
225. return post\_arquivo1arg()
226. @app.route("/arquivos2ar", methods=["POST"])
227. def post\_arquivo2ar(methods=["POST"]):
228. return post\_arquivo2arg()
229. @app.route("/arquivos3ar", methods=["POST"])
230. def post\_arquivo3ar(methods=["POST"]):
231. return post\_arquivo3arg()
232. @app.route("/arquivos4ar", methods=["POST"])
233. def post\_arquivo4ar(methods=["POST"]):
234. return post\_arquivo4arg()
236. @app.route("/salvararg",methods=['POST'])
237. def salvarar(methods=['POST']):
238. return salvararg()
239. @app.route("/salvar1ar",methods=['POST'])
240. def salvar1ar(methods=['POST']):
241. return salvar1arg()
242. @app.route("/salvar2ar",methods=['POST'])
243. def salvar2ar(methods=['POST']):
244. return salvar2arg()
245. @app.route("/salvoar")
246. def salvoar():
247. return render\_template("salvoar.html")
248. @app.route("/salvo3ar")
249. def salvo3ar():
250. return render\_template("salvo3arg.html")
251. @app.route("/salvo1ar", methods=["POST"])
252. def salvo1ar():
253. return salvar1arg()
254. @app.route("/salvo4ar", methods=["POST"])
255. def salvo4ar():
256. return salvar4arg()
257. @app.route("/salvo5ar", methods=["POST"])
258. def salvo5ar():
259. return salvar5arg()
260. @app.route("/salvo6ar", methods=["POST"])
261. def salvo6ar():
262. return salvar6arg()
263. @app.route("/salvo7ar", methods=["POST"])
264. def salvo7ar():
265. return salvar7arg()
266. @app.route("/salvo2ar")
267. def salvo2ar():
268. return render\_template("salvo2ar.html")
270. @app.route("/editar2m",methods=['POST'])
271. def editar2m(methods=['POST']):
272. return editar2mx()
273. @app.route("/editarm",methods=['POST'])
274. def editarm(methods=['POST']):
275. return editarmx()
276. @app.route("/doencasmilhom",methods=['GET', 'POST'])
277. def homepagem(methods=['GET', 'POST']):
278. return homepagemx()
279. @app.route("/gestaoprodutosm",methods=['GET', 'POST'])
280. def gestaoprodutosm(methods=['GET', 'POST']):
281. return gestaoprodutosmx()
282. @app.route('/cornotem',methods=['GET', 'POST'])
283. def page1m(methods=['GET', 'POST']):
284. return page1mx()
285. @app.route('/DREC\_NRECm')
286. def page2m(methods=['GET', 'POST']):
287. return page2mx()
288. @app.route('/pagina\_de\_descricaom')
289. def page3m(methods=['GET', 'POST']):
290. return page3mx()
291. @app.route('/DREC\_NREC salvarm')
292. def page4m(methods=['GET', 'POST']):
293. return page4mx()
294. @app.route('/DREC\_NREC iniciom')
295. def page4am(methods=['GET', 'POST']):
296. return page4amx()
297. @app.route('/rm\_valoresm')
298. def page5m(methods=['GET', 'POST']):
299. return page5mx()
300. @app.route('/filtrom')
301. def page6m(methods=['GET', 'POST']):
302. return page6mx()
303. @app.route('/corpmonitorm')
304. def page7m(methods=['GET', 'POST']):
305. return page7mx()
306. @app.route('/descricaom')
307. def page8m(methods=['GET', 'POST']):
308. return page8mx()
309. @app.route("/arquivos1m", methods=["GET"])
310. def get\_arquivom(methods=["GET"]):
311. return get\_arquivomx()
312. @app.route("/arquivos2m", methods=["GET"])
313. def get\_arquivo2m(methods=["GET"]):
314. return get\_arquivo2mx()
315. @app.route("/arquivos3m", methods=["GET"])
316. def get\_arquivo3m(methods=["GET"]):
317. return get\_arquivo3mx()
318. @app.route("/arquivos4m", methods=["GET"])
319. def get\_arquivo4m(methods=["GET"]):
320. return get\_arquivo4mx()
321. @app.route("/arquivos5m", methods=["GET"])
322. def get\_arquivo5m(methods=["GET"]):
323. return get\_arquivo5mx()
324. @app.route("/arquivos6m", methods=["GET"])
325. def get\_arquivo6m(methods=["GET"]):
326. return get\_arquivo6mx()
328. @app.route("/arquivosm", methods=["POST"])
329. def post\_arquivom(methods=["POST"]):
330. return post\_arquivomx()
331. @app.route("/arquivos1m", methods=["POST"])
332. def post\_arquivo1m(methods=["POST"]):
333. return post\_arquivo1mx()
334. @app.route("/arquivos2m", methods=["POST"])
335. def post\_arquivo2m(methods=["POST"]):
336. return post\_arquivo2mx()
337. @app.route("/arquivos3m", methods=["POST"])
338. def post\_arquivo3m(methods=["POST"]):
339. return post\_arquivo3mx()
340. @app.route("/arquivos4m", methods=["POST"])
341. def post\_arquivo4m(methods=["POST"]):
342. return post\_arquivo4mx()
344. @app.route("/salvarm",methods=['POST'])
345. def salvarm(methods=['POST']):
346. return salvarmx()
347. @app.route("/salvar2m",methods=['POST'])
348. def salvar2m(methods=['POST']):
349. return salvar2mx()
351. @app.route("/salvo3m")
352. def salvo3m():
353. return render\_template("salvo3mx.html")
354. @app.route("/salvo1m", methods=["POST"])
355. def salvo1m():
356. return salvar1mx()
357. @app.route("/salvo4m", methods=["POST"])
358. def salvo4m():
359. return salvar4mx()
360. @app.route("/salvo5m", methods=["POST"])
361. def salvo5m():
362. return salvar5mx()
363. @app.route("/salvo6m", methods=["POST"])
364. def salvo6m():
365. return salvar6mx()
366. @app.route("/salvo7m", methods=["POST"])
367. def salvo7m():
368. return salvar7mx()
369. @app.route("/salvom")
370. def salvom():
371. return render\_template("salvomx.html")
372. @app.route("/salvo2m")
373. def salvo2m():
374. return render\_template("salvo2mx.html")
375. @app.route("/salvar1m", methods=["POST"])
376. def salvar1m():
377. return salvar1mx()

