

CODI FICANT INFORMACIÓ

DE LA TEORIA DE CODIS
A LA SEGURETAT DE LES TARGETES BANCÀRIES

Berta Ventura Güell

TdR 2025 - 2026

Tutor: Lluís Alemany

Assessora UAB: Mercè Villanueva

Institut Josep Brugulat

«Soc de la mena de persones que
estimen el perquè de les coses»

— *Caterina la Gran*

SÍNTESI

Aquest treball analitza el funcionament de les targetes bancàries i com la codificació permet detectar errors, amb especial atenció a l'algoritme de Luhn. La part teòrica proporciona una comprensió detallada dels fonaments matemàtics, els codis detectors i correctors d'errors, i les seves aplicacions en la vida quotidiana. En la part pràctica es crea una aplicació en Python amb interfície gràfica que comprova la validesa dels números de targeta i calcula els dígitos que falten, aplicant l'algoritme de Luhn de manera directa. Finalment, tot el coneixement adquirit es comparteix a través d'una entrada a la Viquipèdia.

SÍNTESIS

Este trabajo analiza el funcionamiento de las tarjetas bancarias y como la codificación permite detectar errores, con especial atención al algoritmo de Luhn. La parte teórica proporciona una comprensión detallada de los fundamentos matemáticos, los códigos detectores y correctores de errores, y sus aplicaciones en la vida cotidiana. En la parte práctica se crea una aplicación en Python con interfaz gráfica que comprueba la validez de los números de tarjeta y calcula los dígitos que faltan, aplicando el algoritmo de Luhn de manera directa. Finalmente, todo el conocimiento adquirido se comparte a través de una entrada en la Wikipedia.

ABSTRACT

This work analyzes the working of bank cards and how coding allows to detect errors, with special attention to the Luhn algorithm. The theoretical part provides a detailed understanding of mathematical foundations, error detection and correction codes, and their applications in everyday life. In the practice part, an application is created in Python with a graphical interface that checks the validity of card numbers and calculates the missing digits, applying Luhn's algorithm directly. Finally, all the knowledge acquired is shared through an entry in Wikipedia.

AGRAÏMENTS

Abans de començar amb aquest treball, m'agradaria expressar el meu agraïment a totes les persones que, amb la seva ajuda, han fet possible la realització d'aquest projecte.

En primer lloc, a la meva família, per tot el suport incondicional durant l'elaboració del treball i per proporcionar-me la motivació necessària en els moments més difícils.

També vull agrair a totes les persones que han aportat un granet de sorra amb suggeriments, crítiques constructives i una lectura atenta dels textos, que han contribuït a millorar el resultat final.

Un reconeixement especial al programa Argó de la UAB, i particularment a la Mercè Villanueva, per l'assessorament proporcionat. La seva experiència i consells han estat de gran ajuda per al desenvolupament del treball.

Finalment, però no menys important, voldria agrair al meu tutor, Lluís Alemany, per la seva dedicació, orientació i implicació en la resolució de tots els dubtes que han anat sorgint al llarg del procés.

ÍNDEX

1. INTRODUCCIÓ.....	11
1.1. Motivació.....	11
1.2. Objectius.....	11
1.3. Metodologia.....	12
2. MARC TEÒRIC.....	13
2.1. Introducció a la codificació.....	13
2.1.1. La informació i la seva transmissió.....	13
2.1.2. Els errors de transmissió.....	13
2.1.3. Dades codificades.....	14
2.1.4. Detecció i correcció d'errors.....	15
2.2. Tipus de codis de detecció i correcció d'errors.....	15
2.2.1. Codis detectors d'errors.....	16
2.2.1.1. Bit de paritat.....	16
2.2.1.2. Codis ISBN, IBAN i EAN.....	17
2.2.1.3. CRC (Cyclic Redundancy Check).....	17
2.2.2. Codis correctors d'errors.....	18
2.2.2.1 Exemples de codis correctors d'errors.....	18
Codi de repetició.....	18
Codi de Hamming.....	19
Codi BCH.....	20
2.3. Fonaments matemàtics.....	21
2.3.1. Aritmètica modular.....	21
2.4. Exemples de codis a la vida quotidiana.....	23
2.4.1.DNI.....	24
2.4.2. ISBN.....	24
2.4.3. IBAN i CCC.....	26
2.4.4. EAN (codi de barres).....	28
2.3.5. CODABAR (targetes bancàries).....	30
2.5. Les matemàtiques darrere de les targetes bancàries.....	30

2.5.1. Què hi ha dins una targeta de crèdit?.....	30
2.5.2. L'algoritme de Luhn: com es valida una targeta.....	33
3. MARC PRÀCTIC.....	36
3.1. Creació del programa.....	36
3.1.1. Cerca d'informació i programari.....	36
3.1.2. Estructura del producte.....	37
3.1.3. Fases de l'elaboració del codi.....	39
3.1.4. Codi del programa.....	41
3.1.4.1. Bases de la programació amb Python i Tkinter.....	41
3.1.4.2. Funcionament del codi.....	47
3.1.4.3. Producte final.....	51
3.2. Anàlisi del programa.....	52
3.2.1. Anàlisi matemàtic de la detecció d'errors amb l'algoritme de Luhn.....	52
3.2.1.1. Substitució d'un sol dígit.....	52
3.2.1.2. Transposició de dos dígit consecutius.....	53
3.2.1.3. Transposició de dos dígit en posicions d'igual paritat.....	56
3.2.2. Comprovació del funcionament del programa.....	57
3.2.2.1. Pantalla del verificador.....	57
3.2.2.2. Pantalla de la calculadora.....	58
3.3. Difusió a la Viquipèdia.....	58
4. CONCLUSIONS.....	59
5. BIBLIOGRAFIA I WEBGRAFIA.....	61
6. ANNEXOS.....	64
Annex I : Fragment del codi de la funció comprovar en el verificador.....	64
Annex II : Fragment del codi de la funció comprovar en la calculadora.....	66
Annex III : Codi del programa.....	68
Annex IV : Captures del funcionament de la pantalla del verificador.....	77
Annex V : Captures del funcionament de la pantalla de la calculadora.....	81
Annex VI : Entrada de l'algoritme de Luhn a la Viquipèdia.....	90

ÍNDEX FIGURES

Figura 1. Sistema de codificació digital [8] i [17].....	14
Figura 2. Les parts d'un ISBN de 10 dígit (a la part superior) i el corresponent EAN-13 (a la part inferior) i codi de barres.....	25
Figura 3. Exemple de l'anvers d'una targeta de crèdit.....	31
Figura 4. Exemple del revers d'una targeta de crèdit.....	31
Figura 5. Captura del resultat final de la pantalla d'inici del programa.....	37
Figura 6. Captura del resultat final del verificador del programa.....	38
Figura 7. Captura del resultat final de la calculadora de dígit del programa.....	39
Figura 8. Diagrama de flux dels subprogrames.....	40
Figura 9. Captura de la pantalla del verificador cas 1.....	77
Figura 10. Captura de la pantalla del verificador cas 2.....	77
Figura 11. Captura de la pantalla del verificador cas 3.....	78
Figura 12. Captura de la pantalla del verificador cas 4.....	78
Figura 13. Captura de la pantalla del verificador cas 5.....	79
Figura 14. Captura de la pantalla del verificador cas 6.....	79
Figura 15. Captura de la pantalla del verificador cas 7.....	80
Figura 16. Captura de la pantalla de la calculadora de dígit cas 1.....	81
Figura 17. Captura de la pantalla de la calculadora de dígit cas 2.....	81
Figura 18. Captura de la pantalla de la calculadora de dígit cas 3.....	82
Figura 19. Captura de la pantalla de la calculadora de dígit cas 4.....	82
Figura 20. Captura de la pantalla de la calculadora de dígit cas 5.....	83
Figura 21. Captura de la pantalla de la calculadora de dígit cas 6.....	83

Figura 22. Captura de la pantalla de la calculadora de dígit cas 7.....	84
Figura 23. Captura de la pantalla de la calculadora de dígit cas 8.....	84
Figura 24. Captura de la pantalla de la calculadora de dígit cas 9.....	85
Figura 25. Captura de la pantalla de la calculadora de dígit cas 10.....	85
Figura 26. Captura de la pantalla de la calculadora de dígit cas 11.....	86
Figura 27. Captura de la pantalla de la calculadora de dígit cas 12.....	86
Figura 28. Captura de la pantalla de la calculadora de dígit cas 13.....	87
Figura 29. Captura de la pantalla de la calculadora de dígit cas 14.....	87
Figura 30. Captura de la pantalla de la calculadora de dígit cas 15.....	88
Figura 31. Captura de la pantalla de la calculadora de dígit cas 16.....	88
Figura 32. Captura de la pantalla de la calculadora de dígit cas 17.....	89
Figura 33. Captura de la pantalla de la calculadora de dígit cas 18.....	89

ÍNDEX TAULES

Taula 1. Taula del càlcul del bit de paritat en funció del tipus de paritat.....	16
Taula 2. Taula de correcció d'errors amb codi de repetició i vot majoritari.....	19
Taula 3. Taula amb els mòduls habituals en codificació.....	22
Taula 4. Taula amb alguns tipus d'errors que es poden donar en un codi.....	23
Taula 5. Taula de correspondència del residu amb la lletra.....	24
Taula 6. Taula de correspondència de cada lletra amb el nombre.....	27
Taula 7. Taula de correspondència del primer dígit en les targetes emeses per una companyia bancària.....	32
Taula 8. Taula de l'exemple de validació d'un número de targeta amb l'aplicació de l'algoritme de Luhn.....	34

Taula 9. Taula dels operadors en el llenguatge Python.....	44
Taula 10. Taula dels símbols que corresponen als diferents condicionals.....	44
Taula 11. Taula dels valors que obtenen els díigits d'un nombre de targeta segons la seva posició al aplicar-los l'algoritme de Luhn.....	53
Taula 12. Taula dels valors que obtenen els díigits quan se'ls aplica l'algoritme de Luhn.....	55
Taula 13. Taula d'anàlisi de casos al utilitzar el verificador del programa.....	57

1. INTRODUCCIÓ

1.1. Motivació

Sempre he estat una persona indecisa a l'hora de triar camins o prendre decisions, però hi ha una cosa que he tingut clara des de petita: m'apassionen les matemàtiques. M'agrada entendre com funcionen les coses, buscar patrons, resoldre problemes... Per això, quan vaig descobrir l'àmbit de la teoria de codificació, em va cridar l'atenció. Em va sorprendre que una branca de les matemàtiques pogués tenir aplicacions tan pràctiques i alhora tan discretes en el nostre dia a dia, com ara en les targetes bancàries.

Aquest treball em permetrà aprofundir en un camp que és nou per a mi i, a més, crec que em pot ajudar a desenvolupar habilitats útils per al meu futur, sigui quin sigui el camí que acabi escollint.

1.2. Objectius

- Comprendre i descriure la teoria de la codificació.
- Saber quins codis detectors i correctors d'errors existeixen i entendre més específicament el codi utilitzat en les targetes de crèdit.
- Explorar el paper de les matemàtiques en la seguretat de les targetes bancàries.
- Comprovar per quin tipus d'errors el codi detector utilitzat en les targetes bancàries funciona bé.
- Saber calcular el dígit de control de les targetes bancàries donades la resta de xifres.
- Saber calcular un dígit qualsevol d'un número de targeta donades la resta de xifres.
- Difondre el coneixement adquirit relacionat amb els fonaments matemàtics darrere dels números de les targetes de crèdit.

1.3. Metodologia

El treball s'inicia amb una part teòrica que serveix per després desenvolupar una part pràctica.

En un principi es fa una recerca d'informació a partir dels conceptes claus de la temàtica del treball utilitzant cercadors com *Worldcat* i *Google Académico*, així com articles, llibres i recursos divulgatius relacionats amb la teoria de codificació i els seus fonaments matemàtics; al llarg del text, les fonts utilitzades es citen entre claudàtors, de manera que es pugui identificar clarament l'origen de cada informació.

Aquesta cerca permet establir una base teòrica estructurada en diferents apartats: des d'una introducció a la codificació fins als fonaments matemàtics, els principals tipus de codis detectors i correctors d'errors i exemples d'aplicacions en la vida quotidiana, aprofundint i centrant la recerca al funcionament de les targetes bancàries.

Un cop establerta aquesta base teòrica, s'inicia la part pràctica del treball. Es dissenya i programa una aplicació en llenguatge Python amb una interfície gràfica creada amb la llibreria Tkinter. Aquest programa permet verificar la validesa d'un número de targeta i calcular el dígit que falta en un número incomplet, aplicant directament l'algoritme de Luhn. Al mateix temps, es du a terme una anàlisi dels errors que l'algoritme és capaç de detectar i dels casos en què pot fallar, comprovant així la seva eficàcia.

Finalment, per tal de difondre el coneixement adquirit sobre l'algoritme de Luhn al llarg de la recerca, s'elabora i publica una entrada a la Viquipèdia amb l'objectiu de posar aquesta informació a l'abast de tothom.

2. MARC TEÒRIC

2.1. Introducció a la codificació

2.1.1. La informació i la seva transmissió

En l'àmbit científic i tecnològic, el concepte d'informació no es refereix només a paraules o textos, sinó a qualsevol dada que pugui ser transmesa, emmagatzemada o processada. Per exemple, una imatge digital, un número de compte bancari, la identificació d'un producte o la seqüència de bits d'una cançó en format MP3 són formes diferents d'informació.

La informació resulta útil si es pot comunicar entre diferents persones, dispositius o sistemes. Aquesta transmissió d'informació es pot produir de moltes maneres, però actualment és molt important la comunicació digital a través d'internet, de xarxes mòbils, de senyals de ràdio o de cables físics.

Així doncs, s'està transmetent informació quan s'envia un missatge per correu electrònic, quan es fa una transferència bancària o quan s'escaneja el codi de barres d'un producte en un supermercat. Ara bé, durant aquest procés es poden transferir errors i provocar que la informació arribi modificada o incompleta, la qual cosa pot tenir conseqüències greus si la dada transmesa és sensible o crítica [8].

2.1.2. Els errors de transmissió

Un error de transmissió és una alteració no desitjada que es produeix quan una dada viatja des del seu origen fins a la seva destinació. Aquests errors poden presentar-se de formes diverses: es pot modificar un dígit, ometre un caràcter, afegir-ne un de nou o intercanviar dues xifres de lloc [17].

Les causes d'aquestes alteracions poden ser molt variades:

- Sorolls o interferències a la línia de transmissió (com ara ones de ràdio, defectes físics en el cablejat...).
- Errors humans, tant en l'escriptura com en la lectura de codis manuals.
- Deteriorament dels suports d'emmagatzematge, com ara CD, DVD o discs durs.
- Mal funcionament dels dispositius electrònics que intervenen en el procés de transmissió.

L'objectiu principal en la transmissió d'informació és transmetre les dades de manera eficient segura i fiable, de forma que es detectin els errors i, en alguns casos, es corregeixin. Per garantir aquesta fiabilitat, s'utilitzen tècniques matemàtiques de codificació que permeten verificar si la informació rebuda és correcta.

