3

O Algoritmo utilizado para calcular o digito de verificação (Check Digit) de Números de Cartão de Crédito é o [Algoritmo de Luhn](https://en.wikipedia.org/wiki/Luhn_algorithm), porém não basta apenas validar a informação sem levar em conta as características de um número de Cartão de Crédito que são:

* Possuir somente números
* Ter entre 12 e 19 dígitos (na verdade não há CC com 17 dígitos)
* Os 6 primeiros dígitos devem indicar um IIN (Issuer identification number) válido. Ele é chamado de BIN no meio bancário.
* Passar na validação do Algoritmo de Luhnn

A verificação do item **BIN** necessita de uma base de informações para comparação. Não existe uma lógica na geração dos BINs. Você pode ter faixas por exemplo que tem saltos e dentro destes saltos o BIN pertecer a uma instituição diferente, inclusive de país diferente. É um clássico erro acreditar que todo cartão que comece com 4 é VISA e com 5 seja Mastercard, por exemplo. Atualmente no Brasil isso é até válido, pois temos poucas empresas de Cartão o que acaba reservando o 4 e o 5 apenas para estas duas Bandeiras.

Deixando estes aspectos de lado, pois não há como validar um BIN via código sem que seja feita algum tipo de consulta, a validação pode ser feita com o seguinte algoritmo

* Verificar se o tamanho do numero do cartão está entre 12 e 19. Atualmente no Brasil é raro ter um cartão diferente de 16. Alguns Hipercard tinham 19 mas já estão saindo com 16 também.
* Verificar se o valor possui somente números
* Verificar se o número passa pela validação do Algoritmo de Luhnn

Abaixo um código em C#. Ele é apenas uma forma de entender pois já há nas versões atuais do ASP.net, validação por **Attributes** para esta verificação, [conforme resposta do @Randrade](https://pt.stackoverflow.com/a/97239/2999)

O Algoritmo de Luhnn está representado na [resposta do @Cigano Morrison Mendez](https://pt.stackoverflow.com/questions/97234/validar-n%C3%BAmero-cart%C3%A3o-de-cr%C3%A9dito/164503" \l "answer-97241) com a ressalva que na última linha, onde está

Mod 10: 85, módulo 10 = 5 (último dígito do cartão)

Deveria estar

Mod 10: 85, módulo 10 = 5, Check Digit = 10 - 5 = 5 (último dígito da sequencia)

Na verdade, não há a necessidade de calcular o *check digit*. Basta incluir o termo na soma e calcular o módulo 10 dela. Se for zero é válido.

// About the Algorithm

/\*\*

@See https://en.wikipedia.org/wiki/Luhn\_algorithm

Steps:

1 - From the rightmost Digit of a Numeric String, Double the value of every digit on odd positions

2 - If the obtained value is greather than 9, subtract 9 from it

3 - Sum all values

4 - Calculate the Modulus of the value on 10 basis, if is zero so the String has a Luhnn Check Valid

\*\*/

public static bool IsValidLuhnn(string val) {

// About the Algorithm

/\*\*

@See https://en.wikipedia.org/wiki/Luhn\_algorithm

Steps:

1 - From the rightmost Digit of a Numeric String, Double the value of every digit on odd positions

2 - If the obtained value is greather than 9, subtract 9 from it

3 - Sum all values

4 - Calculate the Modulus of the value on 10 basis, if is zero so the String has a Luhnn Check Valid

\*\*/

public static bool IsValidLuhnn(string val) {

int currentDigit;

int valSum = 0;

int currentProcNum = 0;

for (int i = val.Length-1; i >= 0; i--) {

//parse to int the current rightmost digit, if fail return false (not-valid id)

if(!int.TryParse(val.Substring(i,1), out currentDigit))

return false ;

currentProcNum = currentDigit << (1 +i & 1);

//summarize the processed digits

valSum += (currentProcNum > 9 ? currentProcNum - 9 : currentProcNum);

}

// if digits sum is exactly divisible by 10, return true (valid), else false (not-valid)

// valSum must be greater than zero to avoid validate 0000000...00 value

return (valSum > 0 && valSum % 10 == 0) ;

}

public static bool isValidCreditCardNumber(string cc) {

// rule #1, must be only numbers

if (cc.All(Char.IsDigit) == false) {

return false;

}

// rule #2, must have at least 12 and max of 19 digits

if (12 > cc.Length || cc.Length > 19) {

return false;

}

// rule #3, must pass Luhnn Algorithm

return IsValidLuhnn(cc);

}