brdoencas.py

1. from flask\_app import templates
2. from flask\_app import static
3. import pandas as pd
4. from flask import Flask, render\_template
5. from flask.globals import request
6. import time
7. daaltera=time.ctime()
8. tabela1 = pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')
9. tabela1a =tabela1
10. tabela1=tabela1.loc[tabela1['Status'] != "old"]
11. cb1a = ''
12. cb2b = ''
13. cb3c = ''
14. cb4d = ''
15. desabilita3 = ''
16. hab=0
17. hab2=0
18. def homepagebr():
19. daaltera=time.ctime()
20. nome=request.headers.get("domino-username")
21. tabela1 = pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')
22. tabela1=tabela1.loc[tabela1['Status'] != "old"]
23. tabela1=tabela1.sort\_values(by=['Product'])
24. doencas=tabela1.columns.values.tolist()
25. del doencas [-6:]
26. del doencas [0:2]
27. if request.method == 'POST':
28. global cb1a
29. global cb2b
30. global cb3c
31. global cb4d
32. if hab!= 1 or len(cb1a) < 1 or len(cb2b) < 1 or len(cb3c) < 1 or len(cb4d) < 1 :
33. cb1 = request.form['cb1']
34. cb2 = request.form['cb2']
35. cb3 = request.form['cb3']
36. cb4 = request.form['cb4']
37. cb1a = cb1
38. cb2b = cb2
39. cb3c = cb3
40. cb4d = cb4
41. desabilita3 = ''
42. desabilita2 = 'disabled'
43. desabilita1 = 'disabled'
44. produto, bio\_tecnologia, pipeline, marca = atualiza(filtra(cb1,cb2,cb3,cb4,tabela1))
45. cb5, cb6, cb7, cb8, cb9, cb10, cb11, cb12, cb13, cb14, cb15,nota,desabilita4,ativo = carrega\_valor(filtra(cb1,cb2,cb3,cb4,tabela1))
46. global valor1t
47. global ativo1
48. ativo1 = ativo
49. valor1t = tabela1.index.values[tabela1['Product']==cb1a]
50. global tabelas1
51. tabelas1=filtra(cb1,cb2,cb3,cb4,tabela1)
52. else:
53. cb1=cb1a
54. cb2=cb2b
55. cb3=cb3c
56. cb4=cb4d
57. desabilita4 = 'disabled'
58. desabilita3 = 'disabled'
59. desabilita2 =''
60. desabilita1 =''
61. produto, bio\_tecnologia, pipeline, marca = atualiza(filtra(cb1,cb2,cb3,cb4,tabela1))
62. cb5 = request.form['cb5']
63. cb6 = request.form['cb6']
64. cb7 = request.form['cb7']
65. cb8 = request.form['cb8']
66. cb9 = request.form['cb9']
67. cb10 = request.form['cb10']
68. cb11 = request.form['cb11']
69. cb12 = request.form['cb12']
70. cb13 = request.form['cb13']
71. cb14 = request.form['cb14']
72. cb15 = request.form['cb15']
73. nota = request.form['story'] + nome + daaltera
74. global cb5a
75. cb5a = cb5
76. global cb6a
77. cb6a = cb6
78. global cb7a
79. cb7a = cb7
80. global cb8a
81. cb8a = cb8
82. global cb9a
83. cb9a = cb9
84. global cb10a
85. cb10a = cb10
86. global cb11a
87. cb11a = cb11
88. global cb12a
89. cb12a = cb12
90. global cb13a
91. cb13a = cb13
92. global cb14a
93. cb14a = cb14
94. global cb15a
95. cb15a = cb15
96. global tab1
97. tab1 = tabela1
98. tabelas1 = tabelas
99. valor1t = tabelas.index.values[tabelas['Product']==cb1a]
100. tabelas1.loc[valor1t] = [cb1a,cb2b,cb5a,cb6a,cb7a,cb8a,cb9a,cb10a,cb11a,cb12a,cb13a,cb14a,cb15a,cb3c,cb4d,nota,"",daaltera,ativo1]
102. else:
103. cb2 = ''
104. cb1 = ''
105. cb3 = ''
106. cb4 = ''
107. cb1a = cb1
108. cb2b = cb2
109. cb3c = cb3
110. cb4d = cb4
111. produto, bio\_tecnologia, pipeline, marca = atualiza(tabela1)
112. cb5, cb6, cb7, cb8, cb9, cb10, cb11, cb12, cb13, cb14, cb15,nota,desabilita4,ativo = carrega\_valor(tabela1)
113. desabilita2 = 'disabled'
114. desabilita1 = 'disabled'
115. desabilita3 = ''
116. return render\_template("doencasmilho.html",produto1=produto,bio\_tecnologia=bio\_tecnologia,pipeline=pipeline,timea=daaltera,
117. marca=marca, cb2=cb2,cb1=cb1,cb3=cb3,cb4=cb4, cb5=cb5,cb6=cb6,cb7=cb7,cb8=cb8,cb9=cb9,
118. cb10=cb10,cb11=cb11, cb12=cb12,cb13=cb13,cb14=cb14,cb15=cb15,desabilita1=desabilita1,
119. desabilita2=desabilita2,desabilita3=desabilita3,desabilita4=desabilita4,nota=nota,doencas=doencas, nome=nome)
121. def editarbr():
122. daaltera=time.ctime()
123. nome=request.headers.get("domino-username")
124. tabela1 = pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')
125. tabela1=tabela1.loc[tabela1['Status'] != "old"]
126. tabela1=tabela1.sort\_values(by=['Product'])
127. doencas=tabela1.columns.values.tolist()
128. del doencas [-6:]
129. del doencas [0:2]
130. if request.method == 'POST':
131. desabilita3 = 'disabled'
132. desabilita2 =''
133. cb1=cb1a
134. cb2=cb2b
135. cb3=cb3c
136. cb4=cb4d
137. produto, bio\_tecnologia, pipeline, marca = atualiza(filtra(cb1, cb2, cb3, cb4, tabela1))
138. cb5, cb6, cb7, cb8, cb9, cb10, cb11, cb12, cb13, cb14, cb15,nota, desabilita1, ativo = carrega\_valor(filtra(cb1, cb2, cb3, cb4, tabela1))
139. global tabelas
140. tabelas=filtra(cb1a, cb2b, cb3c, cb4d, tabela1)
141. desabilita4 = 'disabled'
142. global hab
143. hab = 1
144. else:
145. desabilita3 = ''
146. desabilita2 = 'disabled'
147. desabilita4 = ''
148. hab = 0
149. return render\_template("doencasmilho.html",produto1=produto,bio\_tecnologia=bio\_tecnologia,pipeline=pipeline,timea=daaltera,
150. marca=marca, cb2=cb2,cb1=cb1,cb3=cb3,cb4=cb4, cb5=cb5,cb6=cb6,cb7=cb7,cb8=cb8,cb9=cb9,
151. cb10=cb10,cb11=cb11, cb12=cb12,cb13=cb13,cb14=cb14,cb15=cb15,desabilita1=desabilita1,
152. desabilita2=desabilita2,desabilita3=desabilita3, desabilita4=desabilita4,hab=hab,doencas=doencas, nome=nome)
154. def salvarbr():
155. tabela1 = pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')
156. tab1=tabela1
157. #salva o valor atual da tablela para backup
158. tabelassalvar = tab1.loc[valor1t]
159. tabela3=tab1.drop(index=valor1t)
160. tabelassalvar['Status']= 'old'
161. #salva o valor que foi alterado para a tabela atual
162. tabela3=pd.concat([tabela3, tabelas1])
163. tabela1=tabela3
164. doecasmcompleta2=pd.concat([tabela3, tabelassalvar])
165. doecasmcompleta2=doecasmcompleta2.sort\_values(by=['Product'])
166. global hab
167. hab=0
168. doecasmcompleta2.to\_csv('/domino/datasets/local/MD/doecasmcompleta.csv',index = False)
169. return render\_template("salvo.html")
171. def atualiza(tabela):
172. tabela1=tabela
173. tabela1=tabela1.loc[tabela1['Status'] != "old"]
174. produto = tabela1['Product'].unique()
175. bio\_tecnologia = tabela1['Biotecnologias\_disponiveis'].unique()
176. pipeline = tabela1['Pipeline'].unique()
177. marca = tabela1['Marca'].unique()
178. return produto, bio\_tecnologia, pipeline, marca
179. produto, bio\_tecnologia, pipeline, marca = atualiza(tabela1)
181. def carrega\_valor(tab4):
182. a = len(tab4.index)
183. if a == 1 and cb1a != '' and cb2b != '' and cb3c != '' and cb4d != '':
184. cb5 = tab4["Turcicum"].values[0]
185. cb6 = tab4["Ferrugem\_Polysora"].values[0]
186. cb7 = tab4["Cercospora"].values[0]
187. cb8 = tab4["Mancha\_Branca"].values[0]
188. cb9 = tab4["Complexo\_Enfezamento"].values[0]
189. cb10 = tab4["Estria\_Bacteriana"].values[0]
190. cb11 = tab4["Diplodia\_Espiga"].values[0]
191. cb12 = tab4["Giberela\_Espiga"].values[0]
192. cb13 = tab4["Fusarium\_Colmo"].values[0]
193. cb14 = tab4["Antracnose\_Colmo"].values[0]
194. cb15 = tab4["Bipolaris"].values[0]
195. nota = tab4['notas\_de\_mudancas'].values[0]
196. ativo = tab4['situacao'].values[0]
197. desabilita4 = ''
198. else:
199. cb5 = ''
200. cb6 = ''
201. cb7 = ''
202. cb8 = ''
203. cb9 = ''
204. cb10 = ''
205. cb11 = ''
206. cb12 = ''
207. cb13 = ''
208. cb14 = ''
209. cb15 = ''
210. nota = 'Coloque a informação da mudança aqui. '
211. ativo =''
212. desabilita4 = 'disabled'
213. return cb5, cb6, cb7, cb8, cb9, cb10, cb11, cb12, cb13, cb14, cb15,nota,desabilita4,ativo
215. def filtra(p,b,pi,m,tabela3):
216. tabela2=tabela3
217. if p != '':
218. tabela2=tabela2[tabela2['Product']== p]
219. else:
220. tabela2=tabela2
221. if b != '':
222. tabela2=tabela2[tabela2['Biotecnologias\_disponiveis']== b]
223. else:
224. tabela2=tabela2
225. if pi != '':
226. tabela2 = tabela2[tabela2['Pipeline'] == pi]
227. else:
228. tabela2 = tabela2
229. if m != '':
230. tabela2 = tabela2[tabela2['Marca'] == m]
231. else:
232. tabela2 = tabela2
233. return tabela2