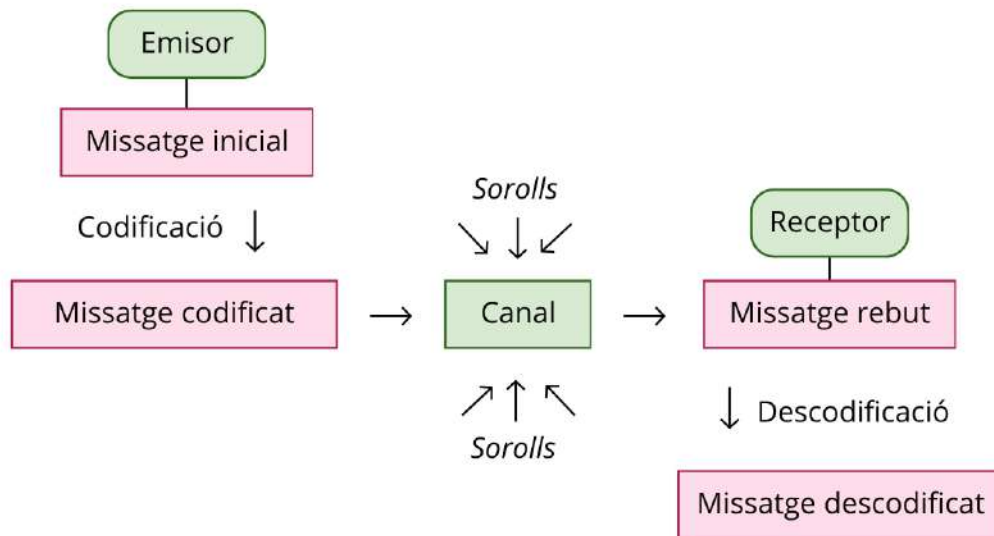


Figura 1. Sistema de codificació digital [8] i [17]. (Elaboració pròpia, 2025)

2.1.3. Dades codificades

Codificar, en el context d'aquest treball, significa aplicar una transformació matemàtica a la informació original per tal d'afegir-hi dades de control. Aquest procés rep el nom de codificació de la informació, i constitueix la base de la teoria de codis, una branca de les matemàtiques discretes¹. Aquesta teoria estudia com representar i manipular dades per tal de garantir-ne la fiabilitat, especialment en entorns on poden produir-se errors.

Els dos objectius fonamentals de la codificació són:

- Detectar si hi ha hagut un error (codificació amb detecció)
- Detectar i corregir errors de manera automàtica (codificació amb correcció)

¹ Matemàtiques discretes: branca de les matemàtiques que estudia estructures que no són contínues, sinó formades per elements aïllats i separats entre si, com ara els nombres enters, els grafs, els codis, la combinatòria o la criptografia. Per contra, les matemàtiques contínues s'ocupen de conceptes que varien sense salts, com les funcions reals, les corbes, les superfícies o el càlcul diferencial i integral.

2.1.4. Detecció i correcció d'errors

Els sistemes de codificació es poden classificar en dues grans categories segons la seva capacitat: codis detectors d'errors i codis correctors d'errors [8].

D'una banda, els codis detectors d'errors permeten identificar si s'ha produït una alteració en la transmissió d'un missatge. Ara bé, no poden determinar quina és la dada incorrecta ni corregir-la automàticament. Un exemple és el número ISBN d'un llibre. Si algun dels dígit s'introdueix de forma errònia, el sistema detecta que hi ha un error i rebutja el codi, però no pot indicar exactament quin dígit és incorrecte.

D'altra banda, els codis correctors d'errors no només poden detectar la presència d'errors, sinó també identificar-los i corregir-los automàticament. Per aconseguir-ho, el sistema afegeix informació addicional al missatge original. Això li permet reconstruir correctament la informació fins i tot si alguna part ha estat alterada durant la transmissió. Un exemple d'aquest tipus de codificació es troba en els CD de música o els DVD: quan hi ha una petita rascada o defecte que impedeix la lectura d'una part de la informació, el sistema és capaç de recuperar-la correctament gràcies als codis correctors d'errors.

2.2. Tipus de codis de detecció i correcció d'errors

A l'apartat anterior s'ha vist que, per tal d'assegurar que la informació arribi correctament d'un lloc a un altre, cal codificar-la. Aquesta codificació pot tenir com a funció la detecció d'errors o, fins i tot, la seva correcció automàtica. Però, com s'aconsegueix això?

Per donar resposta a aquesta necessitat, s'han desenvolupat diferents tipus de codis basats en principis matemàtics. Cada codi ha estat dissenyat per a una funció concreta i per a un context específic [17].

En aquest apartat es presenten aquestes dues grans categories:

- Els **codis detectors d'errors**, que només alerten que alguna cosa no és correcta.
- Els **codis correctors d'errors**, que poden deduir quin ha estat l'error i reconstruir la informació original.

2.2.1. Codis detectors d'errors

Els codis detectors d'errors són els més simples. El seu objectiu no és arreglar el problema, sinó donar un avís que hi ha hagut alguna alteració durant la transmissió. Aquest avís és suficient en molts casos. Per exemple, si es compra un producte en línia i s'introdueix malament el número d'una targeta, el sistema diu que "el número no és vàlid", i demana que es torni a introduir. No intenta endevinar quin número es volia escriure, simplement fa una alerta.

Per aconseguir garantir que els errors en la transmissió de dades siguin detectats, aquests codis afegeixen un dígit de control (o un conjunt petit de dígit) al final del missatge. Aquest dígit s'obté a partir d'una fórmula matemàtica aplicada al conjunt de dades originals, i serveix per comprovar si el missatge ha estat alterat. Aquest sistema s'anomena dígit de control o *check digit*².

A continuació es presenten alguns dels codis detectors més coneguts i utilitzats.

2.2.1.1. Bit de paritat

El bit de paritat és un dels mètodes més antics i senzills per detectar errors, utilitzat principalment en informàtica i en transmissions digitals. El funcionament bàsic consisteix a comptar quants bits amb valor 1 té un missatge i afegir un bit addicional (0 o 1) perquè el total tingui una paritat determinada.

Hi ha dos tipus principals de paritat: paritat parella i paritat senar. En la paritat parella, si el missatge té un nombre parell d'uns, s'afegeix un 0; si la quantitat d'uns és senar, el bit de paritat és 1. Per contra, en la paritat senar s'afegeix un 0 quan la quantitat d'uns és senar i un 1 quan és parella [6].

Es pren com a exemple el missatge original: 1011001 i es compten els uns (1) que conté; en aquest cas, n'hi ha 4.

Taula 1. Taula del càlcul del bit de paritat en funció del tipus de paritat. (Elaboració pròpia, 2025)

Missatge original	Quantitat d'uns	Tipus de paritat	Bit de paritat afegit	Missatge codificat
1011001	4 (parell)	Parella	0	1011001 0
1011001	4 (parell)	Senar	1	1011001 1

² Dígit de control o *check digit*: xifra afegida a un codi per comprovar si és correcte.

En el cas de la paritat parella, com que la quantitat d'uns és parella (4), el bit de paritat serà 0, i el missatge codificat queda 10110010. Si durant la transmissió es produeix un error i un bit canvia, per exemple el missatge rebut és 10100010, ara la quantitat d'uns és 3 (senar), i el sistema detecta que la paritat no correspon, indicant que hi ha hagut un error.

En el cas de la paritat senar, com que la quantitat d'uns és parella (4), el bit de paritat serà 1, i el missatge codificat queda 10110011. Si durant la transmissió un bit canvia i el missatge rebut és 10110010, ara la quantitat d'uns és 3 (senar), que no coincideix amb la paritat esperada, i es detecta l'error.

Aquest sistema permet detectar errors quan es produeix una modificació d'un sol bit. No obstant això, si canvien dos bits (per exemple, un 1 passa a 0 i un 0 passa a 1), la paritat podria mantenir-se, i l'error passaria desapercebut. Per això, tot i la seva senzillesa i utilitat, el bit de paritat té un límit en la detecció d'errors.

2.2.1.2. Codis ISBN, IBAN i EAN³

Els codis ISBN (per a llibres), IBAN (per a comptes bancaris) i EAN (per a productes comercials) són exemples de codis reals i aplicats a la vida quotidiana, que fan servir mètodes matemàtics específics per calcular un dígit de control. Aquest dígit s'utilitza per comprovar si el codi ha estat escrit correctament, i permet detectar errors en la seva introducció o transmissió [17].

2.2.1.3. CRC (Cyclic Redundancy Check)

El CRC (*Cyclic Redundancy Check*) és un mètode de detecció d'errors molt utilitzat en l'àmbit de la informàtica i les telecomunicacions, especialment en transmissions de dades per internet, arxius comprimits o comunicacions de xarxa. El seu funcionament es basa en operacions matemàtiques realitzades sobre els bits del missatge, i permet detectar si aquest ha estat alterat durant la transmissió [13].

Tot i la seva eficàcia, aquest sistema basat en operacions d'aritmètica binària i polinomis generadors, no es desenvolupa amb detall en aquest treball, ja que

³ En l'apartat 2.4, dedicat a exemples concrets de la vida quotidiana, es veurà com es calculen exactament els dígits de control d'aquests codis i quines fórmules matemàtiques s'utilitzen.

l'objectiu és aprofundir en codis més propers als que es poden trobar en aplicacions de la vida quotidiana. Tanmateix, es considera rellevant mencionar-lo per oferir una visió general dels sistemes existents de detecció i correcció d'errors.

2.2.2. Codis correctors d'errors

Els codis correctors funcionen afegint, al missatge original, una quantitat considerable d'informació extra organitzada de forma molt específica. Aquesta informació, com s'ha mencionat anteriorment, permet detectar si hi ha hagut un error i, en cas afirmatiu, identificar exactament on s'ha produït. Finalment —i aquí rau la seva diferència principal respecte als codis detectors—, permet substituir el valor erroni pel correcte.

Aquest procés és possible gràcies a les matemàtiques, i en particular a l'ús d'eines com les matrius, els vectors i l'aritmètica modular. Aquesta última es tractarà en més detall en l'apartat 2.3. del treball.

2.2.2.1 Exemples de codis correctors d'errors

Tot i que aquest treball no se centra específicament en els codis correctors d'errors, a continuació es presenten alguns exemples dels codis més coneguts i utilitzats.

Codi de repetició

El codi de repetició és un dels codis correctors d'errors més senzills i fàcils d'entendre. La idea bàsica consisteix a repetir cada bit del missatge original diverses vegades (en aquest cas, tres vegades) per afegir redundància. Aquesta redundància permet detectar i corregir errors que puguin aparèixer durant la transmissió.

El funcionament és simple: si un bit original és un 1, s'envia 111; si és un 0, s'envia 000. Quan el receptor rep el missatge, comprova cada grup de tres bits i decideix quin és el valor correcte mitjançant la regla de majoria⁴. Per exemple, si es rep 110, el receptor interpreta que el bit original era 1 perquè la majoria (dos bits) són 1. Així, el codi de repetició ($n = 3$) pot corregir com a màxim un error dins del grup de tres

⁴ La *regla de majoria* o *vot majoritari* és un mètode simple per decidir quin valor representa millor un conjunt de dades repetides. Consisteix a triar el valor que apareix més vegades dins d'un grup determinat, suposant que la probabilitat d'error és baixa. Aquesta regla s'utilitza en codis de repetició per corregir errors simples. (https://en.wikipedia.org/wiki/Majority_rule)

bits. Pel que fa a la detecció d'errors, cal una distinció important: si el receptor només aplica la regla de la majoria, un grup amb dos bits canviats (per exemple, d'un 111 es rep 100) s'interpretarà com el bit contrari i no s'indicarà cap error. En canvi, si el receptor comprova explícitament si el grup rebut coincideix exactament amb 000 o amb 111 i assenyala "error" quan no coincideix, aleshores qualsevol patró diferent (com 100 o 011) evidencia que hi ha hagut un error i, per tant, es poden detectar fins a dos bits erronis. Finalment, si els tres bits es modifiquen (per exemple, d'un 000 es passa a 111), el receptor no detectaria cap error perquè el patró rebut seria un codi vàlid però incorrecte.

Per aquest mateix motiu, cal tenir en compte que el codi té una limitació important que provoca que el codi de repetició sigui útil només en canals amb errors poc freqüents i aïllats.

A continuació es mostra una taula amb un exemple pràctic del codi de repetició que ajuda a entendre com funciona la detecció i correcció d'errors, així com quan la correcció pot fallar:

Taula 2. Taula de correcció d'errors amb codi de repetició i vot majoritari. (Elaboració pròpia, 2025)

Missatge original	Missatge codificat (enviat)	Missatge rebut (amb error)	Missatge corregit (amb vot majoritari)	La correcció és correcta?
1 0 1	111 000 111	111 010 111	1 0 1	Sí
1 0 1	111 000 111	010 010 111	0 0 1	No

En la primera fila, s'observa que només hi ha un error dins d'un grup (000 canviat a 010), i la correcció funciona bé. En canvi, a la segona fila, hi ha dos errors dins del primer grup (111 canviat a 010), i això provoca que el sistema interpreti erròniament el bit original, fallant en la correcció.

Codi de Hamming

El codi de Hamming és un dels codis correctors més senzills i famosos. Va ser desenvolupat per Richard W. Hamming⁵ l'any 1950, i encara avui s'utilitza en

⁵ Richard Hamming (1915-1998) va ser un matemàtic nord-americà que va treballar en el Projecte Manhattan programant alguns dels primers ordinadors. Allà va desenvolupar un programa per calcular si l'explosió de la bomba atòmica podia incendiar l'atmosfera, una de les grans pors científiques del moment [7].

diversos sistemes bàsics. La seva idea principal consisteix a afegir bits de control o de paritat en posicions estratègiques dins del missatge. Aquests bits addicionals permeten fer un seguiment de les diferents posicions, de manera que, si un d'ells no concorda amb el valor esperat, el sistema pot identificar exactament on s'ha produït l'error [13].

En termes matemàtics, els codis de Hamming es poden descriure formalment amb matrius i amb operacions d'àlgebra lineal sobre el cos binari (és a dir, amb sumes i multiplicacions que només tenen en compte els valors 0 i 1). No obstant això, en aquest treball no s'entra en aquest nivell de detall ni es presenten exemples, ja que requereixen coneixements més avançats i no formen part dels objectius de recerca.

En el cas més conegut, el codi de Hamming (7,4), se'n codifiquen 4 bits d'informació i s'hi afegeixen 3 bits de redundància, formant blocs de 7 bits. Amb aquesta construcció, el sistema pot corregir com a màxim un error i detectar fins a dos errors dins de cada bloc de 7 bits. Quan es rep un missatge alterat, el receptor calcula un valor anomenat síndrome, que indica la posició del bit erroni i permet corregir-lo automàticament. En canvi, si hi ha dos bits alterats en el mateix bloc, el codi no pot corregir-los, però sí detectar que hi ha hagut una alteració.

