Tupinamba

**Banco de Dados**

O banco de dados utilizado neste sistema é o MySQL 8.0.31, possuindo apenas uma única tabela que armazena as informações necessárias para ativação.

A tabela funciona como uma tabela auxiliar, a cada novo retrabalho é necessário deletar todas as chaves contidas nela e armazenar chaves novas escolhidas para aquele novo retrabalho.

Script SQL:

Criando a base de dados com o nome “dto\_keys”, se houver outra base de dados já existente com o mesmo nome, o comando não irá funcionar

CREATE DATABASE dto\_keys;

Colocando a base de dados em uso, sempre que abrir o BD para realizar qualquer tipo de manipulação é necessário executar este comando.

USE dto\_keys;

Criando a tabela no banco de dados com o nome “general\_keys”, contendo cinco colunas

CREATE TABLE IF NOT EXISTS dto\_keys.general\_keys(

idkey INT(11) NOT NULL AUTO\_INCREMENT,

keycontent VARCHAR(29) NOT NULL,

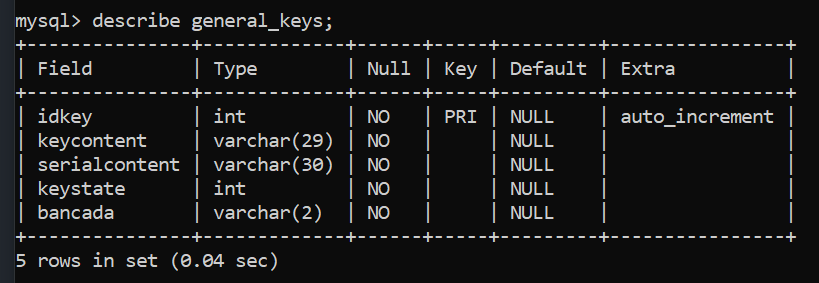
serialcontent VARCHAR(30) NOT NULL,

keystate INT(1) NOT NULL,

bancada VARCHAR(2) NOT NULL,

PRIMARY KEY (idkey)

);



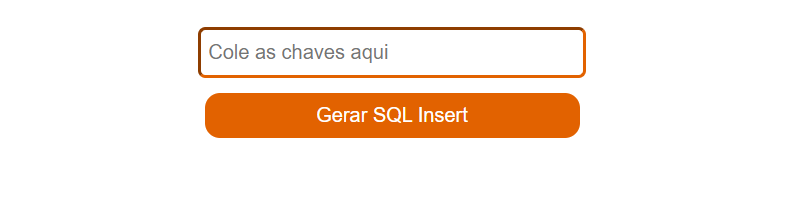
|  |  |  |
| --- | --- | --- |
| **Nome da Coluna** | **Tipo** | **Descrição** |
| idkey | INTEGER | Coluna que identifica uma linha na tabela (conjunto de dados), tornando ela única no banco de dados |
| keycontent | VARCHAR(29) | Coluna que armazena dados do tipo texto, permitindo no máximo 29 caracteres, tamanho exato de uma chave de ativação Windows |
| serialcontent | VARCHAR(30) | Coluna que armazena dados do tipo texto, permitindo no máximo 30 caracteres, o tamanho dos seriais variam de acordo com a montadora |
| keystate | INTEGER | Coluna que armazena o status de uma chave, sendo identificada por um único número (0 – Disponível, 1 – Em uso, 2 – Bloqueada, 3 - Ativada) |
| bancada | VARCHAR(2) | Coluna que identifica a bancada a qual utilizou uma chave e teve sucesso na ativação, tipo texto com dois caracteres sendo B (bancada) seguido do índice, exemplo (B1) bancada de número um |

**Programa de Inserção de Chaves**

O programa criado para gerar um script SQL de inserção de chaves se chama InsertKeys, uma página simples que pode ser aberta no navegador, de preferência o Google Chrome, foi desenvolvido em Javascript e na sua execução ele monta uma query com os campos já formatados para serem inseridos no MySQL, pode ser inserido via Workbench, phpMyAdmin ou MySQL Command Line Client.