brnote.py

1. from flask\_app import templates
2. from flask\_app import static
3. import pandas as pd
4. from flask import Flask, render\_template
5. from flask.globals import request
6. import time
7. daaltera=time.ctime()
8. tabela10 = pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
9. descricao = pd.read\_csv('/domino/datasets/local/MD/descricao1.csv')
10. tabela10a =tabela10
11. tabela10=tabela10.loc[tabela10['Status'] != "old"]
12. cb1a = ''
13. cb2b = ''
14. cb3c = ''
15. cb4d = ''
16. desabilita3 = ''
17. tabela12a = tabela10
18. valor1 =tabela10.columns
19. hab=0
20. hab2=0
21. def page1br():
22. daaltera=time.ctime()
23. nome=request.headers.get("domino-username")
24. tabela10 = pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
25. tabela10=tabela10.sort\_values(by=['Hybrid'])
26. descricao = pd.read\_csv('/domino/datasets/local/MD/descricao1.csv')
27. tabela10=tabela10.loc[tabela10['Status'] != "old"]
28. if request.method == 'POST':
29. global cb11a
30. global cb12b
31. global cb13c
32. if hab2!= 1 or len(cb11a) < 1 or len(cb12b) < 1 or len(cb13c) < 1 :
33. cb1 = request.form['cb1']
34. cb2 = request.form['cb2']
35. cb3 = request.form['cb3']
36. cb11a = cb1
37. cb12b = cb2
38. cb13c = cb3
39. cb4 =''
40. cb5=''
41. dados= tabela10.columns
42. rec1='Descreva aqui a nota ou recomendação.'
43. rquer1=''
44. info1='Coloque a informação da mudança aqui. '+ daaltera
45. if len(cb11a) > 1 and len(cb12b) > 1 and len(cb13c) > 1 :
46. global tabela12a
47. tabela12a=filtra1(cb1,cb2,cb3,tabela10)
48. desabilita4 = ''
49. else:
50. desabilita4 = 'disabled'
51. tabela12a=tabela10
52. produt1, bio\_tecnologia1, pipeline1= atualiza1(filtra1(cb1,cb2,cb3,tabela10))
53. hab = 0
54. desabilita2 = 'disabled'
55. desabilita1 = 'disabled'
56. desabilita3 = ''
57. global valor10t
58. valor10t = tabela10.index.values[tabela10['Hybrid']==cb11a]
59. global tabelas10a
60. tabelas10a=filtra1(cb1,cb2,cb3,tabela10)
61. else:
62. cb1=cb11a
63. cb2=cb12b
64. cb3=cb13c
65. produt1, bio\_tecnologia1, pipeline1= atualiza1(filtra1(cb1,cb2,cb3,tabela10))
66. cb4 = request.form['cb4']
67. global cb4n
68. cb4n=cb4
69. cb5 = request.form['cb5']
70. story1 = request.form['story1']
71. rec1=cb5
72. info1=''+ nome +' - '+ daaltera
73. rquer1='required'
74. dados= tabela10.columns
75. dados=dados[3:(len(dados)-2)]
76. desabilita3 = 'disabled'
77. desabilita2 =''
78. desabilita1 =''
79. desabilita4 = 'disabled'
80. hab = 0
81. tabelas10 = tabelas10s
82. valor10t = tabelas10.index.values[tabelas10['Hybrid']==cb11a]
83. tabela10c=tabelas10.loc[valor10t]
84. tabela10c = adicionaitem(cb4,story1,tabela10c, info1)
85. tabelas10.loc[valor10t] = tabela10c
86. tabela12a=tabela10c
87. else:
88. cb2 = ''
89. cb1 = ''
90. cb3 = ''
91. cb11a = cb1
92. cb12b = cb2
93. cb13c = cb3
94. cb4 = ''
95. cb5 =''
96. dados= tabela10.columns
97. dados=dados[5:len(dados)]
98. produt1, bio\_tecnologia1, pipeline1= atualiza1(filtra1(cb1,cb2,cb3,tabela10))
99. tabela12a=filtra1(cb1,cb2,cb3,tabela10)
100. desabilita2 = 'disabled'
101. desabilita1 = 'disabled'
102. desabilita3 = ''
103. desabilita4 = 'disabled'
104. hab = 0
105. rec1='Descreva aqui a nota ou recomendação.'
106. rquer1=''
107. info1='Coloque a informação da mudança aqui. '+ daaltera
108. dados1=descricao['descricao']
109. return render\_template("paginacornote.html",produt1=produt1,bio\_tecnologia1=bio\_tecnologia1,pipeline1=pipeline1,cb2=cb2,cb1=cb1,cb3=cb3,cb4=cb4,dados=dados,
110. hab=hab, dados1=dados1,desabilita1=desabilita1,desabilita2=desabilita2,desabilita3=desabilita3,desabilita4=desabilita4, cb5=cb5, rec1=rec1,info1=info1, cursor1="",rquer1=rquer1,nome=nome )
112. def page3br():
113. a = len(tabela12a.index)
114. if a == 1:
115. valor1=tabela12a
116. valor1.columns = valor1.columns.str.lower().str.replace('\_', ' ')
117. valor1.head()
118. valor1=valor1.dropna(axis=1)
119. dados= valor1.columns
120. valor1=valor1.iloc[-1]
121. else:
122. valor1=""
123. dados=""
124. dados1=""
125. return render\_template("pagina\_de\_descricao.html",dados=dados, valor1=valor1)
127. def page4br():
128. descricao1t=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/D\_N\_RECtmpbr.csv')
129. descricao2t =pd.read\_csv('/domino/datasets/local/MD/DEREC\_NREC2.csv')
130. descricao2f=descricao1t.merge(descricao2t, how='left', suffixes=['', '\_'], indicator=True)
131. descricao2d=descricao2t.merge(descricao1t, how='left', suffixes=['', '\_'], indicator=True)
132. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
133. descricao2f['\_merge']=descricao2f['\_merge'].replace("left\_only",":Novo")
134. descricao2d=descricao2d[descricao2d['\_merge']=='left\_only']
135. descricao2d['\_merge']=descricao2d['\_merge'].replace("left\_only","Atual")
136. table=pd.concat([descricao2f, descricao2d])
137. return render\_template("NDRECSAIDAbr.html",table=[table.to\_html(classes='data', header="true")])
138. def page4abr():
139. descricao1t=pd.read\_csv('/domino/datasets/local/MD/DEREC\_NREC2.csv')
140. table=descricao1t[:100]
141. return render\_template("NDRECSAIDAbr.html",table=[table.to\_html(classes='data', header="true")])
143. def page5br():
144. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
145. descricao2t =pd.read\_excel('/domino/datasets/local/MD/rm\_valores.xlsx')
146. descricao2f=descricao1t.merge(descricao2t, how='left', suffixes=['', '\_'], indicator=True)
147. descricao2d=descricao2t.merge(descricao1t, how='left', suffixes=['', '\_'], indicator=True)
148. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
149. descricao2f['\_merge']=descricao2f['\_merge'].replace("left\_only",":Novo")
150. descricao2d=descricao2d[descricao2d['\_merge']=='left\_only']
151. descricao2d['\_merge']=descricao2d['\_merge'].replace("left\_only","Atual")
152. table=pd.concat([descricao2f, descricao2d])
153. return render\_template("rm\_valoresbrs.html",table=[table.to\_html(classes='data', header="true")])
155. def page6br():
156. descricao1t=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
157. descricao2t =pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')
158. descricao2f=descricao1t.merge(descricao2t, how='left', suffixes=['', '\_'], indicator=True)
159. descricao2d=descricao2t.merge(descricao1t, how='left', suffixes=['', '\_'], indicator=True)
160. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
161. descricao2f['\_merge']=descricao2f['\_merge'].replace("left\_only",":Novo")
162. descricao2d=descricao2d[descricao2d['\_merge']=='left\_only']
163. descricao2d['\_merge']=descricao2d['\_merge'].replace("left\_only","Atual")
164. table=pd.concat([descricao2f, descricao2d])
165. return render\_template("filtrobrs.html",table=[table.to\_html(classes='data', header="true")])
167. def page7br():
168. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
169. descricao2t =pd.read\_excel('/domino/datasets/local/MD/corpmonitor.xlsx')
170. descricao2f=descricao1t.merge(descricao2t, how='left', suffixes=['', '\_'], indicator=True)
171. descricao2d=descricao2t.merge(descricao1t, how='left', suffixes=['', '\_'], indicator=True)
172. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
173. descricao2f['\_merge']=descricao2f['\_merge'].replace("left\_only",":Novo")
174. descricao2d=descricao2d[descricao2d['\_merge']=='left\_only']
175. descricao2d['\_merge']=descricao2d['\_merge'].replace("left\_only","Atual")
176. table=pd.concat([descricao2f, descricao2d])
177. return render\_template("corpmonitorbrs.html",table=[table.to\_html(classes='data', header="true")])
179. def page8br():
180. descricao1t=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
181. descricao2t =pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
182. descricao2f=descricao1t.merge(descricao2t, how='left', suffixes=['', '\_'], indicator=True)
183. descricao2d=descricao2t.merge(descricao1t, how='left', suffixes=['', '\_'], indicator=True)
184. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
185. descricao2f['\_merge']=descricao2f['\_merge'].replace("left\_only",":Novo")
186. descricao2d=descricao2d[descricao2d['\_merge']=='left\_only']
187. descricao2d['\_merge']=descricao2d['\_merge'].replace("left\_only","Atual")
188. table=pd.concat([descricao2f, descricao2d])
189. return render\_template("cornnotebrs.html",table=[table.to\_html(classes='data', header="true")])
191. def editar2br():
192. if request.method == 'POST':
193. desabilita3 = 'disabled'
194. cb1=cb11a
195. cb2=cb12b
196. cb3=cb13c
197. cb4=''
198. cb5=''
199. dados= tabela10.columns
200. dados=dados[3:(len(dados)-2)]
201. desabilita3 = 'disabled'
202. desabilita2 =''
203. desabilita1 =''
204. desabilita4 = 'disabled'
205. rec1=''
206. info1=''
207. global hab2
208. hab2 = 1
209. produt1, bio\_tecnologia1, pipeline1= atualiza1(filtra1(cb1,cb2,cb3,tabela10))
210. global tabelas10s
211. tabelas10s=filtra1(cb11a, cb12b, cb13c, tabela10)
212. cursor1='autofocus'
213. rquer1='required'
214. else:
215. desabilita3 = ''
216. desabilita2 = 'disabled'
217. desabilita4 = ''
218. hab2 = 0
219. dados1=descricao['descricao']
220. return render\_template("paginacornote.html", produt1=produt1,bio\_tecnologia1=bio\_tecnologia1,pipeline1=pipeline1,cb2=cb2,cb1=cb1,cb3=cb3,cb4=cb4,dados=dados, hab2=hab2, dados1=dados1,desabilita1=desabilita1,
221. desabilita2=desabilita2,desabilita3=desabilita3,desabilita4=desabilita4,cb5=cb5, rec1=rec1,info1=info1,cursor1=cursor1,rquer1=rquer1 )
223. def salvar2br():
224. tabela10 = pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
225. #salva o valor atual da tablela para backup
226. tabela10ssalvar = tabela10.loc[valor10t]
227. tabela10ssalvar['Status']= 'old'
228. #salva o valor que foi alterado para a tabela atual
229. tabela10.loc[valor10t] = tabelas10s
230. notedescrip1=pd.concat([tabela10a, tabela10ssalvar])
231. notedescrip1=notedescrip1.sort\_values(by=['Hybrid'])
232. hab2 = 0
233. notedescrip1.to\_csv('/domino/datasets/local/MD/notedescrip1.csv',index = False)
234. return render\_template("salvo2.html")
236. def filtra1(p,b,pi,tabela13):
237. tabela12=tabela13
238. if p != '':
239. tabela12=tabela12[tabela12['Hybrid']== p]
240. else:
241. tabela12=tabela12
242. if b != '':
243. tabela12=tabela12[tabela12['Tecnologia']== b]
244. else:
245. tabela12=tabela12
246. if pi != '':
247. tabela12 = tabela12[tabela12['Pipeline'] == pi]
248. else:
249. tabela12 = tabela12
250. return tabela12
252. def atualiza1(tabela10a):
253. tabela12=tabela10a
254. produt1 = tabela12['Hybrid'].unique()
255. bio\_tecnologia1 = tabela12['Tecnologia'].unique()
256. pipeline1 = tabela12['Pipeline'].unique()
257. return produt1, bio\_tecnologia1, pipeline1
258. produt1, bio\_tecnologia1, pipeline1 = atualiza1(tabela10)
260. def adicionaitem(cb4,story1, tabela10c, info1):
261. if len(cb4) > 1:
262. if len(story1) > 1:
263. tabela10c[cb4] = story1
264. tabela10c['update']=info1
265. print(story1)
266. return tabela10c
268. def gestaoprodutosbr():
269. nome=request.headers.get("domino-username")
270. cb1=''
271. cb2=''
272. cb3=''
273. cb4=''
274. cb5=''
275. situacao=''
276. desabilita1=''
277. desabilita2='disabled'
278. desabilita4='disabled'
279. return render\_template("arquivogestao.html",cb2=cb2,cb1=cb1,cb3=cb3,cb4=cb4,cb5=cb5,nome=nome,produto1=" ",bio\_tecnologia="",pipeline="",marca=" ",desabilita1=desabilita1,situacao=situacao,
280. desabilita2=desabilita2,desabilita4=desabilita4 )
281. def salvarbr1():
282. datade=daaltera.replace(':','\_')
283. datade=datade.replace(' ','\_')
284. nome=request.headers.get("domino-username")
285. descricao1t=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/D\_N\_RECtmpbr.csv')
286. descricao1t.to\_excel('/domino/datasets/local/MD/exD\_N\_REC.xlsx', index = False)
287. descricao2t =pd.read\_csv('/domino/datasets/local/MD/DEREC\_NREC2.csv')
288. descricao2t.to\_csv('/domino/datasets/local/backup\_console/DEREC\_NREC2'+ nome + datade +'.csv',index = False)
289. descricao1t.to\_csv('/domino/datasets/local/MD/DEREC\_NREC2.csv',index = False)
290. return render\_template("salva1.html")
292. def salvarbr4():
293. datade=daaltera.replace(':','\_')
294. datade=datade.replace(' ','\_')
295. nome=request.headers.get("domino-username")
296. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
297. descricao2t =pd.read\_excel('/domino/datasets/local/MD/rm\_valores.xlsx')
298. descricao1t.to\_excel('/domino/datasets/local/MD/rm\_valores.xlsx', index = False)
299. descricao1t.to\_excel('/domino/datasets/local/MD/exrm\_valores.xlsx', index = False)
300. descricao2t.to\_excel('/domino/datasets/local/backup\_console/rm\_valores'+ nome + datade +'.xlsx',index = False)
301. return render\_template("salvo3.html")
303. def salvarbr5():
304. datade=daaltera.replace(':','\_')
305. datade=datade.replace(' ','\_')
306. nome=request.headers.get("domino-username")
307. descricao1dt=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
308. descricao1dt.to\_excel('/domino/datasets/local/MD/dadosparafiltro.xlsx', index = False)
309. descricaod2t =pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')
310. descricaod2t.to\_csv('/domino/datasets/local/backup\_console/dadosparafiltro'+ nome + datade +'.csv',index = False)
311. descricao1dt.to\_csv('/domino/datasets/local/MD/dadosparafiltro.csv',index = False)
312. descricaot = descricao1dt
313. descricao1n=pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
314. descricao1t =pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')
315. descricaot=descricaot[['Product','Biotecnologias\_disponiveis','Pipeline','Marca','situacao']]
316. descricaon=descricaot[['Product','Biotecnologias\_disponiveis','Pipeline','situacao']]
317. descricaon=descricaon.rename(columns={'Product':'Hybrid'})
318. descricaon=descricaon.rename(columns={'Biotecnologias\_disponiveis':'Tecnologia'})
319. descricao2n=descricaon
320. descricao2t=descricaot
321. descricao1nold=descricao1n[descricao1n['Status']=='old']
322. descricao1n=descricao1n[descricao1n['Status']!='old']
323. descricao1nf=descricao1n
324. descricao1old=descricao1t[descricao1t['Status']=='old']
325. descricao1t=descricao1t[descricao1t['Status']!='old']
326. descricao1f=descricao1t
327. #Atualiza plahnilha de doenças
328. descricao1t=descricao1t[['Product','Biotecnologias\_disponiveis','Pipeline','Marca']]
329. descricao2t=descricao2t[['Product','Biotecnologias\_disponiveis','Pipeline','Marca']]
330. descricao2f=descricao1t.merge(descricao2t, how='outer', suffixes=['', '\_'], indicator=True)
331. descricao2f=descricao2f[descricao2f['\_merge']=='left\_only']
332. descricao2f=descricao2f.drop(columns='\_merge')
333. descricao2f['situacao']= "desativado"
334. descricao2f=pd.concat([descricaot, descricao2f])
335. descricao1f=descricao1f.drop(columns='situacao')
336. descricao2final=descricao1f.merge(descricao2f, how='outer',on=['Product','Biotecnologias\_disponiveis','Pipeline','Marca'])
337. descricao2final=pd.concat([descricao2final, descricao1old])
338. descricao2final=descricao2final.drop\_duplicates()
339. # atualiza planilha do corne note
340. descricao1n=descricao1n[['Hybrid','Tecnologia','Pipeline']]
341. descricao2n=descricao2n[['Hybrid','Tecnologia','Pipeline']]
342. descricao2nf=descricao1n.merge(descricao2n, how='outer', suffixes=['', '\_'], indicator=True)
343. descricao2nf=descricao2nf[descricao2nf['\_merge']=='left\_only']
344. descricao2nf=descricao2nf.drop(columns='\_merge')
345. descricao2nf['situacao']= "desativado"
346. descricao2nf=pd.concat([descricaon, descricao2nf])
347. descricao1nf=descricao1nf.drop(columns='situacao')
348. descricao2nfinal=descricao1nf.merge(descricao2nf, how='outer',on=['Hybrid','Tecnologia','Pipeline'])
349. descricao2nfinal=pd.concat([descricao2nfinal, descricao1nold])
350. descricao2nfinal=descricao2nfinal.drop\_duplicates()
351. descricao2final.to\_csv('/domino/datasets/local/MD/doecasmcompleta.csv',index = False)
352. descricao2nfinal.to\_csv('/domino/datasets/local/MD/notedescrip1.csv',index = False)
353. descricao2nfinal.to\_excel('/domino/datasets/local/MD/exnotedescrip1.xlsx', index = False)
354. return render\_template("salvo3.html")
356. def salvarbr6():
357. datade=daaltera.replace(':','\_')
358. datade=datade.replace(' ','\_')
359. nome=request.headers.get("domino-username")
360. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
361. descricao2t =pd.read\_excel('/domino/datasets/local/MD/corpmonitor.xlsx')
362. descricao1t.to\_excel('/domino/datasets/local/MD/corpmonitor.xlsx', index = False)
363. descricao1t.to\_excel('/domino/datasets/local/MD/excorpmonitor.xlsx', index = False)
364. descricao2t.to\_excel('/domino/datasets/local/backup\_console/corpmonitor'+ nome + datade +'.xlsx',index = False)
365. return render\_template("salvo3.html")
367. def salvarbr7():
368. datade=daaltera.replace(':','\_')
369. datade=datade.replace(' ','\_')
370. nome=request.headers.get("domino-username")
371. descricao1t=pd.read\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
372. descricao1t.to\_excel('/domino/datasets/local/MD/exnotedescrip1.xlsx', index = False)
373. descricao2t=pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')
374. descricao2t.to\_csv('/domino/datasets/local/backup\_console/notedescrip1'+ nome + datade +'.csv',index = False)
375. descricao1t.to\_csv('/domino/datasets/local/MD/notedescrip1.csv',index = False)
376. return render\_template("salvo3.html")