Codi BCH

El codi BCH (Bose–Chaudhuri–Hocquenghem) és una generalització del codi de Hamming mencionat anteriorment. Tot i que funciona amb el mateix principi bàsic, és molt més potent, ja que pot detectar i corregir múltiples errors, fins i tot si estan encadenats i no són aïllats. Aquest codi fa ús de càlculs més complexos sobre cossos finits⁶, cosa que li dona una major eficàcia en aplicacions amb un risc més elevat d'errors.

Codi Reed-Solomon

El codi Reed–Solomon és especialment útil en casos en què els errors no es produeixen de forma puntual, sinó que afecten blocs sencers de dades. Es tracta d'un codi molt potent que pot corregir grups d'errors que apareixen junts. El seu

⁶Cossos finits: són estructures matemàtiques amb un nombre limitat d'elements on es poden fer les quatre operacions bàsiques —suma, resta, multiplicació i divisió (excepte per zero)— de manera que el resultat sempre queda dins del conjunt.

funcionament es basa en l'ús de polinomis amb coeficients sobre cossos finits, com podria ser el cos finit $\mathbb{Z}/p$ (el conjunt dels nombres enters mòdul p) essent p un nombre primer; i en tècniques d'àlgebra avançada. És utilitzat en tecnologies com CD, DVD, transmissions digitals o codis QR [11].

2.3. Fonaments matemàtics

Els codis que s'han mencionat fins ara no són només una qüestió tecnològica o informàtica, sinó que tenen una base matemàtica molt sòlida. Per poder crear-los, aplicar-los i entendre com funcionen, cal dominar certs conceptes propis de les matemàtiques discretes, una branca que estudia objectes que es poden comptar: nombres enters, caràcters, bits...

Aquest apartat se centra en una eina fonamental per entendre molts dels codis utilitzats actualment: l'aritmètica modular, també coneguda com a aritmètica del rellotge. Aquesta tècnica, que té els seus orígens en el treball del matemàtic grec Euclides (s. III a.C.), és un recurs clau en la codificació i validació de dades com l'algoritme de Luhn, l'IBAN o el codi ISBN [15].

Cal tenir en compte que, tot i que aquest treball no hi aprofundirà, altres eines matemàtiques com l'àlgebra lineal, especialment l'ús de matrius, també tenen un paper important en la representació i construcció de codis correctors més avançats, com per exemple el codi de Hamming, el codi BCH, el codi Reed-Solomon o fins i tot els codis de repetició.

2.3.1. Aritmètica modular

L'aritmètica modular és un sistema de càlcul on no interessa el resultat total d'una operació, sinó el residu que queda en dividir un nombre per un altre. Aquest residu és el que ens diu "on" ens trobem dins d'un cicle, com si es fessin comptes amb un rellotge [2].

En general, s'escriu:

$$a \equiv b \pmod{m}$$

Amb això s'està dient que si es redueix a mòdul m , s'obté b .

Per exemple, si es divideix 17 entre 5: $17 \div 5 = 3$ (amb residu 2) $\Rightarrow 17 \equiv 2 \pmod{5}$

Aquest sistema és el que s'aplica, per exemple, en els rellotges analògics de 12 hores. Si són les 10 i passen 5 hores, serien les 15, però:

$$15 \equiv 3 \pmod{12} \rightarrow \text{Així doncs, seran les 3.}$$

També es fa servir aquest sistema en altres contextos, com el càlcul d'angles. Un angle de 370° és equivalent a un de 10° , ja que:

$$370 \equiv 10 \pmod{360}$$

Aquest tipus de càlculs permeten reduir qualsevol nombre a un valor entre 0 i $n-1$, on n és el mòdul.

A part d'aquestes aplicacions quotidianes, el mateix mecanisme és clau en àmbits molt més sofisticats. De fet, la capacitat de reduir nombres grans a valors dins d'un interval fix fa que l'aritmètica modular sigui molt útil en codificació i criptografia. És precisament per aquest motiu que aquest treball inclou un apartat dedicat a l'aritmètica modular, ja que per entendre correctament els codis que s'analitzaran més endavant cal dominar aquesta eina. Ara bé, no tots els codis fan servir els mateixos mòduls. Segons la seva funció i el tipus de dades que tracten, alguns sistemes utilitzen mòduls específics. A continuació, es mostren alguns dels mòduls més habituals en codificació:

Taula 3. Taula amb els mòduls habituals en codificació. (Elaboració pròpia, 2025)

Mòdul	Ús habitual
mod 2	Aquest mòdul només té dos valors possibles: 0 o 1. És la base del sistema binari, utilitzat en la codificació digital d'informació en ordinadors i dispositius electrònics.
mod 10	S'utilitza per calcular dígit de control en codis on es vol detectar errors simples, com canviar una xifra o invertir-ne dues. Exemples: en targetes de crèdit, codis EAN, ISBN-13...
mod 11	Aquest mòdul és útil perquè és més eficaç que el mod 10 per detectar errors de transposició (canviar dues xifres de lloc). S'aplica en l'antic format d'ISBN-10, on si el residu és 10 es representa amb una X.
mod 97	Aquest mòdul és especialment útil per codis llargs i sensibles com els bancaris (exemple: IBAN), ja que permet una validació robusta.

2.4. Exemples de codis a la vida quotidiana

En aquest punt es mostra com els codis estudiats prèviament s'apliquen a objectes i sistemes que són utilitzats diàriament en la vida quotidiana: llibres, documents d'identitat, targetes bancàries, productes del supermercat... En tots aquests casos, la combinació de dígit que forma cada codi no és aleatòria, sinó que segueix un sistema matemàtic concret dissenyat per verificar-ne la validesa i protegir-los contra possibles errors de transmissió o introducció [3].

Un dels elements clau en aquests codis és el dígit de control, un nombre addicional que es calcula a partir dels altres dígit segons una fórmula específica, generalment basada en l'aritmètica modular. Aquest dígit, com s'ha explicat anteriorment, serveix per detectar errors comuns en la transmissió o l'escriptura del codi, com canvis, omissions o transposicions de xifres.

Taula 4. Taula amb alguns tipus d'errors que es poden donar en un codi. (Elaboració pròpia, 2025)

Tipus d'error	Descripció	Missatge original	Missatge amb l'error
Substitució	Un dígit es modifica per un altre	12345	17345
Transposició	Dos dígit s'intercanvien de lloc (adjacents o no).	12345	13245
Omissió ⁷	Falta un dígit (no s'ha transcrit).	12345	1245

En aquest apartat, s'explicarà detalladament l'estructura de diversos codis habituals en la nostra vida quotidiana, la fórmula que s'utilitza per calcular el seu dígit de control i, sobretot, com s'aplica pràcticament aquest càlcul per verificar la validesa dels codis.

A continuació, s'analitzaran codis com el DNI, l'ISBN dels llibres, els codis bancaris IBAN i CCC, els codis de productes EAN i, finalment, s'introduirà breument el sistema de codificació de les targetes bancàries, que aquest tindrà un apartat específic i més extens més endavant, ja que es tracta del tema principal d'aquest treball.

⁷ En el cas de l'omissió, pot ocórrer que es conegui l'espai on ha desaparegut el dígit o no. És a dir, el missatge amb l'error també podria ser 12_45.

2.4.1.DNI

A Espanya, el DNI (Document Nacional d'Identitat) inclou un número personal de vuit xifres seguit per una lletra. Aquesta lletra no és aleatòria, sinó que consisteix en el dígit de control i serveix per comprovar que el número és correcte. Si s'escriu una lletra diferent a la qual correspondria segons el càlcul establert, el sistema informàtic pot detectar que s'ha comès un error i no acceptar aquest document.

La lletra del DNI es calcula a partir del número mitjançant una operació matemàtica molt senzilla basada en l'aritmètica modular. Per tal d'obtenir la lletra corresponent, primer de tot cal aplicar mòdul 23, és a dir, dividir el número de vuit dígit entre 23 i obtenir el residu d'aquesta divisió [16]. Aquest residu és un número entre 0 i 22 al qual se li assigna una lletra segons una taula de correspondència fixa:

Taula 5. Taula de correspondència del residu amb la lletra. (Ministerio del Interior, 2025)

RESIDU	0	1	2	3	4	5	6	7	8	9	10	11
LLETRA	T	R	W	A	G	M	Y	F	P	D	X	B

RESIDU	12	13	14	15	16	17	18	19	20	21	22
LLETRA	N	J	Z	S	Q	V	H	L	C	K	E

Per exemple, si el número del DNI és 12345678, aquest es divideix entre 23 i se'n mira el residu: $12345678 \div 23 = 536768$ amb residu 14.

Segons la taula, el residu 14 correspon a la lletra Z, per tant, el DNI complet serà: 12345678Z.

2.4.2. ISBN

L'ISBN (*International Standard Book Number*) és un codi numèric que s'utilitza per identificar de manera única qualsevol llibre publicat a escala mundial. Aquest número facilita el registre, la distribució i la venda dels llibres en llibreries i biblioteques, així com la seva classificació en catàlegs editorials. Cada edició d'un llibre, sigui en format imprès, electrònic o audiollibre, rep un codi ISBN diferent.

Tot i que actualment l'ISBN-10 ha estat substituït pel format ISBN-13, un codi de tretze dígit compatible amb els codis de barres EAN-13⁸ (European Article Number)

⁸ El funcionament del codi EAN-13 s'explica pròximament en l'apartat 2.3.4. EAN (codi de barres)

utilitzats en el comerç internacional, resulta interessant comprendre el funcionament de l'ISBN original de deu xifres, ja que constitueix un exemple clar d'algoritme de dígit de control i permet entendre els conceptes bàsics de detecció d'errors.

L'ISBN original segueix una estructura estandarditzada de deu xifres:

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$$

Els primers 9 dígits contenen informació del llibre, com l'editorial, títol o edició, i l'últim dígit, a_{10} , és el dígit de control. Aquest es calcula mitjançant una suma ponderada de les primeres 9 xifres, on cada xifra es multiplica pel seu número de posició (d'1 a 9). La suma resultant s'aplica a mòdul 11: es calcula el residu de dividir aquesta suma entre 11, i aquest residu determina el dígit de control. I, en el cas que el residu sigui 10, el dígit de control s'escriu com una 'X', seguint la convenció de nombres romans [11].



Figura 2. Les parts d'un ISBN de 10 dígits (a la part superior) i el corresponent EAN-13 (a la part inferior) i codi de barres. (Wikipedia, 2025)

Nota: L'ISBN-10 té quatre parts: grup (país o regió), registrant (editorial), publicació (edició o format) i dígit de control. L'ISBN-13 té cinc parts: prefix GS1 (978 o 979), grup, registrant, publicació i dígit de control; la part "EAN" indica el país de Bookland.

La fórmula per al càlcul és:

$$(a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6 + 7a_7 + 8a_8 + 9a_9) \equiv a_{10} \pmod{11}$$

Per exemplificar el càlcul, es considera l'ISBN-10: **8482986929**, corresponent al llibre *Matemáticas y códigos secretos* [14] on es calcula la suma ponderada de les nou primeres xifres multiplicades per la seva respectiva posició:

$$8 + 2 \times 4 + 3 \times 8 + 4 \times 2 + 5 \times 9 + 6 \times 8 + 7 \times 6 + 8 \times 9 + 9 \times 2 = 273$$

Tot seguit, s'aplica mòdul 11, és a dir es fa la divisió entre 11 i simplement cal fixar-se amb el residu, aquest és el resultat de mòdul 11 que hauria de correspondre amb el dígit de control.

$$\begin{array}{r} 273 \quad | \quad 11 \\ 9) \quad \quad 24 \end{array}$$

El residu de la divisió correspon al dígit de control a_{10} , que en aquest cas és 9, per tant, l'ISBN-10 és correcte.

Aquest sistema de validació, basat en l'àritmètica modular, resulta molt útil per detectar errors de transcripció o introducció de dades, garantint així la correcta identificació dels llibres dins del sistema global d'ISBN.

2.4.3. IBAN i CCC

El codi IBAN (*International Bank Account Number*) és un identificador internacional que s'utilitza per numerar els comptes bancaris de manera estandarditzada a tot Europa i en altres països que han adoptat aquest sistema. Va ser creat amb l'objectiu de facilitar les transferències internacionals i garantir la seguretat i la precisió en les operacions bancàries. Per aconseguir-ho, incorpora un mecanisme, basat en aritmètica modular, de verificació que permet detectar errors en la transcripció o introducció del codi.

A l'Estat espanyol, l'IBAN està format per un total de 24 caràcters. Els dos primers són lletres que indiquen el país (ES, en aquest cas), els dos següents són dígits de control que es calculen a partir de la resta del codi, i els vint restants corresponen al Codi Compte Client, conegut com a CCC. Aquest codi nacional conté informació sobre l'entitat bancària, l'oficina, dos dígits de control interns i el número de compte.

El CCC també incorpora un sistema propi de verificació basat en pesos i sumes ponderades, però el càlcul és més complex i s'allunya de l'objectiu d'aquest treball. Per aquest motiu no s'explicarà amb detall, ja que el focus s'ha posat en el càlcul dels dígits de control de l'IBAN, molt més clar des del punt de vista de l'àritmètica modular. Aquelles persones que vulguin aprofundir en el funcionament del CCC poden consultar l'article *Codis detectors i correctors d'errors i algunes de les seves aplicacions a la societat de la informació* [17].

El càlcul del dígit de control de l'IBAN comença partint del CCC. A continuació, s'hi afegeixen provisionalment les dues lletres del país (ES) seguides dels dos dígit 00 (aquests posteriorment seran substituïts pels dígit de control calculats). En aquest moment, es fa una transformació de les lletres en nombres. Aquesta conversió segueix l'estàndard internacional ISO 13616-1:2020 [11], que defineix l'estructura de l'IBAN i el mètode de validació. Segons aquest estàndard, les lletres es transformen en números assignant a la A el valor 10, a la B el valor 11, i així successivament fins a la Z, que pren el valor 35. Per tant, en el cas d'Espanya, la E es transforma en 14 i la S en 28.

Taula 6. Taula de correspondència de cada lletra amb el nombre. [11] (Elaboració pròpia, 2025)

LLETRA	A	B	C	D	E	F	G	H	I	J	K	L	M
NOMBRE	10	11	12	13	14	15	16	17	18	19	20	21	22

LLETRA	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
NOMBRE	23	24	25	26	27	28	29	30	31	32	33	34	35

Un cop s'ha format aquest número llarg només amb xifres, es calcula el residu de la divisió entre 97, és a dir, es fa el càlcul del mòdul 97. Finalment, aquest residu es resta a 98. El resultat és el dígit de control de l'IBAN. Si el nombre obtingut és d'una sola xifra, se li afegeix un zero davant per formar un nombre de dues xifres, tal com exigeix el format. Aquestes xifres s'afegeixen a l'inici del número just després de les sigles del país, i en conseqüència, s'esborren els 00 del final.