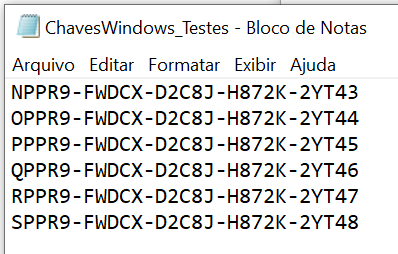
1. Abrindo o programa

Basta abrir o arquivo index com o navegador de preferência e surgirá uma página como esta



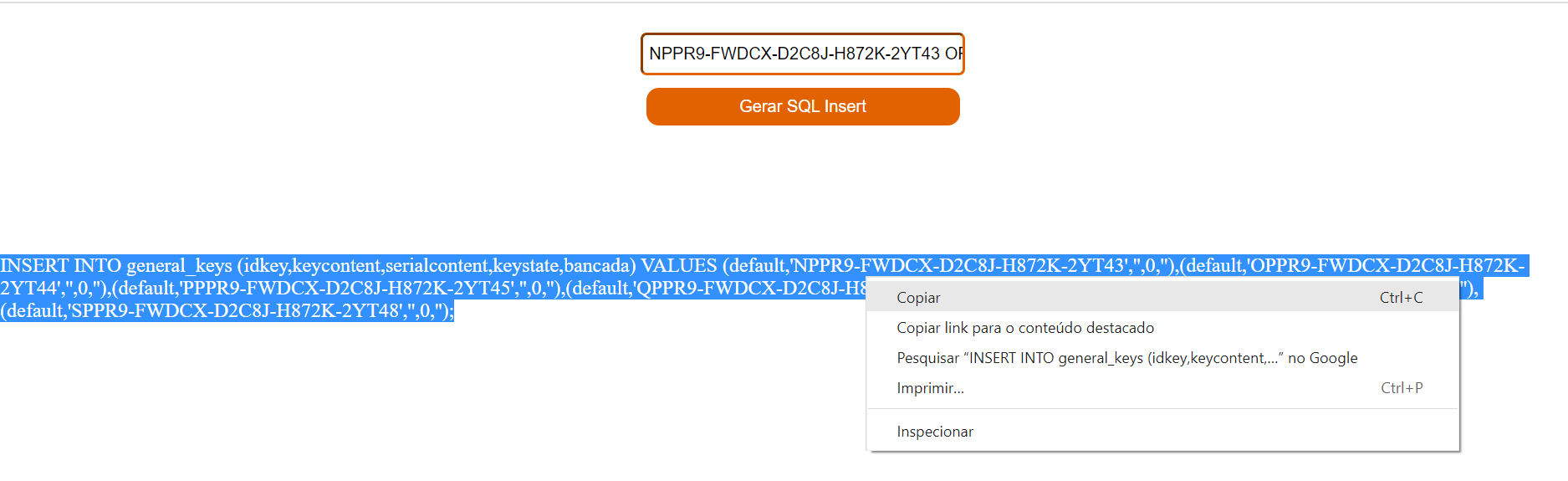
1. Colando as chaves e gerando o script SQL

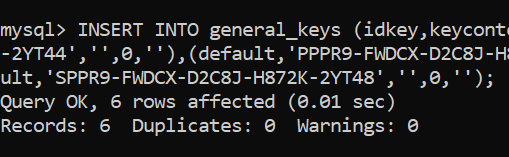
As chaves precisam estar em um arquivo de texto, uma abaixo da outra sem pontuações, pois o programa separa cada chave através do espaço contido entre as quebras de linha, após isso basta copiar do arquivo de texto, colar no input do programa e clicar no botão “Gerar SQL Insert”