brarquivo.py

1. from flask\_app import templates
2. from flask\_app import static
3. import pandas as pd
4. import os
5. from flask import Flask, render\_template, send\_file
6. from flask.globals import request
7. import time
8. daaltera=time.ctime()
9. DIRETORIO= "/domino/datasets/local/MD/"
10. def page2br():
11. return render\_template("NDRECbr.html")
13. def get\_arquivobr():
14. return send\_file('/domino/datasets/local/MD/doecasmcompleta.csv')
16. def get\_arquivo2br():
17. return send\_file('/domino/datasets/local/salvar\_arquivos/arquivo\_para\_md.xlsx')
19. def get\_arquivo3br():
20. return send\_file('/domino/datasets/local/MD/exnotedescrip1.xlsx')
22. def get\_arquivo4br():
23. return send\_file('/domino/datasets/local/MD/dadosparafiltro.xlsx')
25. def get\_arquivo5br():
26. return send\_file('/domino/datasets/local/MD/exD\_N\_REC.xlsx')
28. def get\_arquivo6br():
29. return send\_file('/domino/datasets/local/MD/exrm\_valores.xlsx')
31. def get\_arquivo7br():
32. return send\_file('/domino/datasets/local/MD/corpmonitor.xlsx')
34. def post\_arquivobr():
35. arquivo = request.files.get("meuArquivo")
36. print(arquivo)
37. nome\_do\_arquivo = arquivo.filename
38. os.remove('/domino/datasets/local/salvar\_arquivos/D\_N\_RECtmpbr.csv')
39. arquivo.save('/domino/datasets/local/salvar\_arquivos/D\_N\_REC.xlsx')
40. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/D\_N\_REC.xlsx')
41. descricao1t.to\_csv('/domino/datasets/local/salvar\_arquivos/D\_N\_RECtmpbr.csv',index = False)
42. os.remove('/domino/datasets/local/salvar\_arquivos/D\_N\_REC.xlsx')
43. return render\_template("NDRECs.html")
45. def post\_arquivobr1():
46. arquivo = request.files.get("meuArquivo")
47. print(arquivo)
48. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
49. arquivo.save('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
50. return render\_template("rm\_valores.html")
52. def post\_arquivobr2():
53. arquivo = request.files.get("meuArquivo")
54. print(arquivo)
55. nome\_do\_arquivo = arquivo.filename
56. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
57. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
58. arquivo.save('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
59. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
60. descricao1t.to\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv',index = False)
61. return render\_template("dadosparafiltro.html")
63. def post\_arquivobr3():
64. arquivo = request.files.get("meuArquivo")
65. print(arquivo)
66. nome\_do\_arquivo = arquivo.filename
67. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
68. arquivo.save('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
69. return render\_template("corpmonitor.html")
71. def post\_arquivobr4():
72. arquivo = request.files.get("meuArquivo")
73. print(arquivo)
74. nome\_do\_arquivo = arquivo.filename
75. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.csv')
76. os.remove('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
77. arquivo.save('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
78. descricao1t =pd.read\_excel('/domino/datasets/local/salvar\_arquivos/tempbr.xlsx')
79. descricao1t.to\_csv('/domino/datasets/local/salvar\_arquivos/tempbr.csv',index = False)
80. return render\_template("notedescrip1.html")