És important remarcar que, per verificar un IBAN existent, es pot fer el mateix procés a la inversa: col·locar les lletres al final i afegir-hi seguidament 00. Després eliminar els dos primers dígit corresponents als de control i aplicar mòdul 97 al número resultant i finalment, comprovar que el resultat de restar el residu a 98 correspon amb els dígit de control.

A continuació, es mostra un procediment desenvolupat per exemplificar tot aquest procés.

S'agafa un compte amb el CCC següent: **2100 0813 61 0123456789**. Aquest codi conté la identificació de l'entitat (2100), l'oficina (0813), els dígit de control del CCC (61) i el número de compte (0123456789). Per obtenir l'IBAN corresponent, primer

s'escriu aquest CCC i, a continuació, s'hi afegeixen les dues lletres ES i dos zeros: ES00. El resultat és la cadena provisional: 21000813610123456789ES00.

Tot seguit, se substitueixen les lletres ES pels nombres corresponents: E és 14 i S és 28, de manera que la cadena es converteix en el número següent: 21000813610123456789142800

Ara cal calcular el mòdul 97 d'aquest número, és a dir, el residu de dividir aquest número tan llarg entre 97. El resultat d'aquesta operació és 18. Seguint el procediment, es resta aquest residu a 98, i s'obté el valor 80. Aquest és el dígit de control de l'IBAN. Per tant, l'IBAN complet que correspon a aquest compte és: ES80 2100 0813 61 0123456789.

Aquest mètode és especialment útil perquè permet detectar, de manera ràpida i automàtica, errors habituals com invertir dues xifres, ometre'n alguna o escriure'n una de forma incorrecta. Així, l'aritmètica modular no només és una eina teòrica dins les matemàtiques, sinó que té aplicacions pràctiques reals i de gran utilitat en àmbits com el sistema bancari.

2.4.4. EAN (codi de barres)

El codi EAN és un sistema internacional de codificació que permet identificar de manera única productes comercials mitjançant un codi numèric imprès habitualment en forma de codi de barres a l'embalatge o etiqueta del producte. El format més utilitzat és l'EAN-13, que consta de 13 dígits i està àmpliament implantat des de la seva creació l'any 1976 [15]. Aquest codi incorpora informació sobre el país d'origen, l'empresa fabricant, el producte i un dígit de control final, que valida la correcta lectura del codi i detecta possibles errors.

Els 13 dígits es distribueixen en tres parts: els dos o tres primers dígits indiquen el codi país d'origen del producte, els nou o deu següents identifiquen l'empresa i el producte. Finalment, l'últim dígit és el dígit de control, calculat a partir dels dotze dígits anteriors per garantir la integritat de la informació codificada [15].

Matemàticament, la fórmula per al càlcul del dígit de control a_{13} és:

$$(a_1 + 3a_2 + a_3 + 3a_4 + a_5 + 3a_6 + a_7 + 3a_8 + a_9 + 3a_{10} + a_{11} + 3a_{12} + a_{13}) \equiv 0 \pmod{10}$$

Aquesta condició implica que la suma ponderada de les 13 xifres, on les posicions parelles es multipliquen per 3 i les senars per 1, ha de ser múltiple de 10. Si no ho és, el dígit de control a_{13} és el nombre que cal sumar per arribar al múltiple de 10 immediatament superior.

Per exemple, es considera el codi EAN complet: **5291047134562** (on el dígit de control és el 2 que es vol comprovar).

Primer es calcula la suma ponderada dels dotze primers dígit multiplicant alternativament per 1 i per 3:

$$5 + 3 \times 2 + 9 + 3 \times 1 + 0 + 3 \times 4 + 7 + 3 \times 1 + 3 + 3 \times 4 + 5 + 3 \times 6 = 83$$

A continuació, es busca el dígit a_{13} que faci que la suma total sigui múltiple de 10, és a dir, que compleixi:

$$83 + a_{13} \equiv 0 \pmod{10}$$

A l'afegir a_{13} a 83 ha de donar residu 0 a mòdul 10, cosa que significa que la suma ha de ser múltiple de 10. Per tant, sabent que a_{13} ha de ser un nombre d'un sol dígit (del 0 al 9), la suma $83 + a_{13}$, ha de ser igual al nombre múltiple de 10 següent a 83 que és el 90. Així doncs, l'equació queda així:

$$83 + a_{13} = 90 ; \text{ i es resol } a_{13} = 90 - 83 = 7$$

Així doncs, el dígit de control hauria de ser el 7. Però, l'últim dígit del codi original és 2, que no coincideix amb el càlcul fet, la qual cosa indicaria un error en la lectura o en la còpia del codi.

Aquest sistema basat en l'aritmètica modular mòdul 10 permet detectar automàticament errors causats per defectes d'impressió, danys o errors humans en la introducció manual, de manera similar als sistemes utilitzats en targetes bancàries o ISBN-13.

2.3.5. CODABAR (targetes bancàries)

El número imprès a la part frontal d'una targeta de crèdit o dèbit no és aleatori, sinó que respon a una estructura estandarditzada definida per la norma internacional ISO/IEC 7812⁹. Aquest número, anomenat PAN (*Primary Account Number*), permet identificar l'entitat emissora i el titular del compte, i incorpora un dígit de control que es calcula mitjançant l'algoritme de Luhn, especialment dissenyat per detectar errors de transcripció habituals [15]. L'estructura interna d'aquest número, així com el funcionament detallat de l'algoritme de validació, es desenvoluparan en un apartat posterior d'aquest treball. A més, a la part pràctica es detallarà per què l'algoritme funciona i els casos en què detecta o no errors.

2.5. Les matemàtiques darrere de les targetes bancàries

Avui dia, les targetes bancàries s'han convertit en una eina imprescindible per a milers de milions de persones. Són utilitzades per fer compres, treure diners, pagar en línia o identificar-se davant d'una entitat financera. Però, malgrat que l'acció sigui senzilla i ràpida, darrere del funcionament d'una targeta bancària hi ha un sistema complex de verificació, on les matemàtiques tenen un paper clau per garantir la seguretat.

Aquest apartat analitza les claus matemàtiques que fan possible aquest sistema. S'hi explica què conté exactament una targeta bancària, com es construeix el número que l'identifica i com funciona el mètode que permet verificar si aquest número és vàlid.

2.5.1. Què hi ha dins una targeta de crèdit?

Una targeta bancària pot funcionar com a instrument de dèbit o de crèdit. En el primer cas, permet operar amb els diners disponibles al compte associat; en el segon, ofereix un crèdit temporal que el titular haurà de retornar. Malgrat aquesta diferència, totes les targetes comparteixen una estructura funcional similar que

⁹ La norma ISO/IEC 7812 és l'estàndard internacional que regula l'estructura i el format dels números d'identificació de targetes bancàries. Aquesta norma defineix, entre altres coses, el codi d'identificador del banc emissor (IIN) i l'estructura del número complet (PAN), inclòs el dígit de control final. Està publicada per l'Organització Internacional de Normalització (ISO) i la Comissió Electrotècnica Internacional (IEC).

combina elements visibles amb altres de no visibles. En aquest apartat s'analitzen des dels components principals d'una targeta bancària i les dades que s'hi poden trobar [1] fins als sistemes de codificació que hi ha al darrere.

En les figures 3 i 4 es pot veure els elements i la informació que conté l'anvers i revers d'una targeta de crèdit:



Figura 3. Exemple de l'anvers d'una targeta de crèdit.

Nota: 1. Logotip del banc, 2. Xip EMV¹⁰, 3. Holograma, 4. Número de targeta de crèdit, 5. Companyia de la targeta de crèdit, 6. Data de caducitat, 7. Nom del propietari (Viquipèdia, 2025)

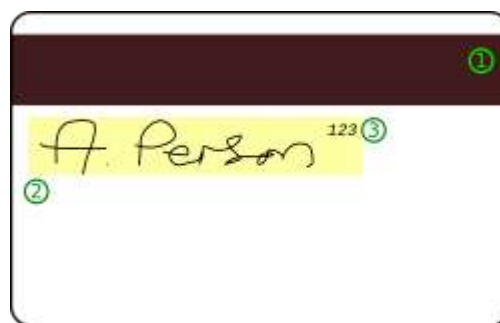


Figura 4. Exemple del revers d'una targeta de crèdit.

Nota: 1. Banda magnètica, 2. Signatura, 3. Número de seguretat (CSC¹¹) (Viquipèdia, 2025)

- El nom del titular, generalment imprès a la part frontal.
- La data de caducitat, que indica el mes i any fins al qual la targeta és vàlida.
- El logotip de l'entitat emissora i de la marca de la targeta (VISA, MasterCard..).
- A la part posterior, el codi CSC, CVV o CVC (de 3 o 4 dígits), utilitzat com a mesura de seguretat addicional en operacions en línia.
- Una banda magnètica i, en la majoria de targetes actuals, un xip EMV, que emmagatzema dades codificades de manera més segura i permet comunicació amb els terminals de pagament.

¹⁰ «EMV és la sigla “Europay, Mastercard i Visa (en anglès)”. El xip EMV és l'estàndard global utilitzat per als xips de targetes de crèdit a tot el món.»

<https://www.creditkarma.com/es/tarjetas-de-credito/i/como-funciona-el-chip-emv-de-tarjeta-de-credito>

¹¹ «El codi de seguretat d'una targeta es coneix com a CSC, però altres vegades pot figurar com CVV, CW2, CVC2 o número de CID. En definitiva, és una nova mesura de seguretat, una comprovació criptogràfica de la informació de la targeta, destinada a evitar qualsevol ús fraudulent d'aquesta.»

<https://es.quora.com/Cu%C3%A1l-es-el-CSC-de-una-tarjeta-de-d%C3%A9bito>

- Un número format per 16 díigits, conegut com a *Primary Account Number* (PAN). Aquest número és fonamental, ja que les targetes de crèdit i dèbit que emeten les entitats bancàries i grans superfícies compten amb un codi numèric que segueix un patró matemàtic molt estricte i que permet identificar tant l'entitat emissora com el titular de la targeta. Aquest codi s'organitza generalment en 16 díigits, agrupats en quatre grups de quatre díigits cadascun, que podem representar així:

ABCD EFGH IJKL MNOP

Cada conjunt de díigits conté una informació concreta [14]:

- **A:** El primer dígit indica el tipus de targeta o l'empresa gestora. Aquest número pot variar i donar informació sobre si es tracta d'una targeta emesa per una aerolínia (7), per una companyia petrolera (1), un banc...

Taula 7. Taula de correspondència del primer dígit en les targetes emeses per una companyia bancària. (Urgellés, J. G., 2018)

Dígit E	Entitat
3	American Express
4, 0, 2	Visa
5, 0	Master Card
6	Discover

- **ABCDEFGH:** Juntament amb el primer dígit, els sis següents formen l'anomenat INN (*Issuer Identification Number*), que serveix per identificar la targeta i l'entitat que l'ha emesa. Aquest codi combina informació sobre el tipus de banc emissor i la zona geogràfica on s'ha generat la targeta. Per raons de seguretat, els detalls exactes dels codis dels emissors i dels països no són públics.
- **HIJKLMNO:** Els següents vuit díigits identifiquen el titular de la targeta de manera única. Aquest bloc conté un número de sèrie que pot incloure informació sobre el nivell de la targeta (clàssica, or, platí), el límit de crèdit o altres característiques internes gestionades per l'entitat bancària. Aquest identificador garanteix que cada client tingui un número exclusiu dins de l'entitat emissora.

- **P:** Finalment, l'últim dígit és el dígit de control, que no s'assigna de manera arbitrària sinó que es calcula amb un algoritme matemàtic anomenat algorisme de Luhn. Aquest dígit serveix per validar que el número complet de la targeta és correcte, evitant errors de transcripció i intentant detectar possibles fraus.

En aquest treball s'estudia principalment el número de la targeta, ja que és una dada que pot ser verificada matemàticament mitjançant un mètode específic que comprova si la seva estructura és correcta.

2.5.2. L'algoritme de Luhn: com es valida una targeta

L'algoritme de Luhn és un procediment matemàtic ideat el 1954 per Hans Peter Luhn¹² amb l'objectiu de verificar si un número, com el d'una targeta de crèdit, és estructuralment vàlid. Aquest sistema no garanteix que una targeta sigui real o estigui activa, però permet detectar errors freqüents de transcripció, com ara una xifra mal escrita o dues xifres intercanviades. Per aquest motiu, s'utilitza àmpliament en la verificació automàtica de números de targeta i altres codis identificatius, gràcies a la seva rapidesa i eficàcia [15].

L'algoritme s'aplica mitjançant una sèrie de passos molt concrets. Per entendre millor com funciona, s'explica tot seguit de forma detallada fent servir un exemple: la targeta amb número **4539 3195 0343 6467**. Cal tenir present que l'últim dígit del número, que en aquest cas és el 7, és el dígit de control i serveix per verificar si tota la resta del número és correcta.

Així doncs, es posa per exemple la validació d'un número de targeta:

Pas 1: Primer de tot, es pren el número de la targeta i es llegeix d'esquerra a dreta. D'aquest, s'agafaran els dígit de les posicions senars, és a dir, un dígit sí i un no, que són els que es multiplicaran per dos en el següent pas. En aquest cas el número seria aquest: **4539 3195 0343 6467**; per tant, els dígit que cal agafar són els següents: 4, 3, 3, 9, 0, 4, 6, 6.

¹² Hans Peter Luhn (1896–1964) va ser un pioner en ciència de la informació i informàtica, conegut per l'algoritme de Luhn i el sistema d'indexació KWIC, que van influir en la recuperació d'informació i la mineria de textos. (<https://www.luhn.ai/post/who-was-hans-peter-luhn>)

Pas 2: Seguidament, s'agafen els dígit que s'han pres anteriorment, els de les posicions senars, i es multipliquen per dos. Com a resultat, queda: 8, 6, 6, 18, 0, 8, 12, 12.

Pas 3: Ara, s'agafen els resultats de les multiplicacions realitzades en el pas anterior: 8, 6, 6, 18, 0, 8, 12, 12. Si algun resultat té dues xifres (és més gran que 9), se sumen les xifres del resultat per reduir-lo a una sola xifra, o bé se li'n resta 9. Per tant, els dígit modificats són: 8, 6, 6, 9, 0, 8, 3, 3.

Pas 4: A continuació, cal agafar els dígit que no s'han tocat, és a dir, els dígit que es troben a les posicions parelles del número inicial (inclòs el dígit de control): **4539 3195 0343 6467** i s'han de sumar juntament amb els dígit modificats del pas anterior.

Pas 5: Finalment cal aplicar l'aritmètica modular amb mòdul 10. El número de targeta serà vàlid si la suma total feta anteriorment en mòdul 10 dona residu 0, és a dir, quan $x \equiv 0 \pmod{10}$, essent x la suma total. Dit d'una altra manera, per poder confirmar la validesa del número de targeta, cal que la suma total després de seguir tots els passos anteriors, sigui múltiple de 10.