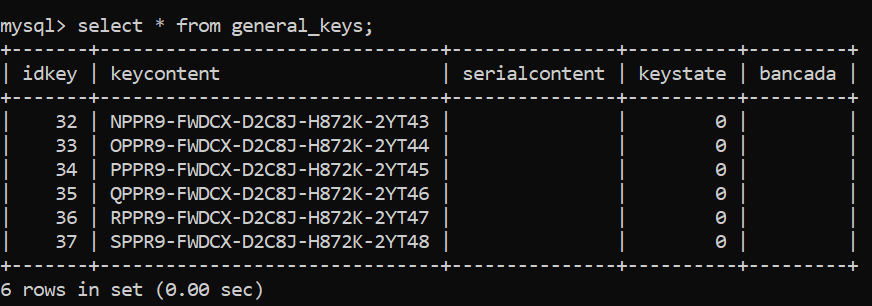
1. Script SQL gerado

Após isso o Script é gerado, basta copiar e executar dentro do banco de dados





Após a inserção é retornado “Query OK”, ou seja, teve sucesso na execução e “6 rows affected”, ou seja, seis chaves inseridas na tabela, dando um select o resultado deverá ser igual a imagem abaixo.



**Manipulação do Banco de Dados**

Colocar banco de dados em uso

use dto\_keys;

Selecionar todos os dados da tabela

select \* from general\_keys;

Selecionar apenas chaves ativadas

select \* from general\_keys where keystate=3;

Selecionar apenas chaves bloqueadas

select \* from general\_keys where keystate=2;

Selecionar apenas chaves disponíveis para uso

select \* from general\_keys where keystate=0;

Selecionar apenas chaves em uso

select \* from general\_keys where keystate=1;

Esvaziar tabela para novo retrabalho

delete from general\_keys where idkey > 0;

Trocar status em uso para disponível

update general\_keys set keystate=0 where keystate=1;

Inserindo uma única chave

INSERT INTO general\_keys (idkey,keycontent,serialcontent,keystate,bancada) VALUE (default,'9GT2P-WND82-6DWDP-X3CQF-W8F4A','',0,'');

Para inserir mais de uma chave é recomendado utilizar o programa InsertKeys

Selecionar tabela ao final do retrabalho (Ativadas)

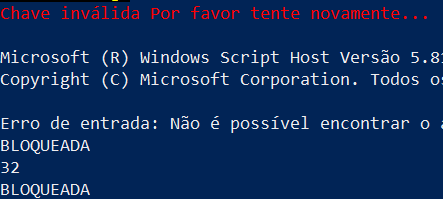
SELECT keycontent,serialcontent FROM general\_keys WHERE keystate=3;

Selecionar tabela ao final do retrabalho (Bloqueadas)

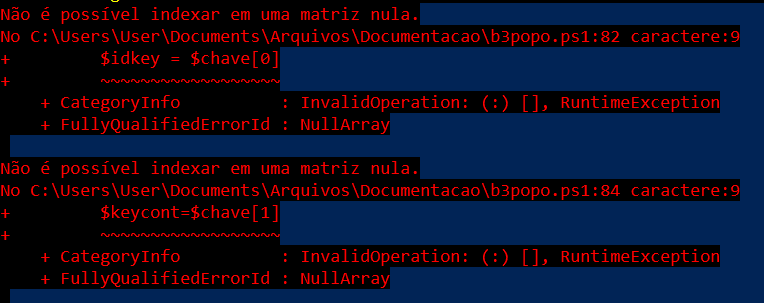
SELECT keycontent FROM general\_keys WHERE keystate=2;

**Mensagens retornadas no PowerShell**

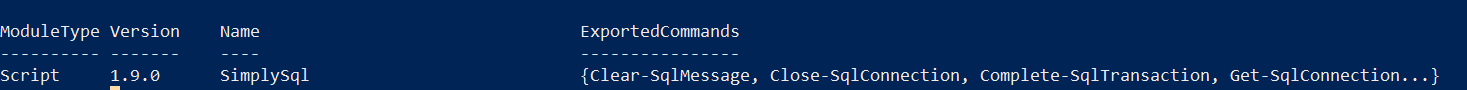
Quando a chave estiver bloqueada será mostrado uma mensagem como esta, “BLOQUEADA” e o ID dessa chave