Atualiza\_bancobr.py

import hvac

import base64

import json

from google.oauth2 import service\_account

from google.cloud import bigquery

import pandas as pd

import pandas\_gbq

import sys

import os

import db\_dtypes

import pandas as pd

import numpy as np

from sklearn import linear\_model

import statsmodels.api as sm

import statsmodels.formula.api as smf

import statsmodels.stats.api as sms

def get\_secret\_from\_vault(approle\_id, approle\_secret, vault\_path):

client = hvac.Client(url='https://vault.agro.services')

#client.is\_authenticated()

client.auth\_approle(approle\_id, approle\_secret)

return client.read(vault\_path)

approle\_id = os.environ['APPROLEID']

approle\_secret=os.environ['APPROLESECRET']

secrets = get\_secret\_from\_vault(approle\_id = approle\_id,

approle\_secret = approle\_secret,

vault\_path='secret/csw/service-identities/md-latam-bq-viewers')

if 'data' in secrets and type(secrets['data']['data']) == str:

service\_account\_creds = json.loads(base64.b64decode(secrets['data']['data']))

else:

# in case credentials are saved directly as json object in vault (not encoded) you can get it directly

service\_account\_creds = secrets

bq\_credentials = service\_account.Credentials.from\_service\_account\_info(service\_account\_creds)

bq\_proj='bcs-market-dev-lake'

bq\_client = bigquery.Client(project = bq\_proj, credentials = bq\_credentials)

bq\_query = '''

SELECT \*

FROM latam\_datasets.stab\_brazil\_corn

'''

bqresults = pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

def comparativo():

descricao1 = pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')

descricao1['situacao']=descricao1['situacao'].str.lower()

descricao1=descricao1.loc[(descricao1['situacao'] == "ativo") | (descricao1['situacao'] == "lançamento")]

descricao1=descricao1[['Product','Biotecnologias\_disponiveis','Pipeline']]

descricao1['commercialName']=descricao1['Product']+descricao1['Biotecnologias\_disponiveis']

descricao1.to\_csv('/domino/datasets/local/MD/comparativobra.csv',index = False)

comparativo=descricao1

return comparativo

def stability():

bq\_query = '''

SELECT \*

FROM latam\_datasets.stab\_brazil\_corn

'''

data1= pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

data1['Variable'].unique()

data2=data1.query('Variable == "YLD"')

data2=data2.query('Protocol == "FTN"')

data2=data2.drop(columns=['Protocol','Country','Macroregion','Variable','Other','Other\_mean','updated\_at'])

data2=data2.drop\_duplicates()

data2['Head\_mean']=(data2['Head\_mean'].astype(float))\*100

data2['EI']=(data2['EI'].astype(float))\*100

data\_concorrente = pd.read\_excel("/domino/datasets/local/MD/concorrentes.xlsx")

#separa os hybridos que serão filtrados

descricao1 = pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')

descricao1['situacao']=descricao1['situacao'].str.lower()

descricao1=descricao1.loc[(descricao1['situacao'] == "ativo") | (descricao1['situacao'] == "lançamento")]

descricao1['produto']=(descricao1['Product']+descricao1['Biotecnologias\_disponiveis'])

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Winter","Subtropical Winter Winter")

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Summer","Subtropical Summer Summer")

safra1=descricao1['Pipeline'].unique()

descricao1['Marca']=descricao1['Marca'].str.upper()

safra=safra1.tolist()

dffinal=descricao1

fdf =dffinal

nomes = fdf['produto'].tolist()

pipeline = fdf['Pipeline'].unique().tolist()

fdf[fdf['Marca']=="LICENSING"]="BAYER"

fdf[fdf['Marca']=="Bayer"]="BAYER"

Marca =fdf['Marca'].unique().tolist()

# faz a união de dois frames

data\_marca=dffinal.query('Marca =="DEKALB" | Marca=="AGROESTE" | Marca=="AGROCERES"')

dfbayer = dffinal.query('Marca =="BAYER"')

Marca= ["DEKALB","AGROESTE","AGROCERES"]

dfferente=dffinal.query('Marca !="DEKALB" & Marca!="AGROESTE" & Marca!="AGROCERES" & Marca !="BAYER"')

#criar uma tabela com os hibridos concorrentes

concorrencia\_list=(data\_concorrente['Hybrids'].tolist() + dfferente['produto'].tolist())

ps4\_list=dfbayer['produto'].tolist()

#Filtrando final

lista\_final=(data\_marca['produto'].tolist() + concorrencia\_list)

data3=data2.loc[data2['Head'].isin(lista\_final)]

data3=data3.loc[data3['Pipeline'].isin(safra1)]

descricao1=descricao1.rename(columns = {'produto':'Head'})

descricao1=descricao1.rename(columns = {'produto':'Head'})

descricao1=descricao1.drop(columns=['Product','Biotecnologias\_disponiveis','Marca','fenotipe'])

data3=pd.merge(descricao1, data3, how = 'right', on = ['Head','Pipeline'])

data3['Pipeline']=data3['Pipeline'].str.replace("Subtropical Winter Winter","Subtropical Winter")

data3['Pipeline']=data3['Pipeline'].str.replace("Subtropical Summer Summer","Subtropical Summer")

data3['Pipeline']=data3['Pipeline'].str.replace(' ','\_')

data3=data3.loc[(data3['situacao'] == "ativo") | (data3['situacao'] == "lançamento")]

data3=data3.drop(columns=['situacao'])

data4 = data3

data4 = data4.rename(columns = {'Year':'Season','Pipeline':'Protocol','FIELD\_name':'Field','EI':'Environment','Head\_mean':'Check\_Mean'})

data4a=data4

data4a=data4a.replace(" ","NaN")

filtro=data4a[data4a['Environment']=="NaN"]

data4a=data4a.drop(filtro.index)

data4a.to\_csv('Stability\_Cultivio\_Brands\_1602.csv',index = False)

stability1=data4

data4a.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_Stability\_Cultivio\_Brands',

if\_exists = 'replace')

return stability1

def m\_b():

data4=stability()

data5=data4[['Protocol','Head']]

data5['valoresun']=(data5['Protocol'] +'\_'+ data5['Head'])

data5['ndata']=data5.groupby('valoresun')['valoresun'].transform('count')

datapoints=data5[['Protocol','Head','ndata']]

datapoints=datapoints.drop\_duplicates()

dfb=datapoints[datapoints['ndata']>4]

hybs5=dfb['Head'].unique().tolist()

data5=data4.loc[data4['Head'].isin(hybs5)]

pipe=data5['Protocol'].unique()

dffinal\_df = pd.DataFrame(columns=['Pipeline','commercialName','m','b','r2','p\_value'])

for o in range(len(pipe)):

df1=data5[data5['Protocol']==pipe[o]]

names = df1['Head'].unique()

for n in range(len(names)):

df2=df1[df1['Head']==names[n]]

df2['Check\_Mean']=df2['Check\_Mean'].fillna(0)

df2['Environment']=df2['Environment'].fillna(0)

y= pd.DataFrame(df2['Check\_Mean'])

X= pd.DataFrame(df2['Environment'])

results = smf.ols('Check\_Mean~Environment', data=df2).fit()

lm = linear\_model.LinearRegression()

model = lm.fit(X,y)

p\_value= results.pvalues[1]

r2=results.rsquared

b = lm.intercept\_

b=str(b)[1:-1]

m = lm.coef\_

m =str(m)[2:-2]

commercialName=df2['Head'].unique()

commercialName=str(commercialName)[2:-2]

Pipeline=df2['Protocol'].unique()

Pipeline=str(Pipeline)[2:-2]

dffinal\_df.loc[len(dffinal\_df)]=[Pipeline,commercialName,m,b,r2,p\_value]

dffinal\_df=dffinal\_df.dropna(thresh=3)

dffinal\_df.to\_csv('m\_b\_Cultivio\_Brands\_1602.csv',index = False)

m\_b1=dffinal\_df

m\_b1.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_m\_b\_Cultivio\_Brands',

if\_exists = 'replace')

return m\_b1

def phenotype():

#carrega o banco historico

bq\_query = '''

SELECT \*

FROM latam\_datasets.hss\_brazil\_historical\_corn

WHERE OBS\_observationRefCd IN ('YLD','MST','S50D','P50D','EHT','PHT','TKW') AND

protocolType IN ('FTN') AND

QC\_Flag is null

'''

#carrega o banco atual

data\_historical=pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

bq\_query ='''

