```
strQuery = "SELECT * FROM nomedatabela WHERE nome = '" & SafeSqlLiteral(txtInput.Text, 2)
& "'";

Imports:
<%@ import Namespace="System" %>
<%@ import Namespace="System.Text.RegularExpressions" %>

Código do método SafeSqlLiteral chamado no código SQL:

public string SafeSqlLiteral(System.Object theValue, System.Object theLevel){

    // intLevel representa o quão completo o valor vai ser verificado para códigos
maliciosos
    // intLevel (1) - Faz o básico. Este nível vai contrariar a maior parte dos ataques
SQL injection.
    // intLevel (2) - Vai ser adicionado à maioria das palavras usadas nas queryes SQL
para previnir o acesso não autorizado ao banco de dados. É seguro para ser  imprimido de
volta pra o código HTML. Mas não pode ser usado para usuários ou senhas.

    string strValue = (string)theValue;
    int intLevel = (int)theLevel;

    if (strValue != null) {
        if (intLevel > 0) {
            strValue = strValue.Replace("'", "''"); // Muito importante! Somente esta
linha pode prevenir muitos dos ataques SQL.
            strValue = strValue.Replace("--", "");
            strValue = strValue.Replace("[", "[[]");
            strValue = strValue.Replace("%", "[%]");
        }
        if (intLevel > 1) {
            string[] myArray = new string[] { "xp_ ","update ","insert ","select ","drop
","alter ","create ","rename ","delete ","replace "};
            int i = 0;
            int i2 = 0;
            int intLenghtLeft = 0;
            for (i = 0; i < myArray.Length; i++){
                string strWord = myArray[i];
                Regex rx = new Regex(strWord, RegexOptions.Compiled |
RegexOptions.IgnoreCase);
                MatchCollection matches = rx.Matches(strValue);
                i2 = 0;
                foreach (Match match in matches) {
                    GroupCollection groups = match.Groups;
                    intLenghtLeft = groups[0].Index + myArray[i].Length + i2;
                    strValue = strValue.Substring(0, intLenghtLeft - 1) + " " +
strValue.Substring(strValue.Length - (strValue.Length - intLenghtLeft), strValue.Length -
intLenghtLeft);
                    i2 += 5;
                }
            }
        }
        return strValue;
    }
```

```
    else {
        return strValue;
    }
}
```