****

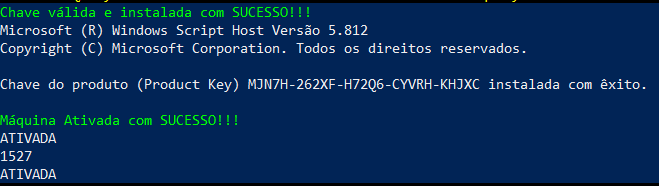
Quando aparecer uma mensagem como esta abaixo, significa que no banco de dados não possui mais chaves disponíveis, “Não é possível indexar em uma matriz nula”, porque houve a requisição no BD e nenhuma chave foi retornada, com isso não é possível preencher a matriz

****

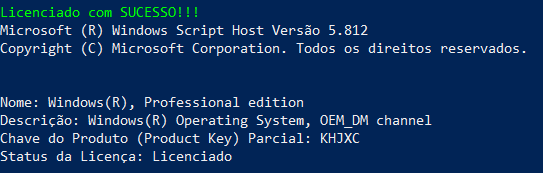
Mensagem mostrada no início da execução do programa, informa a versão do conector, o nome, no caso deste programa é o “SimplySql” e mostra os comandos que o módulo importa

****

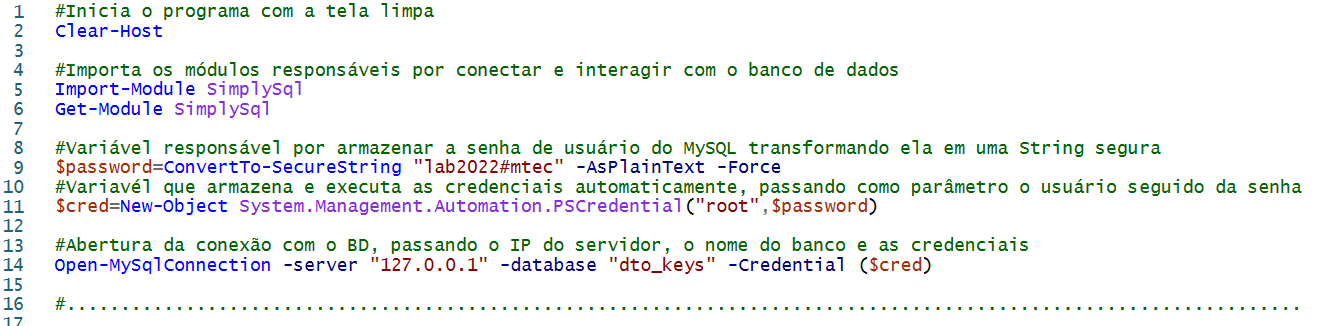
Mensagem que informa que a chave foi ativada com sucesso na máquina, mostrando o texto “ATIVADA” e o ID da chave

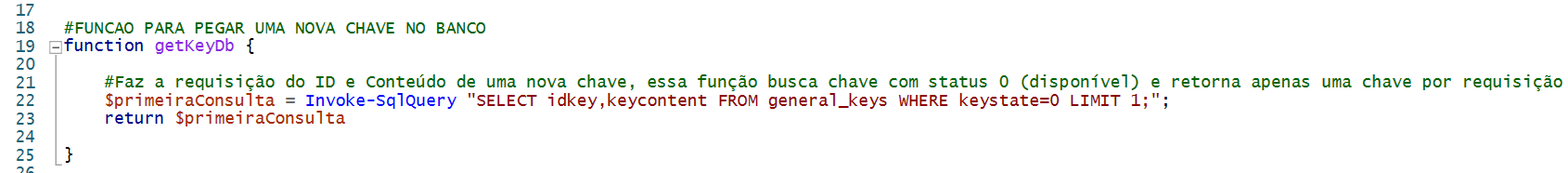
****

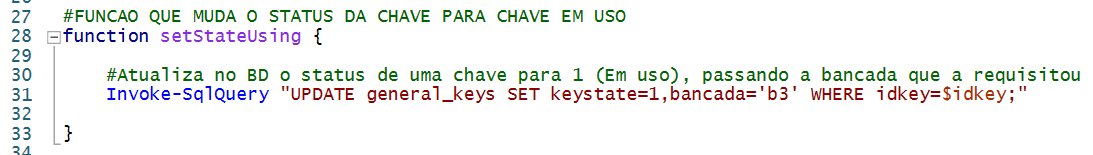
Mensagem que informa o licenciamento da chave com sucesso, mostrando a versão do Windows, a parcial da chave e o status licenciado