SELECT \*

FROM latam\_datasets.hss\_brazil\_current\_corn

WHERE OBS\_observationRefCd IN ('YLD','MST','S50D','P50D','EHT','PHT','TKW') AND

protocolType IN ('FTN') AND

QC\_Flag is null

'''

data\_current=pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

data\_historical['OBS\_descriptorAbbreviation'].fillna('N/A', inplace=True)

data\_current['OBS\_descriptorAbbreviation'].fillna('N/A', inplace=True)

data\_historical['OBS\_code']=(data\_historical['OBS\_observationRefCd'] + "." + data\_historical['OBS\_descriptorAbbreviation'])

data\_current['OBS\_code']=(data\_current['OBS\_observationRefCd'] + "." + data\_current['OBS\_descriptorAbbreviation'])

data\_current=data\_current.drop(columns=['OBS\_observationRefCd'])

data\_current=data\_current.drop(columns=['OBS\_descriptorAbbreviation'])

data\_historical=data\_historical.drop(columns=['OBS\_observationRefCd'])

data\_historical=data\_historical.drop(columns=['OBS\_descriptorAbbreviation'])

data\_current=data\_current.drop(columns=['midas\_germplasm\_id'])

data\_historical=data\_historical.drop(columns=['midas\_germplasm\_id'])

a = data\_historical.columns.values.tolist()

data\_historical['OBS\_code']=data\_historical['OBS\_code'].replace("PHT.R1","PHT.N/A")

data\_historical['OBS\_code']=data\_historical['OBS\_code'].replace("PHT.VT","PHT.N/A")

code=["TKW.N/A","EHT.N/A","MST.N/A","YLD.N/A","S50D.N/A","PHT.N/A","P50D.N/A"]

data\_current=data\_current.loc[data\_current['OBS\_code'].isin(code)]

data\_historical=data\_historical.loc[data\_historical['OBS\_code'].isin(code)]

#data\_current=data\_current.loc[a]

brazil\_nonpivot=pd.concat([data\_current, data\_historical])

brazil\_nonpivot['new\_pipeline']=(brazil\_nonpivot['FIELD\_pipeline']+"\_"+brazil\_nonpivot['FIELD\_seasonName'])

brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace(' ','\_')

brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace("Tropical\_Lowlands\_Summer","Tropical\_Highlands\_Summer")

#brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace('\_',' ')

descricao1 = pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')

descricao1['commercialName']=(descricao1['Product']+descricao1['Biotecnologias\_disponiveis'])

prefixo1=descricao1['Biotecnologias\_disponiveis'].unique()

descricao1=descricao1.drop(columns=['Biotecnologias\_disponiveis'])

descricao1=descricao1.dropna(subset=['fenotipe'])

descricao1=descricao1.drop(columns=['Marca'])

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Winter","Subtropical Winter Winter")

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Summer","Subtropical\_Summer\_Summer")

descricao1['Pipeline']=descricao1['Pipeline'].str.replace(' ','\_')

descricao1=descricao1.rename(columns={'Pipeline': 'new\_pipeline'})

brazil\_nonpivot=pd.merge(descricao1,brazil\_nonpivot, how = 'left', on = ['commercialName','new\_pipeline'])

brazil\_nonpivot=brazil\_nonpivot[(brazil\_nonpivot['situacao']== "ativo") | (brazil\_nonpivot['situacao'] == "Lançamento")]

descricao1=descricao1.loc[(descricao1['situacao'] == "ativo") | (descricao1['situacao'] == "Lançamento")]

safra1=descricao1['new\_pipeline'].unique()

safra=safra1.tolist()

dffinal=descricao1

fdf =dffinal

nomes = fdf['commercialName'].tolist()

pipeline = fdf['new\_pipeline'].unique().tolist()

# faz a união de dois frames

#Filtrando final

lista\_final=(fdf['commercialName'].tolist())

pipes=safra

data3=brazil\_nonpivot

data4=data3

data4=brazil\_nonpivot

# ver como fazer Fenotipico

data4=data4[['new\_pipeline','commercialName','Product','OBS\_code','OBS\_numValue']]

data4=data4.loc[(data4['OBS\_code'] != "YLD.N/A") ]

data4=data4.loc[(data4['OBS\_code'] != "MST.N/A") ]

data4['OBS\_code']=data4['OBS\_code'].replace("TKW.N/A","TKW")

data4['OBS\_code']=data4['OBS\_code'].replace("EHT.N/A","EHT")

data4['OBS\_code']=data4['OBS\_code'].replace("S50D.N/A","S50D")

data4['OBS\_code']=data4['OBS\_code'].replace("P50D.N/A","P50D")

data4['OBS\_code']=data4['OBS\_code'].replace("PHT.N/A","PHT")

data4a=data4

#data4=data4.drop(columns=['Product'])

fenotipico=data4.groupby(['new\_pipeline','Product','commercialName','OBS\_code'])['OBS\_numValue'].mean

fenotipico=data4.pivot\_table(index=['commercialName','new\_pipeline','Product'], columns='OBS\_code',values='OBS\_numValue')

fenotipico.reset\_index(inplace=True)

fenotipico=fenotipico.round(0)

fenotipico = fenotipico.drop\_duplicates()

end\_piv= fenotipico

end\_piv['new\_pipeline']=end\_piv['new\_pipeline'].str.replace("Subtropical\_Summer\_Summer","Subtropical\_Summer")

end\_piv['new\_pipeline']=end\_piv['new\_pipeline'].str.replace("Subtropical\_Winter\_Winter","Subtropical\_Winter")

descricao3 = pd.read\_excel('/domino/datasets/local/MD/rm\_valores.xlsx')

descricao3=descricao3.drop(columns=['RM'])

descricao3=descricao3.drop\_duplicates()

descricao3=descricao3.rename(columns={'FIELD pipeline':'new\_pipeline'})

descricao3['new\_pipeline']= descricao3['new\_pipeline'].str.replace(' ','\_')

descricao3=pd.merge(end\_piv,descricao3, how = 'left', on = ['commercialName','new\_pipeline'])

end\_piv.to\_csv('phenotype22a.csv',index = False)

#mudança de comercial name para product

data4a['commercialName']=data4a['Product']

fenotipicoa=data4a.groupby(['new\_pipeline','commercialName','OBS\_code'])['OBS\_numValue'].mean

fenotipicoa=data4a.pivot\_table(index=['commercialName','new\_pipeline'], columns='OBS\_code',values='OBS\_numValue')

fenotipicoa.reset\_index(inplace=True)

fenotipicoa=fenotipicoa.round(0)

fenotipicoa = fenotipicoa.drop\_duplicates()

end\_piva= fenotipicoa

end\_piva['new\_pipeline']=end\_piva['new\_pipeline'].str.replace("Subtropical\_Summer\_Summer","Subtropical\_Summer")

end\_piva['new\_pipeline']=end\_piva['new\_pipeline'].str.replace("Subtropical\_Winter\_Winter","Subtropical\_Winter")

end\_piva.to\_csv('phenotype23a.csv',index = False)

rm1=pd.read\_excel("/domino/datasets/local/MD/rm\_valores.xlsx")

brazil\_nonpivot['OBS\_code']=brazil\_nonpivot['OBS\_code'].replace("MST.N/A","MST")

brazil\_nonpivot['OBS\_code']=brazil\_nonpivot['OBS\_code'].replace("YLD.N/A","YLD")

pipes1=brazil\_nonpivot['new\_pipeline'].unique()

pipes=pipes1.tolist()

data5r=rm1[['commercialName','new\_pipeline','RM']]

df9 = pd.DataFrame()

d=0

for o in range(len(pipes)):

dados1=brazil\_nonpivot[brazil\_nonpivot['new\_pipeline']==pipes[o]]

trait=["YLD","MST"]

dados2=dados1.loc[dados1['OBS\_code'].isin(trait)]

seasons=dados2['FIELD\_plantingSeason'].unique()

season=seasons.tolist()

for s in range(len(season)):

dados3=dados2.loc[dados2['FIELD\_plantingSeason']==season[s]]

data\_piv=dados3.pivot\_table(index=['FIELD\_plantingSeason','new\_pipeline','FIELD\_field\_latitude','FIELD\_field\_longitude','FIELD\_locationName','FIELD\_plantingDate','FIELD\_name','commercialName'], columns='OBS\_code',values='OBS\_numValue')

data\_piv.reset\_index(inplace=True)

data4=data\_piv[['FIELD\_plantingSeason','new\_pipeline','FIELD\_field\_latitude','FIELD\_field\_longitude','FIELD\_locationName','FIELD\_plantingDate','FIELD\_name','commercialName','MST']]

data4['new\_pipeline']=data4['new\_pipeline'].str.replace("Subtropical\_Summer\_Summer","Subtropical\_Summer")

data4['new\_pipeline']=data4['new\_pipeline'].str.replace("Subtropical\_Winter\_Winter","Subtropical\_Winter")

pipe1=data4['new\_pipeline'].unique()

data5=pd.merge(data4, data5r, how = 'left', on = ['commercialName','new\_pipeline'])

datapoints=data5

datapoints=datapoints.dropna(subset=['MST'])

datapoints['ndata']= datapoints.groupby('FIELD\_name')['FIELD\_name'].transform('count')

dfa=datapoints[datapoints['ndata']>=10]

bb=dfa['FIELD\_name'].unique()

data5=data5.loc[data5['FIELD\_name'].isin(bb)]

sim=data5

sim=sim.dropna(subset=['RM'])

sim['ndata']= sim.groupby('FIELD\_name')['FIELD\_name'].transform('count')

datapoints=sim

datapoints=datapoints.dropna(subset=['MST'])

dfa=datapoints[datapoints['ndata']>=4]

bb=dfa['FIELD\_name'].unique()

bb15=bb

dados5=data5.loc[data5['FIELD\_name'].isin(bb)]

dados15=dados5

if(dados5['FIELD\_name'].count() > 0):