Taula 8. Taula de l'exemple de validació d'un número de targeta amb l'aplicació de l'algoritme de Luhn. (Elaboració pròpia, 2025)

4	5	3	9	3	1	9	5	0	3	4	3	6	4	6	7	Σ
	5		9		1		5		3		3		4		7	37
8		6		6		18		0		8		12		12		
8		6		6		9		0		8		3		3		43
Suma total: $37+43 = 80$					$80 \equiv 0 \pmod{10}$				El dígit de control és correcte: 7							

En aquest cas $80 \equiv 0 \pmod{10}$, cosa que significa que 80 és divisible per 10, per això el residu dona 0. Per tant, el número 4539 3195 0343 6467 és vàlid segons l'algoritme de Luhn.

Ara bé, si el dígit de control fos diferent, per exemple un 8 en lloc d'un 7, la suma total seria 81, i no seria vàlida. Només quan la suma total és exactament un múltiple de 10 (com per exemple 70, 80 o 90), el número de la targeta es considera vàlid estructuralment.

Aquest procés matemàtic permet detectar errors immediatament després que el número sigui introduït. Aquesta comprovació es fa automàticament cada vegada que s'usa una targeta, ja sigui per fer una compra, una transferència o una operació bancària. És per això que l'algoritme de Luhn continua sent una eina fonamental en la seguretat i verificació del sistema financer modern.

3. MARC PRÀCTIC

La part pràctica del treball consisteix a crear i utilitzar un programa per validar números de targeta bancària amb l'algoritme de Luhn. L'objectiu és disposar d'una eina que permeti calcular els dígit de control i comprovar quins errors es poden detectar i quins no. Per aconseguir-ho, s'ha desenvolupat un programa que verifica números de targeta i calcula un dígit que falta en un número incomplet. Posteriorment, s'han analitzat els resultats obtinguts per entendre millor el sistema de detecció d'errors i el seu abast. A més, la informació recollida en el marc teòric ha servit per elaborar contingut divulgatiu a la Viquipèdia, per tal de difondre el coneixement sobre aquest algoritme.

3.1. Creació del programa

3.1.1. Cerca d'informació i programari

Abans d'iniciar la programació, s'ha dut a terme una fase d'investigació destinada a seleccionar les eines més adequades i a adquirir els coneixements necessaris per a la seva implementació.

En aquesta etapa inicial cal decidir quin programa utilitzar. Després de consultar amb el professor de programació de l'institut, Josep Planas, s'ha optat per utilitzar el llenguatge Python a causa de la seva senzillesa i llegibilitat, així com la llibreria Tkinter per poder treballar amb *widgets*, és a dir, els elements de la interfície gràfica que permeten interactuar amb el programa, com etiquetes de text, caixes per introduir dades o botons que executen accions.

Tot i que ja es tenen alguns coneixements previs de programació, mai no s'havia treballat amb Tkinter, així que ha estat necessari buscar informació extra per aprendre'n l'ús. Per això s'han consultat pàgines web especialitzades en Python i Tkinter [19], com GeeksforGeeks [9] i també alguns tutorials en vídeo del canal Mundo Python¹³ que mostren exemples pràctics. Aquesta recerca ha permès familiaritzar-se amb els conceptes bàsics i les tècniques necessàries abans de començar a programar.

¹³ Els tutorials que s'han visualitzat són: <https://youtu.be/jqRHhWjKDD8?si=oDnDBc5piL3vbWzh> i https://youtu.be/nh0Jeclyvt8?si=qs2rLbvPV-3_BH7G

3.1.2. Estructura del producte

Amb la base documental i pràctica ja establerta, s'ha iniciat la fase de disseny estructural del producte, després de tenir clars els requisits i les tècniques necessàries per a l'elaboració del codi. En primer lloc, s'ha elaborat un llistat que servís com a esbós visual del programa, per assegurar que inclou totes les parts necessàries. S'ha determinat que el programa ha de comptar amb una secció destinada a verificar un número de targeta i una altra per calcular un dígit que falti en un número incomplet. Així, s'arriba a la conclusió que ha de constar d'una pantalla d'inici, un programa "VERIFICADOR" basat en l'algoritme de Luhn i un programa "CALCULADORA" per determinar el dígit que falta.

La pantalla d'inici s'ha dissenyat amb l'objectiu de ser visualment clara i senzilla, de manera que centralitzi les dues funcions del programa en un únic entorn. S'han col·locat dos botons centrats, l'un sota l'altre, amb el text "VERIFICADOR" i "CALCULADORA", de manera que en prémer-los l'usuari accedeixi directament a la secció corresponent. Per millorar la comoditat i la usabilitat, també s'ha afegit un botó de "Sortir" a la part superior dreta, que permet tancar el programa amb un sol clic.

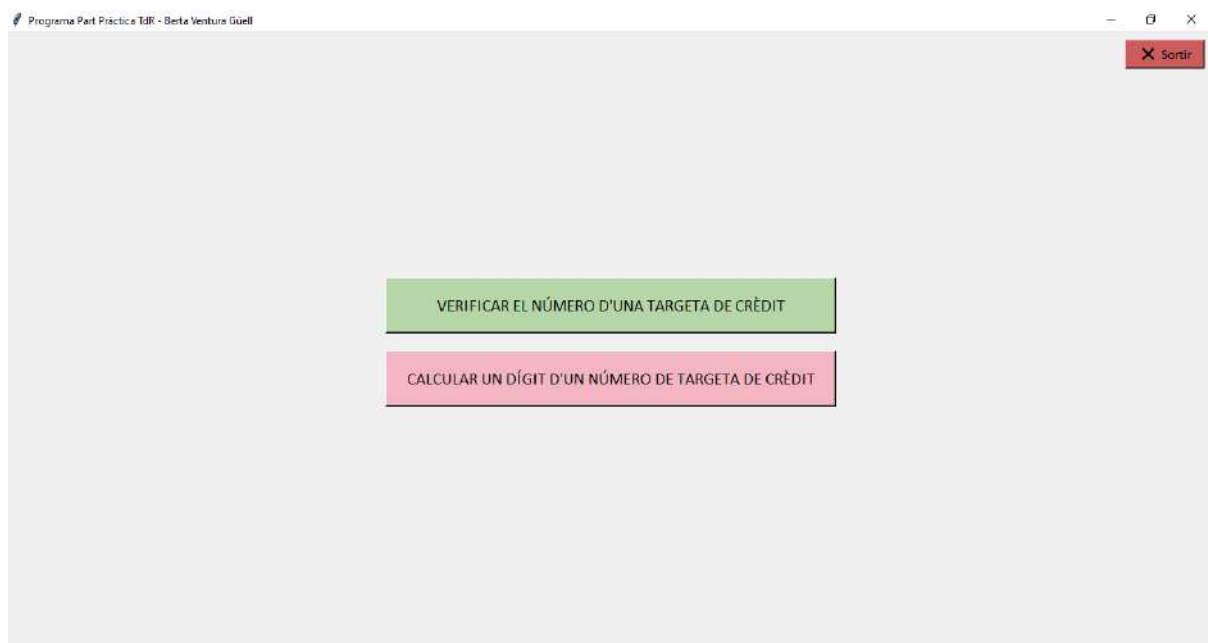


Figura 5. Captura del resultat final de la pantalla d'inici del programa. (Elaboració pròpia, 2025)

El programa del verificador està pensat per permetre introduir un número de targeta de crèdit i comprovar-ne la validesa. Per aconseguir-ho, s'ha creat una entrada de

text, és a dir, un requadre on l'usuari pot escriure el número de targeta, i a sota, un teclat tàctil de 3 x 4 amb els números del 0 al 9, un botó d'esborrar (AC) i un botó "VALIDAR". Tots aquests elements se centren per millorar l'estètica i facilitar la interacció. Entre l'entrada de text i el teclat, es deixa un espai per mostrar missatges de validació indicant si el número introduït és correcte o incorrecte, així com alertes quan es fa servir un número amb menys de 16 dígit. Per evitar errors, es bloqueja la possibilitat d'introduir caràcters no numèrics, limitant l'ús exclusiu del teclat creat amb el programa, i també es restringeix l'entrada a 16 dígit per impedir números amb una quantitat superior de caràcters. A més, s'inclou un requadre amb un títol i una breu explicació del funcionament del programa. També es mostra un missatge a la part inferior centrada que adverteix que, tot i que un número sigui vàlid segons l'algoritme de Luhn, això no és garantia que sigui un número real o existent. Finalment, s'afegeix un botó per tornar a la pantalla d'inici per així permetre a l'usuari navegar fàcilment pel programa.



Figura 6. Captura del resultat final del verificador del programa. (Elaboració pròpia, 2025)

Pel que fa al programa de la calculadora, aquest ha de permetre introduir un número incomplet de targeta i calcular el dígit que falta. De manera similar al verificador, es necessita una entrada de text i un teclat, però en aquest cas es creen 16 caixes separades, una per cada dígit, de manera que el número es vegi complet, però cada dígit quedi en una cel·la individual. El teclat inclou un botó amb "?" per indicar la posició del dígit a calcular. Per garantir un funcionament correcte, es bloqueja

l'entrada de més d'un caràcter per cel·la i es programa que, un cop introduït un dígit en una cel·la, el següent es col·loqui automàticament a la següent. També es bloqueja introduir més d'un '?', mostrant un missatge informatiu si s'intenta fer-ho, ja que només es pot calcular un dígit alhora. Un cop premut el botó de calcular, el programa mostra el número incomplet, el dígit calculat i el número complet resultant. Igual que en el verificador, s'afegeix un missatge d'avís indicant que el dígit calculat permet complir l'algoritme de Luhn, però que això no garanteix que el número sigui d'una targeta existent. Finalment, també s'inclou un botó per tornar a la pantalla d'inici.



Figura 7. Captura del resultat final de la calculadora de dígit del programa. (Elaboració pròpia, 2025)

3.1.3. Fases de l'elaboració del codi

Per dur a terme el desenvolupament del programa, s'han seguit diverses fases que permeten organitzar el procés de manera clara i progressiva. Com que el programa consta de dos subprogrames, es comença elaborant el verificador i la calculadora de manera separada. Aquesta estratègia permet assegurar el correcte funcionament de cadascun abans de combinar-los en un únic programa, on posteriorment s'afegeix la pantalla d'inici des d'on es pot accedir a ambdós subprogrames.

Abans de crear aquests subprogrames amb Python, s'ha elaborat un diagrama de flux per tal de fer més visuals tots els passos que s'hauran de codificar:

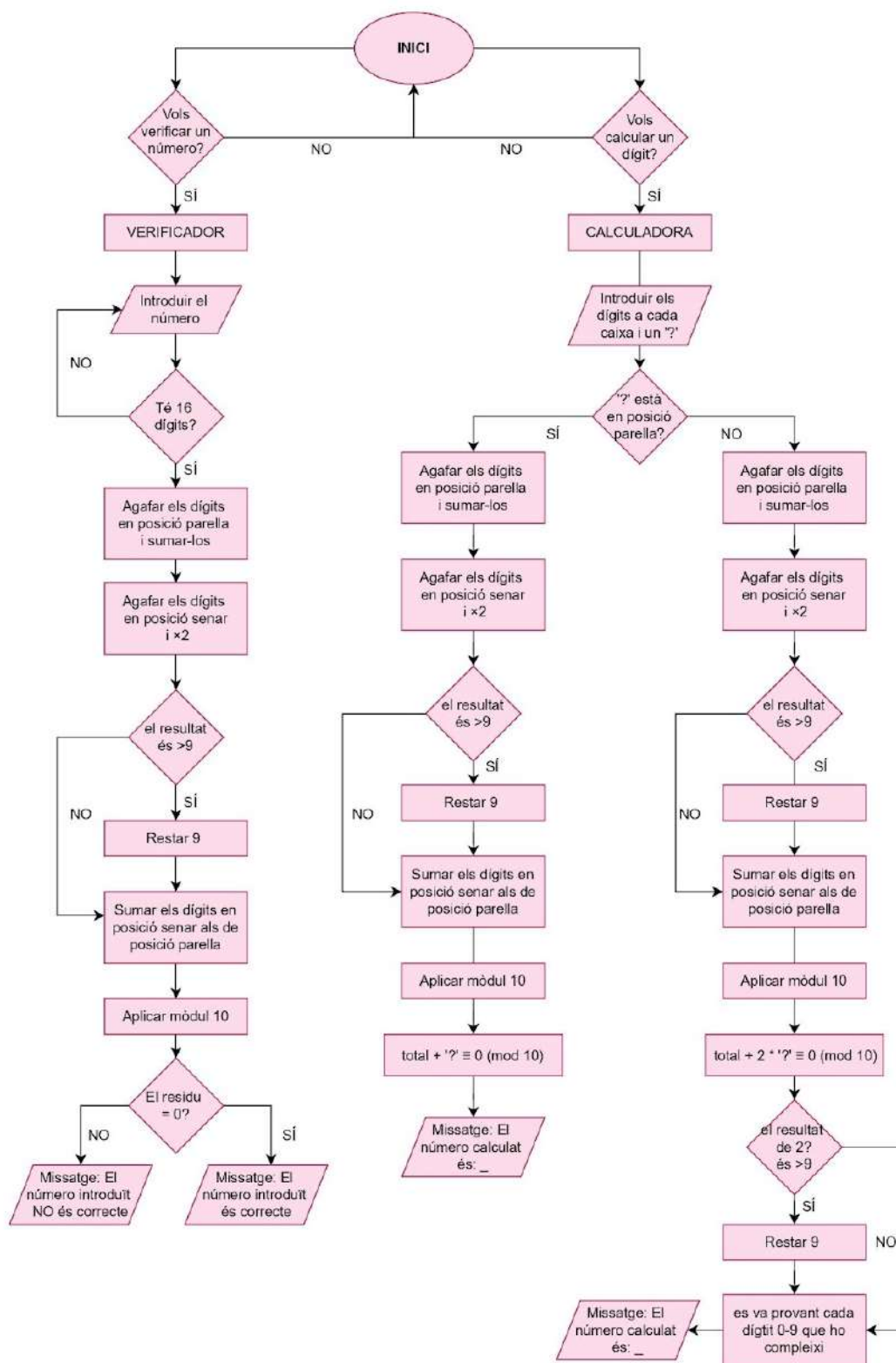


Figura 8. Diagrama de flux dels subprogrames. (Elaboració pròpia, 2025)

Es comença pel programa del verificador. En primer lloc, es creen tots els *widgets* i es van disposant en la pantalla segons el disseny planificat, amb la garantia que la seva ubicació i funció coincideixi amb l'esbós inicial. A continuació, s'implementen les funcions necessàries per al funcionament del programa, com ara les de reiniciar l'entrada, esborrar dades i verificar la validesa del número de targeta. Amb això, s'obté una primera versió operativa del codi, la qual serveix com a base funcional sobre la qual es treballa posteriorment.

La fase següent consisteix a desenvolupar el programa de la calculadora de dígit. Aquest subprograma requereix més atenció a causa de la seva complexitat i dels requisits específics que ha de complir. Malgrat això, les etapes de treball són similars a les del verificador: primer la creació dels *widgets* i, a continuació, la implementació de les funcions corresponents, adaptades a les particularitats del càlcul del dígit que falta.