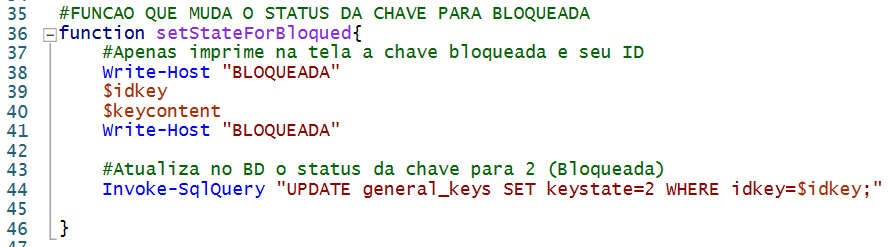
****

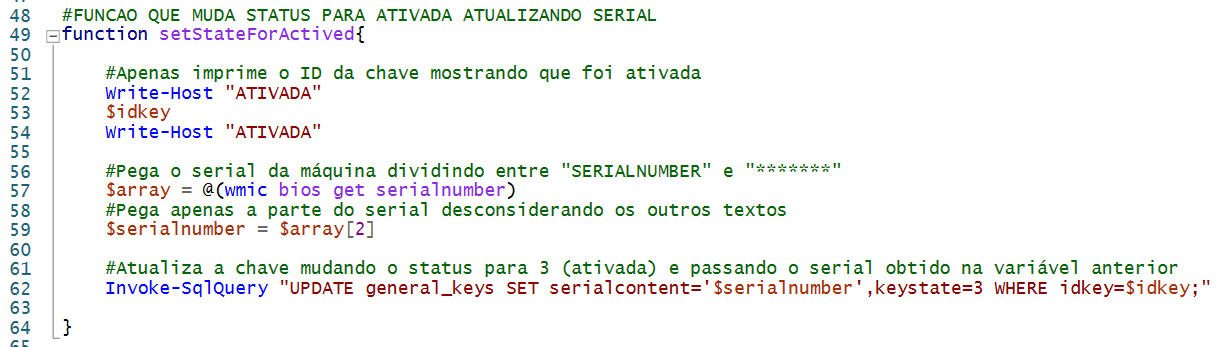
**Imagens do Script PowerShell**

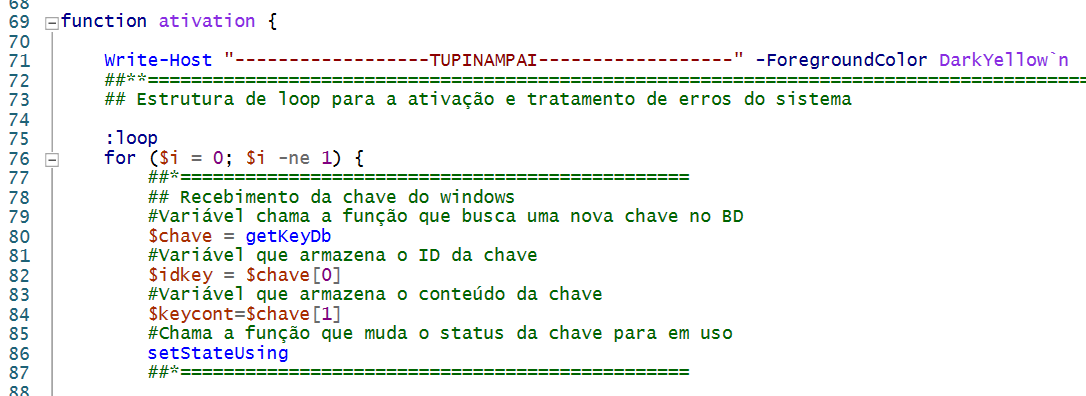
****

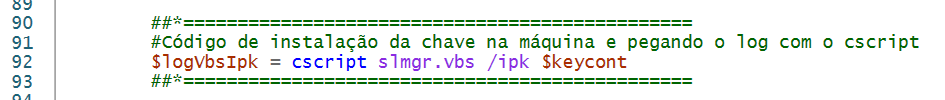
****

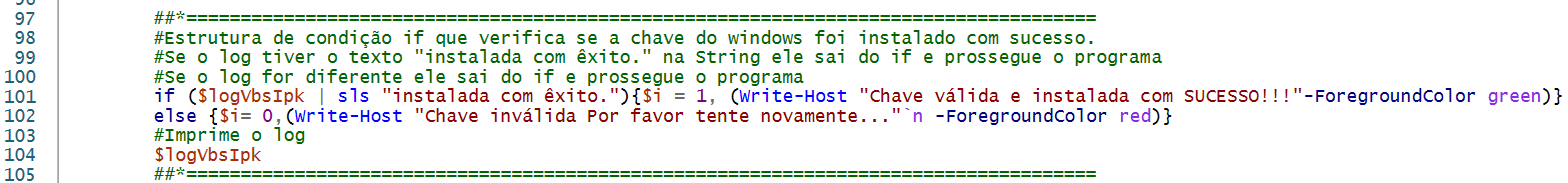
****

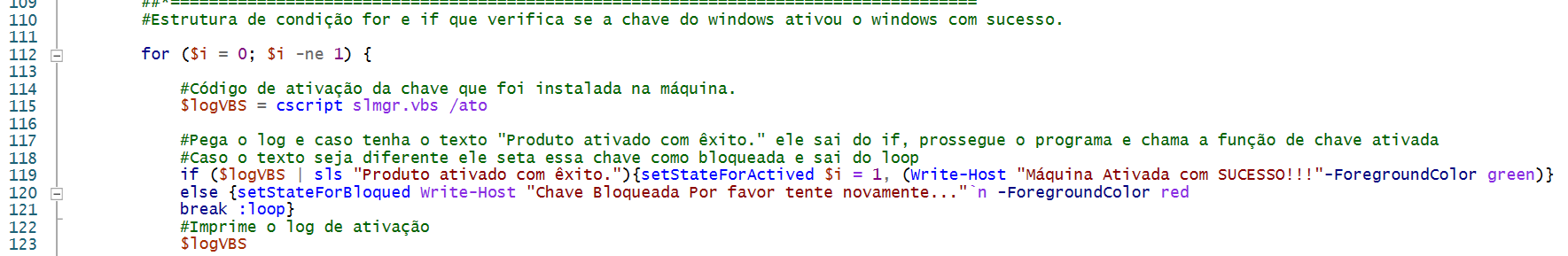
****

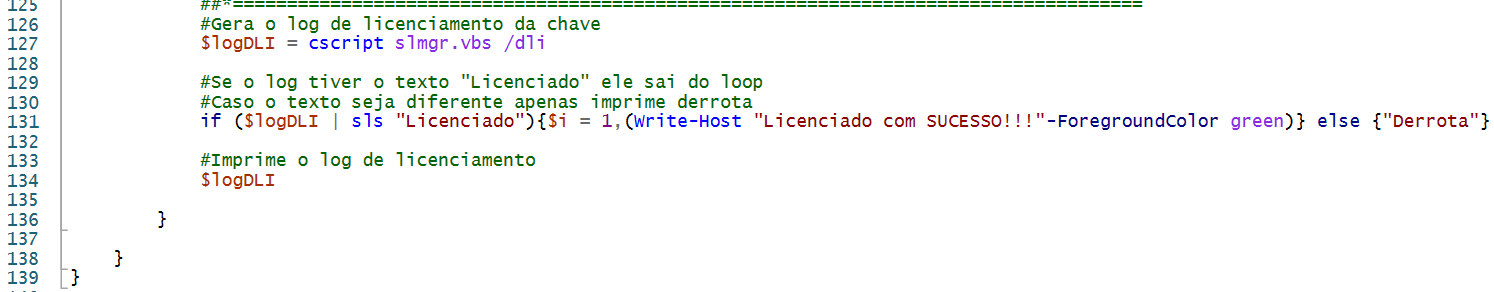
****

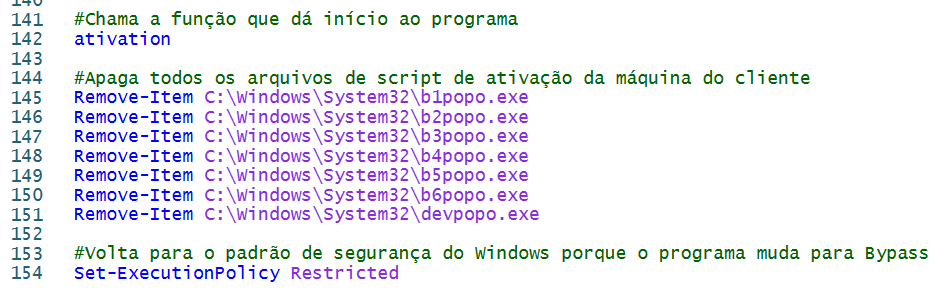
****

****

****

****

****

****

**Código do Script PowerShell**

#Inicia o programa com a tela limpa

Clear-Host

#Importa os módulos responsáveis por conectar e interagir com o banco de dados

Import-Module SimplySql

Get-Module SimplySql

#Variável responsável por armazenar a senha de usuário do MySQL transformando ela em uma String segura

$password=ConvertTo-SecureString "lab2022#mtec" -AsPlainText -Force