Fields=dados5['FIELD\_name'].unique()

for i in range(len(Fields)):

fld=dados5[dados5['FIELD\_name']==Fields[i]]

values=fld['MST'].astype(float)

if(len(values) > 4):

fld=fld.dropna(subset=['MST'])

fld1=fld

fld1=fld1.dropna(subset=['RM'])

fld['RM']=fld['RM'].fillna(0)

fld['RM']=fld['RM'].astype(float)

y= pd.DataFrame(fld1['RM'])

X= pd.DataFrame(fld1['MST'])

results = smf.ols('MST~ RM', data=fld1).fit()

lm = linear\_model.LinearRegression()

model = lm.fit(X,y)

slope= lm.coef\_

#slope=str(slope)[2:-2]

slope=float(slope)

intercept=lm.intercept\_

#intercept=str(intercept)[1:-1]

intercept=float(intercept)

rsquare=results.rsquared

p\_value= results.pvalues[0]

fld['RM\_Calculated']=(slope\*fld['MST']+intercept)

fld['p\_value']=p\_value

fld['rsquare']=rsquare

df9=pd.concat([df9,fld])

d=d+1

df9['concat']=(df9['new\_pipeline'] +"\_"+ df9['FIELD\_plantingSeason'].astype(str) +"\_"+ df9['FIELD\_name'] )

check=df9.groupby(['new\_pipeline','FIELD\_plantingSeason','FIELD\_name']).mean(['rsquare','p\_value'])

check.reset\_index(inplace=True)

check['concat']=(check['new\_pipeline'] +"\_"+ check['FIELD\_plantingSeason'].astype(str) +"\_"+ check['FIELD\_name'] )

dfb=check[check['p\_value'] > 0.15]

dfbb=dfb['concat'].unique()

bem=df9.loc[df9['concat'].isin(dfbb)]

bem=bem[['commercialName','FIELD\_plantingSeason','new\_pipeline','FIELD\_name','RM\_Calculated','p\_value','rsquare']]

#mudanca de comercial name para product

for prefixo in prefixo1:

bem['commercialName']= bem['commercialName'].str.replace(prefixo,"")

#bem['commercialName']= bem['commercialName'].str.replace("PRO4","")

#bem['commercialName']= bem['commercialName'].str.replace("PRO2","")

bem['new\_pipeline']=bem['new\_pipeline'].str.replace("Subtropical\_Summer\_Summer","Subtropical\_Summer")

bem['new\_pipeline']=bem['new\_pipeline'].str.replace("Subtropical\_Winter\_Winter","Subtropical\_Winter")

final=bem.groupby(['new\_pipeline','commercialName']).mean(['RM\_Calculated'])

final.reset\_index(inplace=True)

final=final[['new\_pipeline','commercialName','RM\_Calculated']]

final1=final[final['RM\_Calculated'] < 145.0]

final1.to\_csv('rm\_calculated2a.csv',index = False)

rmcalculated=final1

descricao = descricao3

descricao1r = rmcalculated

descricao['commercialName']=descricao['Product']

descricao['new\_pipeline']= descricao['new\_pipeline'].str.replace(' ','\_')

descricao['new\_pipeline']=descricao['new\_pipeline'].replace("Subtropical\_Winter\_Winter","Subtropical\_Winter")

descricao['new\_pipeline']=descricao['new\_pipeline'].replace("Subtropical\_Summer\_Summer","Subtropical\_Summer")

descricao2=pd.merge(descricao, descricao1r, how = 'left', on = ['commercialName','new\_pipeline'])

#descricao2=pd.merge(descricao2, descricao3, how = 'left', on = ['commercialName','new\_pipeline'])

descricao2.drop\_duplicates()

descricao2['GDU\_S50']= descricao2['GDU\_S50'].fillna(-1)

descricao2['GDU\_P50']= descricao2['GDU\_P50'].fillna(-1)

descricao2['GDU\_S50']= descricao2['GDU\_S50'].astype(int)

descricao2['GDU\_P50']= descricao2['GDU\_P50'].astype(int)

descricao2['GDU\_S50']= descricao2['GDU\_S50'].astype(str)

descricao2['GDU\_P50']= descricao2['GDU\_P50'].astype(str)

descricao2['GDU\_S50']= descricao2['GDU\_S50'].str.replace('-1','')

descricao2['GDU\_P50']= descricao2['GDU\_P50'].str.replace('-1','')

descricao2.columns = ['commercialName','new\_pipeline','Product','EHT','P50D','PHT','S50D','TKW','gdus\_to\_emergence','GDU\_S50','GDU\_P50','gdus\_to\_maturity','GDU\_Emergence\_Flowering','Arquitetura\_Foliar','Tipo\_de\_Grao','Cor\_Grao','Empalhamento','Ciclo','Secagem','RM\_Calculated']

descricao2=descricao2[['commercialName','new\_pipeline','EHT','P50D','PHT','S50D','TKW','RM\_Calculated','gdus\_to\_emergence','GDU\_S50','GDU\_P50','gdus\_to\_maturity','GDU\_Emergence\_Flowering','Arquitetura\_Foliar','Tipo\_de\_Grao','Cor\_Grao','Empalhamento','Ciclo','Secagem']]

descricao2.to\_csv('Fenotype82a.csv',index = False)

descricao2.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_Fenotype',

if\_exists = 'replace')

return fenotipico,rmcalculated,dffinal,descricao2

def h2h():

bq\_query ='''

SELECT \*

FROM latam\_datasets.h2h\_brazil\_corn

'''

data1= pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

data2=data1[data1['Variable'] == "YLD"]

data2=data2[data2['Macroregion'] == "ALL"]

data2=data2[data2['Year'] == "ALL"]

data2=data2[data2['Protocol'] == "FTN"]

data2['Head\_mean']=data2['Head\_mean']\*100

data2['diffmean']=data2['diffmean']\*100

descricao1 = pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')

descricao1['Head']=(descricao1['Product']+descricao1['Biotecnologias\_disponiveis'])

descricao1=descricao1.drop(columns=['Biotecnologias\_disponiveis'])

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Winter","Subtropical Winter Winter")

descricao1['Pipeline']=descricao1['Pipeline'].str.replace("Subtropical Summer","Subtropical Summer Summer")

#lista=lista.rename(columns ={'brand':'Head\_brand'})

descricao1=descricao1.rename(columns ={'situacao':'Head\_situacao'})

descricao1=descricao1.rename(columns ={'Product':'Head\_p'})

descricao1['Marca']=descricao1['Marca'].str.upper()

lista=descricao1

lista=lista.rename(columns ={'Marca':'Head\_brand'})

data21=pd.merge(data2, lista, how = 'inner', on = ['Head','Pipeline'])

lista=lista.rename(columns ={'Head':'Other'})

lista=lista.rename(columns ={'Head\_brand':'Other\_brand'})

lista=lista.rename(columns ={'Head\_situacao':'Other\_situacao'})

lista=lista.rename(columns ={'Head\_p':'Other\_p'})

datafinal=pd.merge(data21, lista, how = 'inner', on = ['Other','Pipeline'])

dfremove=datafinal[datafinal['Head\_p']==datafinal['Other\_p']]

datafinal=datafinal.drop(dfremove.index)

dtafinalbay=datafinal[datafinal['Head\_brand']=="BAYER"]

dtafinaldk=datafinal[datafinal['Head\_brand']=="DEKALB"]

index\_dk=datafinal[datafinal['Head\_brand']=="DEKALB"].index

datafinal=datafinal.drop(index\_dk)

dtafinalas=datafinal[datafinal['Head\_brand']=="AGROESTE"]

index\_as=datafinal[datafinal['Head\_brand']=="AGROESTE"].index

datafinal=datafinal.drop(index\_as)

dtafinalag=datafinal[datafinal['Head\_brand']=="AGROCERES"]

index\_ag=datafinal[datafinal['Head\_brand']=="AGROCERES"].index

datafinal=datafinal.drop(index\_ag)

index\_dk=dtafinaldk[(dtafinaldk['Other\_brand']=="AGROESTE") | (dtafinaldk['Other\_brand']=="AGROCERES")].index

dtafinaldk=dtafinaldk.drop(index\_dk)

index\_ag=dtafinalag[(dtafinalag['Other\_brand']=="AGROESTE") | (dtafinalag['Other\_brand']=="DEKALB")].index

dtafinalag=dtafinalag.drop(index\_ag)

index\_as=dtafinalas[(dtafinalas['Other\_brand']=="AGROCERES") | (dtafinalas['Other\_brand']=="DEKALB")].index

dtafinalas=dtafinalas.drop(index\_as)

datafinal1=pd.concat([ dtafinaldk, dtafinalag,dtafinalas,dtafinalbay])

datafinal1=datafinal1[(datafinal1['Head\_situacao']== "ativo") | (datafinal1['Head\_situacao'] == "Lançamento")]

datafinal1['percent\_loss']=(100-datafinal1['percent\_wins'])

datafinal1= datafinal1[["Pipeline","EI","Head","Other","n","pvalue","Head\_mean","Other\_mean","diffmean","number\_wins","percent\_wins","percent\_loss"]]

datafinal1=datafinal1.sort\_values(['Head'])

datafinal1['Pipeline']=datafinal1['Pipeline'].str.replace("Subtropical Summer Summer","Subtropical Summer")

datafinal1['Pipeline']=datafinal1['Pipeline'].str.replace("Subtropical Winter Winter","Subtropical Winter")

datafinal1['Pipeline']=datafinal1['Pipeline'].str.replace(' ','\_')

datafinal1.to\_csv('H2H\_final0223a.csv',index = False)

h2hfinal = datafinal1

#h2hfinal.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_H2H',

#if\_exists = 'replace')

return h2hfinal

def filtradoencas():

tabela1 = pd.read\_csv('/domino/datasets/local/MD/doecasmcompleta.csv')

tabela1=tabela1.loc[tabela1['Status'] != "old"]

tabela1=tabela1.loc[(tabela1['situacao'] == "ativo") | (tabela1['situacao'] == "Lançamento")]

data\_dados=tabela1.drop\_duplicates()

data\_dados=data\_dados.iloc[:,0:15]

data\_dados['Pipeline']=data\_dados['Pipeline'].str.replace(' ','\_')

data\_dados=data\_dados.drop\_duplicates()

data\_dados.to\_csv('Diseases\_Tablea.csv',index = False)