Un cop comprovada l'eficàcia i la correcta execució de cada subprograma de manera independent, s'elabora la versió definitiva del codi. En aquesta etapa final, es crea primer la pantalla d'inici i després s'hi incorporen els dos subprogrames previs, aconseguint així unificar tota la funcionalitat en un sol programa complet i coherent, amb accés centralitzat a les dues eines des de la mateixa interfície.

3.1.4. Codi del programa

En aquest apartat s'explica una mica com funciona el codi del programa que s'ha creat. Però és important abans explicar alguns conceptes bàsics de programació per facilitar la comprensió de les pròximes pàgines. Aquesta explicació està basada en el llenguatge Python i concretament amb l'ús de la llibreria Tkinter, ja que és el que s'ha utilitzat en aquest treball.

3.1.4.1. Bases de la programació amb Python i Tkinter

Tkinter és una biblioteca que permet construir interfícies gràfiques d'usuari (GUI), és a dir, finestres amb botons, caixes de text i altres elements que faciliten la interacció amb el programa. Sense aquesta llibreria, el funcionament del codi quedaria restringit a la consola, però gràcies a Tkinter es pot aconseguir un programa molt

més intuïtiu i fàcil d'utilitzar. Així doncs, per tal d'utilitzar tot el que conté aquesta llibreria cal importar-la. Per fer-ho cal escriure la següent línia a l'inici del codi:

```
from tkinter import *
```

El primer que cal entendre és que en Tkinter tot es basa en la creació d'una finestra principal, que actua com a contenidor. Aquesta finestra s'inicialitza mitjançant la classe Tk(). Sobre aquest espai s'hi poden anar col·locant tots els elements que l'usuari veurà i amb els quals interactuarà. Un exemple és:

```
finestra = Tk()
```

Els elements bàsics que ofereix Tkinter són els anomenats *widgets*. Aquests es poden entendre com les peces fonamentals d'una interfície gràfica. Alguns dels *widgets* més utilitzats són:

Button: és un botó que, quan es clica, executa una acció determinada. A Tkinter, un botó es crea amb el *widget* Button i se li associa una funció que indica què ha de fer quan s'activi.

```
boto = Button(finestra, text="2", font=("Calibri 12"), width=5,  
             height=2, command=lambda: clic_boto(2), bg="red"))
```

Q En aquesta línia, text defineix el que apareix al botó, font el tipus i mida de lletra, width i height les dimensions del botó (amplada i alçada), command=lambda la funció (clic_boto(2)) que s'executa quan es clica, i bg el color de fons.

Label: serveix per mostrar un text fix dins la finestra. És útil per donar informació o indicar què s'ha d'escriure en un camp.

```
etiqueta = Label(finestra, text="Hola", bg="grey")
```

Entry: és una caixa de text en la qual l'usuari pot introduir dades. Al programa, per exemple, és on s'escriu el número de targeta que es vol validar.

```
entrada = Entry(finestra, font=("Calibri 12"))
```

A part dels *widgets*, Tkinter també ofereix diferents maneres de col·locar aquests elements a la finestra. Les que s'han utilitzat en el programa són les següents:

El mètode `grid()` organitza els *widgets* dins d'una graella de files i columnes, la qual cosa permet un disseny més estructurat.

```
widget.grid(row=0, column=1, columnspan=2, padx=10, pady=5)
```

Q Aquí `row` i `column` indiquen la fila i columna dins de la graella, `columnspan` la quantitat de columnes que ocupa, i `padx/pady` l'espai al voltant del widget.

D'altra banda, `place()` permet col·locar els elements indicant coordenades concretes.

```
widget.place(relx=1, rely=0, anchor='ne', x=-10, y=10)
```

Q `relx` i `rely` indiquen la posició relativa dins la finestra, `anchor` defineix quin punt (cardinal) del *widget* s'alinea amb aquestes coordenades, i `x` i `y` són desplaçaments addicionals (marges).

Un cop col·locats tots els *widgets*, és necessari que el programa entri en un estat d'“espera activa” en què pugui reaccionar a les accions de l'usuari. Això es fa amb el mètode `mainloop()`, que manté la finestra oberta i escoltant fins que es tanqui manualment.

```
finestra.mainloop()
```

A més dels *widgets*, és necessari entendre alguns conceptes de programació que són fonamentals per al funcionament del codi. Un d'aquests conceptes són les variables, que es poden entendre com contenidors que guarden informació que el programa pot utilitzar i modificar. Per exemple:

```
i = 7
```

Aquesta línia de codi que s'ha posat com a exemple crea una variable anomenada “i” i li assigna el valor 7. Però, aquesta informació es pot modificar posteriorment, per exemple sumant o restant valors:

```
i = i + 2
```

```
i = i - 1
```

En els casos anteriors, la variable `i` inicialment tenia un valor de 7, però després de les modificacions aquest valor ha canviat. En el primer cas ha passat a valdre 9 i en

el segon 6. Això ha estat possible gràcies als operadors que s'han utilitzat, + i - . A part d'aquests, a Python també se'n troben d'altres, alguns són:

Taula 9. Taula dels operadors en el llenguatge Python. (Elaboració pròpia, 2025)

Símbol	Operació
+	Suma
-	Resta
*	Multiplicació
/	Divisió
**	Potència
%	Mòdul

Python reconeix automàticament el tipus de dada que conté la variable, que pot ser un nombre enter (`int`), un número decimal (`float`), una cadena de text (`str`) o un valor booleà¹⁴ (`bool`), que només pot ser `True` o `False`.

Una altra part clau de qualsevol programa són els condicionals, declaracions que permeten decidir què fer segons si una condició és certa o falsa. La condició principal és `if`, que executa un bloc de codi només si la seva expressió és certa. En el cas que no ho sigui, si es vol mostrar una acció, cal utilitzar la declaració `else`. De la mateixa manera, també es pot utilitzar `if not` per executar una acció quan es fa referència a una condició que d'entrada no es compleix.

L'expressió del condicional conté un argument que es representa amb diferents símbols que poden indicar igualtat, diferència, més gran/petit que...

Taula 10. Taula dels símbols que corresponen als diferents condicionals. (Elaboració pròpia, 2025)

==	és exactament igual a ...
!=	és diferent a ...
<	el valor és més petit que ...
>	el valor és més gran que ...
<=	el valor és més petit o igual que ...
>=	el valor és més gran o igual que ...

¹⁴ Un booleà és un tipus de dada que només pot tenir dos valors possibles: `True` (cert) o `False` (fals). Serveix per indicar si alguna cosa és certa o no dins del programa, com per exemple el resultat d'una comparació o d'una condició.

A part dels condicionals, també és molt habitual en programació haver de repetir certes instruccions diverses vegades. Per aconseguir-ho existeixen els bucles, que permeten executar un mateix bloc de codi repetidament sense necessitat d'escriure'l manualment una vegada i una altra. El bucle que s'ha utilitzat en la codificació del programa ha estat el `for`, que serveix per recórrer una seqüència d'elements, com pot ser una llista de números, un text o qualsevol col·lecció de dades. Per exemple, si es vol mostrar per pantalla els números de l'1 al 5, es podria escriure:

```
for i in range(1,6):  
    print(i)
```

Q En aquest cas, la funció `range(1,6)` genera una successió de números des de l'1 fins al 5, i el bucle `for` fa que, a cada volta, la variable `i` prengui un d'aquests valors. Així, la instrucció `print(i)` s'executa cinc vegades, mostrant primer l'1, després el 2 i així successivament fins al 5.

És important remarcar que tant dins d'un condicional com dins d'un bucle, totes les instruccions que en formen part s'han d'escriure amb un desplaçament cap a la dreta per mitjà d'una tabulació (tal com es veu a l'exemple anterior). Aquest desplaçament és el que indica a Python que aquelles línies pertanyen al mateix bloc de codi i, per tant, s'han d'executar conjuntament. Si no es respecta aquest format, el programa mostrarà un error i no s'executarà correctament.

Els bucles i els condicionals sovint es combinen. Per exemple, es podria fer un programa que recorri una llista de números i imprimeixi només aquells que siguin parells:

```
for i in range(10):  
    if i % 2 == 0:  
        print(i, "és un número parell")
```

Q Aquí, el bucle `for` recorre els números del 0 al 9 i el condicional comprova si cadascun d'ells és divisible per 2. Si ho és, mostra el missatge corresponent.

Un altre concepte molt útil per entendre com funciona el codi són les funcions. Una funció és un conjunt d'instruccions agrupades sota un mateix nom que es poden reutilitzar sempre que calgui. El gran avantatge és que permeten organitzar el codi, fer-lo més entenedor i evitar haver d'escriure el mateix codi repetidament. Per crear

una funció en Python s'utilitza la paraula clau `def`, seguida del nom que se li vol donar i d'uns parèntesis. Per exemple:

```
def saluda(nom):  
    print("Hola", nom)
```

Un cop definida, aquesta funció es pot cridar tantes vegades com es vulgui simplement escrivint el seu nom i posant-li l'argument necessari entre parèntesis:

```
saluda("Mònica")
```

El resultat seria que el programa mostra "Hola Mònica".

A més, les funcions poden retornar un valor utilitzant la paraula clau `return`. Per exemple:

```
def quadrat(x):  
    return x*x
```

🔍 Si es crida `quadrat(4)`, el programa retornarà¹⁵ 16, ja que multiplica el número per ell mateix. Aquesta característica és essencial perquè permet dividir un programa complex en petites parts més senzilles, cadascuna amb una funció concreta.

Finalment, en el codi, a part de totes les declaracions que s'han mencionat, també s'hi poden afegir textos en forma de comentari afegint un `#` a l'inici de la línia. Aquests a l'hora de posar en funcionament el programa no es tenen en compte i són molt útils per introduir explicacions dintre del mateix codi. En els casos que és necessari fer un comentari en diverses línies, s'escriuen `"""` a l'inici i final. Així doncs, per comentar, es fa de la següent manera:

```
# això és un comentari  
  
"""  
així es pot  
escriure el comentari  
en múltiples línies  
"""
```

¹⁵ Quan s'utilitza el terme retornar, es refereix al valor o resultat que una funció o declaració dona després d'executar-se, és a dir, el valor que es "mostra" o que es pot utilitzar més endavant en altres parts del programa.

3.1.4.2. Funcionament del codi

Un cop vista la forma general en la qual funciona el llenguatge de programació que s'ha utilitzat, és el torn d'explicar com funciona el codi del programa desenvolupat. Per tal de fer-ho més entenedor, s'ha organitzat l'explicació en tres grans blocs, que corresponen a les tres pantalles del programa: la d'inici, la del verificador i la de la calculadora. Aquesta mateixa distribució també es reflecteix en la manera com està estructurat el codi, fet que facilita la seva lectura i comprensió.

Ara bé, abans d'entrar en el detall de cadascuna d'aquestes pantalles, és necessari entendre una funció que constitueix la primera part del codi i que és clau per a tot el funcionament del programa: `mostrar_frame`. Aquesta funció és la que permet gestionar les diferents pantalles dins d'una mateixa finestra, mostrant-ne només una cada vegada i amagant les altres. Això s'aconsegueix fent servir el concepte de *frames* (caixes) que Tkinter permet col·locar dins la finestra principal.

```
def mostrar_frame(frame_a_mostrar):  
    """Amaga tots els frames i mostra el demanat."""  
    for f in (frame_inici, frame_verificador, frame_calculador):  
        f.pack_forget()  
        frame_a_mostrar.pack(expand=True, fill="both")  
  
    # Mostra/amaga el botó d'inici segons la pantalla  
    if 'boto_inici' in globals():  
        if frame_a_mostrar is frame_inici:  
            boto_inici.place_forget() # no mostrar a la pàgina d'inici  
        else:  
            boto_inici.place(relx=0, rely=0, anchor='nw', x=10, y=10)
```

Q Aquesta funció recorre tots els *frames* del programa i els amaga amb `pack_forget()`. Tot seguit, mostra el *frame* que es vol visualitzar amb `pack(expand=True, fill="both")`, de manera que ocupa tot l'espai disponible. A més, controla la visibilitat del botó d'inici: aquest no es mostra quan s'està a la pantalla inicial, però sí a la resta, situat a la cantonada superior esquerra. Aquesta lògica és la que fa possible navegar de manera fluida dins del programa.

Fragment de codi de la pantalla d'inici

Un cop entesa aquesta funció bàsica, es pot passar a explicar cadascuna de les pantalles començant per la d'inici. La pantalla d'inici té un paper molt senzill: actua com a punt de partida des d'on l'usuari pot accedir a cadascun dels dos subprogrames. En aquesta part del codi, primer de tot es creen les funcions que permetran obrir cadascuna de les pantalles. Aquestes funcions són les encarregades de cridar `mostrar_frame`, que és la que s'encarrega de canviar la finestra que es mostra en aquell moment.

```
def anar_a_verificar():  
    mostrar_frame (frame_verificador)  
  
def anar_a_calculadora():  
    mostrar_frame (frame_calculador)
```

🔍 En aquestes funcions es veu que, en prémer el botó corresponent, el programa no crea una finestra nova, sinó que mostra el *frame* que ja conté tots els elements de la pantalla desitjada. Aquesta és una manera eficient de gestionar diferents pantalles dins d'una mateixa aplicació.

Un cop creades les funcions, es defineixen els botons que apareixen a la pantalla d'inici. En total n'hi ha tres: un per accedir al verificador, un altre per obrir la calculadora i un tercer situat a la part superior dreta que permet sortir del programa. Els dos primers estan centrats a la pantalla i tenen un text descriptiu que facilita entendre a quina funcionalitat condueixen:

```
boto_verificar= Button(contenedor_inicio, text="VERIFICAR EL NÚMERO D'UNA  
TARGETA DE CRÈDIT", font=("Calibri 14"), width=50, height=2,  
command=anar_a_verificar, bg="#b6d7a8")  
  
boto_calcular= Button(contenedor_inicio, text="CALCULAR UN DÍGIT D'UN  
NÚMERO DE TARGETA DE CRÈDIT", font=("Calibri 14"), width=50, height=2,  
command=anar_a_calcular, bg="#f4b6c2")
```

🔍 Aquí el paràmetre `command` és el que estableix la connexió entre cada botó i la funció que s'executarà quan l'usuari el premi. Així, en clicar el botó "VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT", el programa crida la funció `anar_a_verificar`, que mostra la pantalla corresponent; de la mateixa manera passa amb la calculadora.

Fragment de codi de la pantalla del verificador i la calculadora

Les pantalles del verificador i de la calculadora són més complexes i, tot i que cadascuna té elements específics, comparteixen una mateixa estructura: primer es defineixen les funcions que han de donar funcionalitat al programa, i després es creen i es disposen els *widgets* (entrades de text, botons, etiquetes) dins la finestra. Tanmateix, les funcions són les que permeten que el programa faci alguna cosa més que mostrar una interfície buida. Per aquest motiu, l'explicació se centrarà en aquestes.