#Variavél que armazena e executa as credenciais automaticamente, passando como parâmetro o usuário seguido da senha

$cred=New-Object System.Management.Automation.PSCredential("root",$password)

#Abertura da conexão com o BD, passando o IP do servidor, o nome do banco e as credenciais

Open-MySqlConnection -server "127.0.0.1" -database "dto\_keys" -Credential ($cred)

#..................................................................................................................

#FUNCAO PARA PEGAR UMA NOVA CHAVE NO BANCO

function getKeyDb {

#Faz a requisição do ID e Conteúdo de uma nova chave, essa função busca chave com status 0 (disponível) e retorna apenas uma chave por requisição

$primeiraConsulta = Invoke-SqlQuery "SELECT idkey,keycontent FROM general\_keys WHERE keystate=0 LIMIT 1;";

return $primeiraConsulta

}

#FUNCAO QUE MUDA O STATUS DA CHAVE PARA CHAVE EM USO

function setStateUsing {

#Atualiza no BD o status de uma chave para 1 (Em uso), passando a bancada que a requisitou

Invoke-SqlQuery "UPDATE general\_keys SET keystate=1,bancada='b3' WHERE idkey=$idkey;"

}

#FUNCAO QUE MUDA O STATUS DA CHAVE PARA BLOQUEADA

function setStateForBloqued{

#Apenas imprime na tela a chave bloqueada e seu ID

Write-Host "BLOQUEADA"

$idkey

$keycontent

Write-Host "BLOQUEADA"

#Atualiza no BD o status da chave para 2 (Bloqueada)

Invoke-SqlQuery "UPDATE general\_keys SET keystate=2 WHERE idkey=$idkey;"

}

#FUNCAO QUE MUDA STATUS PARA ATIVADA ATUALIZANDO SERIAL

function setStateForActived{

#Apenas imprime o ID da chave mostrando que foi ativada

Write-Host "ATIVADA"

$idkey

Write-Host "ATIVADA"

#Pega o serial da máquina dividindo entre "SERIALNUMBER" e "\*\*\*\*\*\*\*"