Diseases\_Table=data\_dados

Diseases\_Table.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_Diseases\_Table',

if\_exists = 'replace')

return Diseases\_Table

def notadohibrido():

tabela1 = pd.read\_csv('/domino/datasets/local/MD/notedescrip1.csv')

tabela1=tabela1.loc[tabela1['Status'] != "old"]

tabela1=tabela1.loc[(tabela1['situacao'] == "ativo") | (tabela1['situacao'] == "Lançamento")]

data\_dados=tabela1.drop\_duplicates()

data\_dados=data\_dados.iloc[:,0:27]

data\_dados['Pipeline']=data\_dados['Pipeline'].str.replace(' ','\_')

data\_dados=data\_dados.drop\_duplicates()

data\_dados.to\_csv('Hybrid\_Descriptiona.csv',index = False)

hybrid\_descript=data\_dados

hybrid\_descript.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_Hybrid\_Description',

if\_exists = 'replace')

return hybrid\_descript

def nderc():

tabela1 = pd.read\_csv('/domino/datasets/local/MD/DEREC\_NREC2.csv')

tabela1.rename(columns = {'EI (kg/ha)': 'EI\_kg\_ha', 'DREC (plantas/m2)': 'DREC\_plantas\_m2'}, inplace = True)

tabela1.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_DEREC\_NREC',

if\_exists = 'replace')

return tabela1

def agronomicrisk():

bq\_query = '''

SELECT \*

FROM latam\_datasets.hss\_brazil\_historical\_corn

WHERE OBS\_observationRefCd IN ('YLD','MST','S50D','P50D','EHT','PHT','TKW') AND

protocolType IN ('FTN') AND

QC\_Flag is null

'''

#carrega o banco atual

data\_historical=pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

bq\_query ='''

SELECT \*

FROM latam\_datasets.hss\_brazil\_current\_corn

WHERE OBS\_observationRefCd IN ('YLD','MST','S50D','P50D','EHT','PHT','TKW') AND

protocolType IN ('FTN') AND

QC\_Flag is null

'''

data\_current=pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

data\_historical['OBS\_descriptorAbbreviation'].fillna('N/A', inplace=True)

data\_current['OBS\_descriptorAbbreviation'].fillna('N/A', inplace=True)

data\_historical['OBS\_code']=(data\_historical['OBS\_observationRefCd'] + "." + data\_historical['OBS\_descriptorAbbreviation'])

data\_current['OBS\_code']=(data\_current['OBS\_observationRefCd'] + "." + data\_current['OBS\_descriptorAbbreviation'])

data\_current=data\_current.drop(columns=['OBS\_observationRefCd'])

data\_current=data\_current.drop(columns=['OBS\_descriptorAbbreviation'])

data\_historical=data\_historical.drop(columns=['OBS\_observationRefCd'])

data\_historical=data\_historical.drop(columns=['OBS\_descriptorAbbreviation'])

data\_current=data\_current.drop(columns=['midas\_germplasm\_id'])

data\_historical=data\_historical.drop(columns=['midas\_germplasm\_id'])

a = data\_historical.columns.values.tolist()

data\_historical['OBS\_code']=data\_historical['OBS\_code'].replace("PHT.R1","PHT.N/A")

data\_historical['OBS\_code']=data\_historical['OBS\_code'].replace("PHT.VT","PHT.N/A")

code=["TKW.N/A","EHT.N/A","MST.N/A","YLD.N/A","S50D.N/A","PHT.N/A","P50D.N/A"]

data\_current=data\_current.loc[data\_current['OBS\_code'].isin(code)]

data\_historical=data\_historical.loc[data\_historical['OBS\_code'].isin(code)]

#data\_current=data\_current.loc[a]

brazil\_nonpivot=pd.concat([data\_current, data\_historical])

brazil\_nonpivot['new\_pipeline']=(brazil\_nonpivot['FIELD\_pipeline']+"\_"+brazil\_nonpivot['FIELD\_seasonName'])

brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace(' ','\_')

brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace("Tropical\_Lowlands\_Summer","Tropical\_Highlands\_Summer")

#brazil\_nonpivot['new\_pipeline']= brazil\_nonpivot['new\_pipeline'].str.replace('\_',' ')

descricao1 = pd.read\_csv('/domino/datasets/local/MD/dadosparafiltro.csv')

descricao1['commercialName']=(descricao1['Product']+descricao1['Biotecnologias\_disponiveis'])

descricao1=descricao1.drop(columns=['Biotecnologias\_disponiveis'])

descricao1=descricao1.drop(columns=['Marca'])

descricao1['Pipeline']=descricao1['Pipeline'].replace("Subtropical Winter","Subtropical Winter Winter")

descricao1['Pipeline']=descricao1['Pipeline'].replace("Subtropical Summer","Subtropical\_Summer\_Summer")

descricao1['Pipeline']=descricao1['Pipeline'].str.replace(' ','\_')

descricao1=descricao1.rename(columns={'Pipeline': 'new\_pipeline'})

brazil\_nonpivot=pd.merge(descricao1,brazil\_nonpivot, how = 'left', on = ['commercialName','new\_pipeline'])

brazil\_nonpivot=brazil\_nonpivot[(brazil\_nonpivot['situacao']== "ativo") | (brazil\_nonpivot['situacao'] == "Lançamento")]

descricao1=descricao1.loc[(descricao1['situacao'] == "ativo") | (descricao1['situacao'] == "Lançamento")]

safra1=descricao1['new\_pipeline'].unique()

safra=safra1.tolist()

dffinal=descricao1

fdf =dffinal

nomes = fdf['commercialName'].tolist()

pipeline = fdf['new\_pipeline'].unique().tolist()

# faz a união de dois frames

#Filtrando final

lista\_final=(fdf['commercialName'].tolist())

pipes=safra

data3=brazil\_nonpivot

data4=data3

data4=brazil\_nonpivot

data4=data4.loc[(data4['OBS\_code'] == "YLD.N/A") ]

output1 = pd.DataFrame()

for o in range(len(pipeline)):

dados1=data4[data4['new\_pipeline']==pipeline[o]]

eirgf = dados1['EIRange'].unique()

eirgb=[not pd.isnull(number) for number in eirgf]

eirg = eirgf[eirgb]

for s in range(len(eirg)):

dados2=dados1.loc[dados1['EIRange']==eirg[s]]

temp=dados2.groupby(['Product','new\_pipeline','FIELD\_plantingSeason','UniqueName','FIELD\_name']).mean(['OBS\_numValue'])

temp.reset\_index(inplace=True)

temp2 = pd.DataFrame()

temp2=temp[['Product','new\_pipeline']]

temp2['yld\_calc'] = temp['OBS\_numValue'] - temp['EI']

temp2['positive']=temp2['yld\_calc'].apply(lambda x: 0 if x <0 else 1)

temp2['npositive']=temp2.groupby('Product')['positive'].transform('sum')

temp2['ndata']=temp2.groupby('Product')['Product'].transform('count')

temp2['prop\_win']=(temp2['npositive'] / temp2['ndata'])\*100

temp2['prop\_win']=temp2['prop\_win'].round(2)

temp2['EIRange']=eirg[s]

temp2['averageEI']=temp.groupby('Product')['EI'].transform('mean')

output1=pd.concat([output1,temp2])

#output1.to\_csv('agronomictotal.csv',index = False)

output2=output1[['Product','new\_pipeline','npositive','prop\_win','EIRange','averageEI']]

output2=output2.drop\_duplicates()

output2['new\_pipeline']=output2['new\_pipeline'].str.replace('\_',' ')

output2['new\_pipeline']=output2['new\_pipeline'].str.replace("Subtropical Winter Winter","Subtropical Winter")

output2['new\_pipeline']=output2['new\_pipeline'].str.replace("Subtropical Summer Summer","Subtropical Summer")

output2['new\_pipeline']=output2['new\_pipeline'].str.replace(' ','\_')

output2.reset\_index(inplace=True)

output2=output2.drop(columns=['index'])

output2=output2.drop\_duplicates()

output2.to\_csv('agronomicrisk.csv',index = False)

agronomicrisk=output2

agronomicrisk.to\_gbq(project\_id = bq\_proj, credentials = bq\_credentials,destination\_table = 'bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_Agronomicrisk',

if\_exists = 'replace')

return agronomicrisk

def geraplanilha()

listas=['''Diseases\_Table''','''DEREC\_NREC''','''Hybrid\_Description''','''Fenotype''','''Agronomicrisk''','''m\_b\_Cultivio\_Brands''','''Stability\_Cultivio\_Brands''','''H2H''','''Crop\_Monitoring''','''TT\_cumulative''']

w=pd.ExcelWriter("/domino/datasets/local/salvar\_arquivos/arquivo\_para\_md.xlsx")

for lista in listas:

bq\_query = '''

SELECT \*

FROM bcs-market-dev-lake.latam\_md\_corn.BRA\_T\_'''+ lista

df=pd.read\_gbq(bq\_query, project\_id = bq\_proj, credentials = bq\_credentials, use\_bqstorage\_api = True)

nome=str(lista)

print(nome)

df.to\_excel(w,sheet\_name=nome,index=False)

df=pd.read\_csv('/domino/datasets/local/MD/comparativo.csv')

df.to\_excel(w,sheet\_name='comparativo',index=False)

w.save()

return df