Funcions comunes

És important detallar una sèrie de funcions que són comunes a les dues pantalles, ja que aquestes permeten controlar com l'usuari introdueix els valors, corregir possibles errors i reiniciar completament els camps quan cal. Per tal de fer-ho més entenedor, s'inclouen fragments del codi com a exemples, de manera que es pugui seguir la lògica pas a pas. Tanmateix, cal destacar que les funcions que es presenten a continuació corresponen al codi del verificador, tot i que la seva lògica és exactament la mateixa que s'utilitza a la calculadora; per aquest motiu, només s'exposen una vegada per simplificar l'explicació.

Primer de tot, quan l'usuari prem un botó numèric, el programa necessita afegir aquest valor al camp d'entrada corresponent. A la pantalla del verificador, això es fa mitjançant la funció `click_boto`, que insereix el valor premut a la posició actual del camp d'entrada i actualitza la variable que indica aquesta posició. A la calculadora, la mateixa idea s'aplica utilitzant la llista `caixes` i la variable `index_actual` per controlar la posició actual d'entrada.

```
def clic_boto(valor):  
    nonlocal i  
    if i < 16:  
        text1.insert(i, str(valor))  
        i += 1
```

🔍 Aquesta funció defineix `valor` com el número que s'ha premut. La paraula clau `nonlocal i` indica que la funció utilitza la variable `i` declarada fora d'ella, que guarda la posició actual on s'ha d'inserir el dígit. L'estructura `if i < 16:` assegura que només s'introdueixin fins a 16 dígitos. Finalment, `text1.insert(i,`

`str(valor))` insereix el valor en la posició `i` del camp d'entrada, i `i += 1` incrementa la posició per al següent clic.

Així mateix, és important considerar que durant la introducció de dades poden aparèixer errors, i per això existeix la funció `esborrar`, que permet eliminar l'últim dígit afegit. A la calculadora, el mateix comportament es controla amb `index_actual` i la llista de caixes corresponent.

```
def esborrar():  
    nonlocal i  
    if i > 0:  
        text1.delete(i - 1)  
        i += 1
```

Q Aquesta funció accedeix a la variable `i` amb `nonlocal` per referir-se a la posició actual d'inserció. El condicional `if i > 0:` comprova que hi hagi algun dígit per eliminar. `text1.delete(i - 1)` esborra el caràcter a la posició anterior, i `i -= 1` decrementa la posició per mantenir el control correcte de la següent inserció.

Finalment, en situacions en què cal començar de nou, sigui per provar un número diferent o netejar tota l'entrada, es fa servir la funció `reset`. Aquesta elimina tot el contingut dels camps, reinicia la variable que indica la posició inicial i neteja qualsevol missatge de validació. A la calculadora, aquesta acció s'aplica a totes les caixes i a la variable `index_actual`.

```
def reset():  
    nonlocal i  
    text1.delete(0, END)  
    i = 0  
    etiqueta_missatge.config(text="")  
    avis_verificador.place_forget()
```

Q Aquí, `nonlocal` i permet modificar la variable de posició fora de la funció. `text1.delete(0, END)` esborra tot el contingut del camp d'entrada. `i = 0` reinicia la posició inicial. `etiqueta_missatge.config(text="")` neteja qualsevol missatge que s'hagi mostrat, i `avis_verificador.place_forget()` amaga la nota d'avís que només apareix quan el número és correcte.

Funcions concretes

Més enllà d'aquestes funcions bàsiques compartides, cadascun dels subprogrames en té una de principal, que és la que els dona sentit. En el cas del verificador es tracta de comprovar si un número de targeta és vàlid. Aquesta funció¹⁶ segueix l'algoritme de Luhn, que ja s'ha explicat a la part teòrica del treball, i, a més de validar la cadena de dígit, comunica el resultat a l'usuari mostrant missatges a la pantalla.

Per la calculadora de dígit, la funció comprovar¹⁷ té un objectiu diferent, però comparteix la mateixa base: determinar quin dígit (de 0 a 9) substituint el caràcter '?' fa que el número resultant sigui vàlid segons l'algoritme de Luhn.

3.1.4.3. Producte final

La unió de totes aquestes funcions que s'han explicat en detall a l'apartat anterior, juntament amb els *widgets* componen el codi del programa que s'ha creat. A l'Annex III es pot consultar el codi sencer del programa.

Per accedir al programa s'ha de baixar el següent fitxer, però abans cal tenir instal·lada l'aplicació de Phyton en el PC. La versió que s'ha utilitzat en aquest treball és la Phyton 3.13.5. perquè era la més recent en el moment de la seva realització. Tot i això, en l'enllaç que es proporciona s'hi permet baixar l'última versió.

Enllaç des d'on baixar-se l'aplicació Phyton: <https://www.python.org/downloads/>

Fitxer Python del programa:

<https://drive.google.com/file/d/1fSqdCufk6Gl9C2SC6UgtEGCaO-hdfUb/view?usp=sharing>

¹⁶ Per tal de poder comprendre cada bloc i la finalitat de cada instrucció de la funció comprovar del verificador, vegeu ANNEX I on s'hi mostra el mateix codi amb comentaris que l'expliquen detalladament i es veu clarament tot el procés pas a pas.

¹⁷ Per tal de poder comprendre cada bloc i la finalitat de cada instrucció de la funció comprovar de la calculadora de dígit, vegeu ANNEX II on s'hi mostra el mateix codi amb comentaris que l'expliquen detalladament i es veu clarament tot el procés pas a pas.

3.2. Anàlisi del programa

3.2.1. Anàlisi matemàtic de la detecció d'errors amb l'algoritme de Luhn

Quan Hans Peter Luhn va proposar el seu algoritme l'any 1954, l'objectiu principal era trobar un mecanisme senzill i alhora eficient per a la detecció d'errors de digitació en números identificadors llargs, com ara els números de compte o les targetes de crèdit. El repte era aconseguir un sistema capaç de detectar la majoria d'errors humans més habituals sense requerir càlculs massa complexos, de manera que fos fàcilment implementable en els mitjans de l'època [17].

Per tal de dissenyar aquest mètode, Luhn va analitzar primer quins eren els errors més freqüents en la còpia o introducció de seqüències numèriques. Entre aquests destacaven, d'una banda, la substitució d'un sol dígit (per exemple, escriure "5" en lloc de "6"), i de l'altra, la transposició de dos dígitos consecutius (com podria ser escriure "32" en lloc de "23"). Amb aquesta informació, va formular una regla matemàtica basada en la manipulació senzilla dels dígitos: multiplicar per 2 els que ocupen posicions alternes i, en cas que el resultat sigui superior a 9, reduir-lo sumant les seves xifres (operació equivalent a restar 9). Finalment, cal calcular la suma de tots els resultats i verificar si el total és divisible per 10 (aplicar mod 10).

Aquest procediment, conegut com a algoritme de Luhn (que ja ha estat explicat en detall en el marc teòric), permet detectar de manera molt eficaç un ampli ventall d'errors, però no tots. Alguns tipus de confusions numèriques poden passar desapercebuts, i és precisament aquest aspecte el que fa rellevant estudiar matemàticament quins errors són identificats i quins no, així com entendre el motiu que hi ha darrere.

Per això mateix, en el present apartat es presenta una anàlisi detallada dels diferents casos d'error. L'objectiu és mostrar, per mitjà de les justificacions matemàtiques corresponents i d'una forma clara i entenedora, fins a quin punt el mecanisme ideat per Luhn és eficaç i en quines situacions pot fallar.

3.2.1.1. Substitució d'un sol dígit

El primer tipus d'error a analitzar, i probablement també el més freqüent, és el de la substitució d'un únic dígit. En principi, es podria pensar que aquest error hauria de ser detectat en tots els casos, ja que el simple fet de canviar un sol número modifica

el valor parcial assignat a aquest dígit i , en conseqüència, altera la suma final. En aplicar mòdul 10, el resultat ja no coincidiria i , per tant, l'error es detectaria.

Tanmateix, per assegurar-se que no existeixi cap excepció, s'ha dut a terme una anàlisi de totes les possibilitats. Per tal de facilitar-ne la comprensió, es presenta a continuació una taula amb els valors que pot prendre cada dígit segons si es troba en una posició de càlcul parella (on es multiplica per 1) o en una posició senar (on es multiplica per 2 i, si el resultat supera 9, se sumen les xifres o bé se'n resta 9):

Taula 11. Taula dels valors que obtenen els dígit d'un nombre de targeta segons la seva posició al aplicar-los l'algoritme de Luhn. (Elaboració pròpia, 2025)

Dígits	0	1	2	3	4	5	6	7	8	9
Valor dels dígit quan estan en posició parella ($\times 1$)	0	1	2	3	4	5	6	7	8	9
Valor dels dígit quan estan en posició senar ($\times 2$)	0	2	4	6	8	10	12	14	16	18
	0	2	4	6	8	1	3	5	7	9

En observar aquesta taula, es pot comprovar que tots els valors assignats als dígit en posició parella són diferents entre ells, i el mateix passa amb els valors dels dígit en posició senar. Això implica que la substitució d'un dígit per un altre sempre altera el valor parcial corresponent, i en conseqüència també la suma total.

Ara bé, caldria preguntar-se si aquesta variació podria, en algun cas, donar lloc a un total amb el mateix residu en aplicar el mòdul 10, de manera que l'error passés inadvertit. Perquè això fos possible, la diferència entre dos valors hauria de ser exactament 10 o un múltiple de 10. Tanmateix, tal com es pot veure a la taula, la diferència màxima entre valors és de 9 (entre el 0 i el 9), i mai no s'arriba a 10.

Així doncs, es pot concloure que l'algoritme de Luhn detecta el 100% dels casos de substitució d'un sol dígit, sense excepció.

3.2.1.2. Transposició de dos dígit consecutius

La transposició de dos dígit consecutius és un tipus d'error força freqüent, i per això mereix una anàlisi detallada. Quan es produeix una transposició d'aquest tipus, en realitat s'està intercanviant la posició d'un dígit que està en una posició senar amb

un altre que es troba en una posició parella. Aquest fet és rellevant perquè l'algoritme de Luhn aplica regles diferents segons si el dígit ocupa una posició o l'altra: els que es troben en posicions senars es multipliquen per 2, mentre que els que estan en posicions parelles es mantenen igual, és a dir, es multipliquen per 1. Per tant, l'efecte d'una transposició de dos dígitos consecutius és equivalent al d'intercanviar dos dígitos que es troben en posicions diferents, una senar i una parella.

Per deixar-ho més clar s'ha creat aquesta figura:



En un primer moment podria semblar que tots aquests errors haurien de ser detectats sense excepció, ja que en desplaçar un dígit d'una posició senar a una de parella (o viceversa) es modifica el valor assignat a aquest dígit en el càlcul de l'algoritme. En conseqüència, la suma total canviaria i, en aplicar-hi el mòdul 10, també canviaria el residu, cosa que faria visible l'error.

Ara bé, cal matisar aquesta primera impressió. Tal com ja s'ha explicat anteriorment, el fet que s'apliquin dos càlculs diferents segons la posició del dígit influeix de manera directa en la suma final i, per extensió, en el resultat de l'aplicació del mòdul 10. Aquesta operació permet visualitzar clarament la diferència que experimenta un dígit quan canvia de posició. Tot i això, si hi hagués dos dígitos que, un cop aplicada la diferència mòdul 10, oferissin el mateix resultat, l'error de transposició no seria detectat, ja que l'algoritme produiria el mateix residu. Per aquest motiu, s'ha optat per calcular la diferència mòdul 10 per a cadascun dels dígitos i comparar-ne els resultats.

El mecanisme que permet determinar si un error de transposició serà detectat és, doncs, la diferència mòdul 10. Aquesta diferència es calcula restant el valor que tindria un dígit si estigués en una posició senar del valor que tindria si estigués en una posició parella, i posteriorment aplicant mòdul 10 al resultat. D'aquesta manera es pot veure si el canvi de posició d'un dígit provoca o no una alteració en el resultat final de l'algoritme.

Per entendre-ho millor, es pot observar el cas concret del dígit 7. Quan ocupa una posició senar, s'ha de multiplicar per 2 i això dona 14; com que el resultat és superior a 9, se li resta 9 i s'obté 5 com a valor final. En canvi, quan el 7 ocupa una posició parell, el seu valor es manté en 7. La diferència entre aquests dos valors és $5 - 7 = -2$, i en aplicar mòdul 10 se n'obté 8. Això vol dir que la diferència mòdul 10 del dígit 7 és 8.

Seguint aquest procediment amb tots els dígit del 0 al 9 es pot completar la taula corresponent, en la qual es recullen els valors que prenen en posicions senars i parelles, així com la seva diferència mòdul 10.

Taula 12. Taula dels valors que obtenen els dígit quan se'ls aplica l'algoritme de Luhn. (Elaboració pròpia, 2025)

Dígits	0	1	2	3	4	5	6	7	8	9
Valor dels dígit quan estan en posició parella ($\times 1$)	0	1	2	3	4	5	6	7	8	9
Valor dels dígit quan estan en posició senar ($\times 2$)	0	2	4	6	8	10	12	14	16	18
	0	2	4	6	8	1	3	5	7	9
Diferència mòdul 10	0	1	2	3	4	6	7	8	9	0

L'anàlisi d'aquesta taula mostra que tots els dígit presenten diferències distintives excepte el 0 i el 9, que comparteixen la mateixa diferència mòdul 10. Aquest fet implica que si s'intercanvien un 0 i un 9 entre una posició senar i una de parella, el valor final de la suma no varia i, en conseqüència, l'algoritme de Luhn no detecta l'error.

Això vol dir que, en la pràctica, l'algoritme és capaç de detectar qualsevol transposició de dos dígit consecutius excepte en el cas concret en què es passa de 09 a 90 o viceversa.

3.2.1.3. Transposició de dos dígitos en posicions d'igual paritat¹⁸

Aquest tipus d'error és menys habitual, però és rellevant analitzar-lo, ja que s'ha provat que mai és detectat per l'algoritme de Luhn. Per comprendre-ho, es pot utilitzar el mateix esquema aplicat en l'apartat anterior:

×2	×1	×2	×1	×2	×1	×2	×1	×2	×1	×2	×1	×2	×1	×2	×1
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

En observar aquesta figura, es pot veure que els dígitos situats en posicions d'igual paritat —és a dir, tots els que es troben en caselles parelles o tots els que es troben en caselles senars, pintats amb el mateix color— reben els mateixos càlculs. Per tant, l'intercanvi de dos dígitos dins d'aquestes posicions no modifica el resultat del càlcul, ja que cada dígit continua sent sotmès a la mateixa operació multiplicativa assignada a la seva posició.

Per il·lustrar-ho amb un exemple concret, es considera el següent número:

4	5	3	9	3	1	9	5	0	3	4	3	6	4	6	7
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Si s'aplica una transposició entre els dígitos 3 i 9 situats en caselles roses, el número resultant seria:

4	5	3	9	9	1	3	5	0	3	4	3	6	4	6	7
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Com es pot observar, tot i haver intercanviat els dígitos, ambdós continuen situats en caselles roses, cosa que implica que se'ls continuarà aplicant el mateix càlcul multiplicatiu (×2) tal com passaria en el número original. En conseqüència, la suma total no es veu afectada i el residu del mòdul 10 tampoc canvia. El mateix raonament s'aplica si els dígitos intercanviats es trobessin en caselles verdes: en aquest cas no rebrien cap modificació, i la propietat commutativa de la suma ($x + y = y + x$) garanteix que el resultat final es mantingui idèntic.