$array = @(wmic bios get serialnumber)

#Pega apenas a parte do serial desconsiderando os outros textos

$serialnumber = $array[2]

#Atualiza a chave mudando o status para 3 (ativada) e passando o serial obtido na variável anterior

Invoke-SqlQuery "UPDATE general\_keys SET serialcontent='$serialnumber',keystate=3 WHERE idkey=$idkey;"

}

#..................................................................................................................

function ativation {

Write-Host "------------------TUPINAMPAI------------------" -ForegroundColor DarkYellow`n

##\*\*===========================================================================================================================

## Estrutura de loop para a ativação e tratamento de erros do sistema

:loop

for ($i = 0; $i -ne 1) {

##\*===============================================

## Recebimento da chave do windows

#Variável chama a função que busca uma nova chave no BD

$chave = getKeyDb

#Variável que armazena o ID da chave

$idkey = $chave[0]

#Variável que armazena o conteúdo da chave

$keycont=$chave[1]

#Chama a função que muda o status da chave para em uso

setStateUsing

##\*===============================================

##\*===============================================

#Código de instalação da chave na máquina e pegando o log com o cscript

$logVbsIpk = cscript slmgr.vbs /ipk $keycont

##\*===============================================

##\*====================================================================================

#Estrutura de condição if que verifica se a chave do windows foi instalado com sucesso.

#Se o log tiver o texto "instalada com êxito." na String ele sai do if e prossegue o programa

#Se o log for diferente ele sai do if e prossegue o programa

if ($logVbsIpk | sls "instalada com êxito."){$i = 1, (Write-Host "Chave válida e instalada com SUCESSO!!!"-ForegroundColor green)}

else {$i= 0,(Write-Host "Chave inválida Por favor tente novamente..."`n -ForegroundColor red)}

#Imprime o log

$logVbsIpk

##\*====================================================================================

##\*====================================================================================

#Estrutura de condição for e if que verifica se a chave do windows ativou o windows com sucesso.

for ($i = 0; $i -ne 1) {

#Código de ativação da chave que foi instalada na máquina.

$logVBS = cscript slmgr.vbs /ato

#Pega o log e caso tenha o texto "Produto ativado com êxito." ele sai do if, prossegue o programa e chama a função de chave ativada

#Caso o texto seja diferente ele seta essa chave como bloqueada e sai do loop

if ($logVBS | sls "Produto ativado com êxito."){setStateForActived $i = 1, (Write-Host "Máquina Ativada com SUCESSO!!!"-ForegroundColor green)}

else {setStateForBloqued Write-Host "Chave Bloqueada Por favor tente novamente..."`n -ForegroundColor red

break :loop}

#Imprime o log de ativação

$logVBS

##\*====================================================================================

#Gera o log de licenciamento da chave

$logDLI = cscript slmgr.vbs /dli

#Se o log tiver o texto "Licenciado" ele sai do loop

#Caso o texto seja diferente apenas imprime derrota

if ($logDLI | sls "Licenciado"){$i = 1,(Write-Host "Licenciado com SUCESSO!!!"-ForegroundColor green)} else {"Derrota"}

#Imprime o log de licenciamento

$logDLI

}

}

}

#Chama a função que dá início ao programa

ativation

#Apaga todos os arquivos de script de ativação da máquina do cliente

Remove-Item C:\Windows\System32\b1popo.exe

Remove-Item C:\Windows\System32\b2popo.exe

Remove-Item C:\Windows\System32\b3popo.exe

Remove-Item C:\Windows\System32\b4popo.exe

Remove-Item C:\Windows\System32\b5popo.exe

Remove-Item C:\Windows\System32\b6popo.exe

Remove-Item C:\Windows\System32\devpopo.exe

#Volta para o padrão de segurança do Windows porque o programa muda para Bypass

Set-ExecutionPolicy Restricted

**Criação**

**PASSO A PASSO PARA REALIZAR A IMAGEM COM O SCRIPT DE ATIVAÇÃO DO WINDOWS**