Així doncs, qualsevol transposició de dos dígitos en posicions d'igual paritat passa completament inadvertida per l'algoritme de Luhn i, per tant, mai és detectada.

¹⁸ A nivell matemàtic o d'informàtica, quan es diu que dos dígitos estan en posicions de la mateixa paritat, significa que tots dos estan en posicions parelles, o bé tots dos en posicions senars.

3.2.2. Comprovació del funcionament del programa

Un cop analitzats els errors que es poden produir i determinats quins són detectats per l'algoritme de Luhn, és possible comprovar si el programa creat funciona correctament. Aquesta verificació es realitza amb el coneixement previ de l'abast del programa, és a dir, sabent quins errors haurien de ser identificats i quins no.

3.2.2.1. Pantalla del verificador

Per validar el correcte funcionament del verificador, s'ha elaborat una taula amb diferents casos de prova, que inclouen tant números complets, incomplets i invàlids, com situacions específiques que simulen errors de digitació. Cada número s'ha introduït al programa i, comparant el resultat obtingut amb el resultat esperat, s'ha pogut determinar si el programa operava de manera adequada¹⁹.

Taula 13. Taula d'anàlisi de casos al utilitzar el verificador del programa. (Elaboració pròpia, 2025)

	Número introduït	Descripció de la prova	Resultat esperat	Resultat del programa
Cas 1	86570	Número incomplet	invàlid	invàlid
Cas 2	5796428097536978	Número inventat	invàlid	invàlid
Cas 3	4539319503436467	Número real vàlid	vàlid	vàlid
Cas 4	4539319503436468	Substitució d'un sol dígit	invàlid	invàlid
Cas 5	4530319593436467	Intercanvi de posició dels dígit 0 i 9, un en posició senar i un altre en posició parella (consecutius o no).	vàlid (no pot detectar l'error)	vàlid
Cas 6	4539319503436476	Intercanvi de posició de dos dígit diferents a 0 i 9, un en posició senar i l'altre en posició parella (consecutius o no)	invàlid	invàlid
Cas 7	4539319503436764	Intercanvi de posició de dos dígit en posicions parells o senars (les dues)	vàlid (no pot detectar l'error)	vàlid

¹⁹ Vegeu l'ANNEX IV per visualitzar les captures dels resultats del programa de cada un dels casos analitzats.

En observar la taula es pot concloure que el programa funciona correctament, ja que els resultats obtinguts coincideixen amb els resultats esperats en tots els casos possibles d'error detectables per l'algoritme, mentre que aquells errors que l'algoritme no pot detectar apareixen tal com s'ha predit.

3.2.2.2. Pantalla de la calculadora

En el cas de la calculadora, el funcionament es preveu correcte, ja que els errors de digitació no afecten els càlculs, els quals es realitzen de manera individual per a cada dígit. Per comprovar-ho, s'ha introduït el número complet 4539319503436467 i s'ha identificat cada dígit com a incògnit, des del primer fins a l'últim. El resultat obtingut per a cada dígit ha coincidit amb el valor corresponent en el número complet²⁰. Aquest procediment permet concloure que la calculadora opera correctament i proporciona els valors esperats per a cada càlcul individual.

3.3. Difusió a la Viquipèdia

Un dels objectius plantejats en aquest treball és difondre el coneixement adquirit, i per això s'ha escollit la Viquipèdia com a eina de divulgació. En revisar l'entrada existent sobre l'algoritme de Luhn en català, s'ha constatat que és molt més breu i limitada en comparació amb la versió en castellà. Per aquest motiu, s'ha considerat oportú aprofitar tota la informació recollida i treballada per ampliar l'article en català i, d'aquesta manera, contribuir a fer arribar aquest contingut a un públic més ampli.

Per dur a terme aquesta tasca, s'ha treballat sobre l'entrada existent, afegint-hi explicacions més detallades i exemples que permeten comprendre millor el funcionament i les aplicacions de l'algoritme de Luhn. El text complet generat per a la difusió, així com l'enllaç a l'article ampliat, es pot consultar a l'Annex VI del treball.

²⁰ Vegeu l'ANNEX V per visualitzar les captures dels resultats del programa de cada un dels casos analitzats.

4. CONCLUSIONS

En acabar el treball es pot dir que s'han assolit els objectius proposats a l'inici. La manera de treballar ha estat adequada perquè ha combinat una primera part teòrica, que ha donat les bases necessàries per entendre el tema, amb una part pràctica que ha permès comprovar i aplicar aquests coneixements. L'organització seguida en aquesta última part ha estat clau per tal de dur-la a terme sense dificultats: la visualització de tutorials per adquirir els coneixements necessaris, la creació d'un diagrama de flux per planejar el programa, i finalment la codificació del programa per parts. Així doncs, aquesta metodologia ha resultat útil perquè ha facilitat avançar pas a pas, contrastar la informació obtinguda i entendre millor la relació entre els conceptes matemàtics i les aplicacions reals.

Concretament, la recerca ha permès arribar a les següents conclusions:

- La codificació és una eina essencial per assegurar que la informació arribi correctament des del seu origen fins a la destinació. Aplicant transformacions matemàtiques a les dades i afegint-hi informació de control, es pot detectar si hi ha hagut errors durant la transmissió. Comprendre la teoria de la codificació permet veure com aquests sistemes no només detecten errors, sinó que també reforcen la seguretat i la fiabilitat de la informació, facilitant el desenvolupament de processos capaços de protegir dades en situacions diverses, des de comunicacions digitals fins a transaccions bancàries.
- Els codis detectors i correctors d'errors són imprescindibles per garantir la precisió de la informació que es transmet. Mentre que els codis detectors només indiquen que hi ha hagut un error, els codis correctors permeten identificar-ne la ubicació i corregir-lo automàticament. Aquesta comprensió és clau per entendre com funcionen codis aplicats a la vida quotidiana, com els números de les targetes bancàries, els ISBN o els IBAN, i mostra com les matemàtiques asseguruen que la informació sigui segura i útil fins i tot en situacions crítiques.
- Les matemàtiques són imprescindibles per garantir la seguretat i la fiabilitat de les targetes bancàries. L'algoritme de Luhn utilitza l'aritmètica modular per calcular residus que permeten verificar que cada dígit del número compleix les

regles establertes i detectar errors habituals de transcripció o intercanvi de xifres. Aquesta comprovació assegura la validesa estructural del número de targeta.

- L'anàlisi dels errors detectats per l'algoritme de Luhn mostra que aquest detecta de manera completa qualsevol substitució d'un sol dígit, gràcies a l'aplicació de l'aritmètica modular, que assegura que qualsevol canvi alteri el residu final. També és capaç d'identificar gairebé totes les transposicions de dígit consecutius, amb l'única excepció d'algunes combinacions molt concretes (com el canvi de 09 a 90 o viceversa). En canvi, les transposicions de dígit situats en posicions d'igual paritat no són detectades, ja que els càlculs aplicats a aquests dígit no generen variacions en la suma final.
- L'algoritme de Luhn no només permet comprovar si un número de targeta és vàlid, sinó que també fa possible calcular el dígit de control quan es coneixen la resta de xifres. Això és útil perquè el dígit de control és l'element que garanteix la coherència de tot el número: s'escull de manera que la suma final sigui divisible per 10. Així, si es tenen totes les altres xifres, només cal aplicar el procediment de Luhn i determinar quin dígit fa que el resultat sigui múltiple de 10.
- De la mateixa manera, el raonament es pot estendre a qualsevol dígit de la targeta. Si se sap el valor de la resta de xifres, es pot calcular el que falta assegurant que, en aplicar l'algoritme, el resultat obtingut continuï sent divisible per 10. D'aquesta manera, l'algoritme permet deduir qualsevol xifra desconeguda del número de la targeta, no només el dígit de control.
- L'objectiu de difondre el coneixement adquirit sobre els fonaments matemàtics de les targetes de crèdit s'ha assolit en publicar una entrada en català a la Viquipèdia, que fa accessible aquesta informació a qualsevol persona interessada. Així, l'aprenentatge personal s'ha convertit en coneixement compartit i s'ha complert la finalitat divulgadora del treball amb la presència de continguts científics en català.

5. BIBLIOGRAFIA I WEBGRAFIA

- [1] Algoritmo de Luhn. (s. d.). *Matemática Informática y Educación*. Recuperat el març de 2025, de <https://jcastrom.jimdo.com/matematica/matem%C3%A1tica-discreta/aritm%C3%A9tica-modular-y-algoritmo-de-luhn/>
- [2] Aritmética modular. (s. d.). *Matemática Informática y Educación*. Recuperat el març de 2025, de <https://jcastrom.jimdo.com/matematica/matem%C3%A1tica-discreta/aritm%C3%A9tica-modular/>
- [3] Beato Sirvent, J. (2008, febrer). Códigos numéricos para la vida. *SUMA: Revista sobre Enseñanza y Aprendizaje de las Matemáticas*, (57), 43–54. Recuperat el març de 2025, de <https://revistasuma.fespm.es/sites/revistasuma.fespm.es/IMG/pdf/57/043-054.pdf>
- [4] Colaboradores de Wikipedia. (2025, 29 maig). Algoritmo de Luhn. *Wikipedia, la Enciclopedia Libre*. Recuperat l'agost de 2025, de https://es.wikipedia.org/wiki/Algoritmo_de_Luhn
- [5] ¿Cómo se valida una una tarjeta de crédito? (s. d.). Recuperat el març de 2025, de <https://matematicaseducativas.blogspot.com/2011/02/como-se-valida-una-una-tarjeta-de.html>
- [6] Contributors to Wikimedia projects. (2025, 31 març). Codi Hamming. *Viquipèdia, L'enciclopèdia Lliure*. Recuperat el maig de 2025, de https://ca.wikipedia.org/wiki/Codi_Hamming
- [7] Cortés, E. D. (2019, 9 desembre). Respuesta al desafío de Junio, el problema de Hamming. *La Sombra de Dijkstra*. Recuperat el juny de 2025, de <https://www.programando.org/blog/2012/07/08/respuesta-al-desafio-de-junio-el-problema-de-hamming/>
- [8] García, M. A., Martínez, L., & Ramírez, T. (2017, maig). Tema 2: Nociones

- básicas de la Teoría de Códigos [Resumen teórico]. Recuperat el març de 2025, de
https://ocw.ehu.eus/pluginfile.php/50556/mod_page/content/35/Tema2_ocw_2017_con_portada.pdf
- [9] GeeksforGeeks. (2025, 29 agost). Python Tutorial. *GeeksforGeeks*. Recuperat el juliol de 2025, de
<https://www.geeksforgeeks.org/python/python-programming-language-tutorial/>
- [10] Organització Internacional de Normalització. (2020). *Financial services - International bank account number (IBAN). Part 1: Structure of the IBAN (ISO 13616-1:2020)*. Recuperat el juliol de 2025, de
<https://www.iso.org/es/contents/data/standard/08/10/81090.html>
- [11] Quirós, A. [Adolfo]. (2000). *La Teoría de Códigos: una introducción a las Matemáticas de la transmisión de información*. Recuperat el març de 2025, de
https://www.researchgate.net/publication/39141348_Una_breve_introduccion_a_las_matematicas_de_la_transmision_de_informacion_la_teor%C3%ADa_de_codigos
- [12] Romero, J. P., & Romero, J. P. (2010, 9 desembre). ¿Qué significan los números de nuestra tarjeta de crédito? -. - *Matemáticas de Secundaria y Bachillerato*. Recuperat el març de 2025, de
<https://pinae.es/que-significan-los-numeros-de-nuestra-tarjeta-de-credito/>
- [13] Tarazona, C. A. A. (s. d.). Hamming y CRC. *SlideShare*. Recuperat el juliol de 2025, de
<https://es.slideshare.net/carlosamayat/hamming-y-crc>
- [14] Unicaja Banco S.A. (2025). *Cuántos números tiene una tarjeta de crédito*. Recuperat el setembre de 2025, de
<https://www.unicajabanco.es/es/faqs/tarjetas/cuantos-numeros-tiene-tarjeta-credito>
- [15] Urgellés, J. G. (2018). *Matemáticas y códigos secretos*. RBA Libros.

- [16] Vicenç, G. U. J. (2021, 21 octubre). *Cinc cèntims de codificació*. Recuperat l'abril de 2025, de <https://upcommons.upc.edu/handle/2117/354335>
- [17] Villanueva, M. [Mercè] i Fernández-Córdoba, C. [Cristina]. (2019). Codis detectors i correctors d'errors i algunes de les seves aplicacions a la societat de la informació. *Butlletí de la Societat Catalana de Matemàtiques*, 34 (1), 53-89. Recuperat el març de 2025, de <https://revistes.iec.cat/index.php/BSCM/article/view/104070.003>
- [18] Wachira, W. [Wangeçi], Waweru, K. [Kamaku] i Nyaga, L. [Lewis]. (2015). Transposition Error Detection in Luhn's Algorithm. *International Journal of Pure and Applied Sciences and Technology*, 30 (1), 24-28. Recuperat l'agost de 2025, de https://www.academia.edu/19957955/Transposition_Error_Detection_in_Luhn_s_Algorithm
- [19] Widgets — documentació de Guia Tkinter - 0.1.1. (s. d.). Recuperat l'agost de 2025, de <https://guia-tkinter.readthedocs.io/develop/chapters/6-widgets/6.1-Intro.html>

FIGURES

- [20] Contributors to Wikimedia projects. (2025, 18 juny). Targeta de crèdit. *Viquipèdia, L'enciclopèdia lliure*. Recuperat el juliol de 2025, de https://ca.wikipedia.org/wiki/Targeta_de_cr%C3%A8dit
- [21] Wikipedia contributors. (2025, 30 agost). ISBN. *Wikipedia*. Recuperat l'agost de 2025, de <https://en.wikipedia.org/wiki/ISBN>

TAULES

- [22] Ministerio del Interior. (2025). *Cálculo del dígito de control del NIF/NIE*. Recuperat el maig de 2025, de <https://www.interior.gob.es/opencms/es/servicios-al-ciudadano/tramites-y-gestiones/dni/calculo-del-digito-de-control-del-nif-nie/>

6. ANNEXOS

Annex I : Fragment del codi de la funció comprovar en el verificador

```
def comprovar():
    # llegeix el contingut de l'Entry (el número com a cadena)
    numero_targeta = text1.get()

    # validacions: només es poden introduir dígitos i exactament 16
    if not numero_targeta.isdigit():
        # isdigit retorna False si hi ha caràcters no numèrics
        etiqueta_missatge.config(text="Has d'introduir dígitos.", fg="red")
        # si no és vàlid, s'amaga la nota explicativa
        avis_verificador.place_forget()
        return
    if len(numero_targeta) != 16:
        # comprova la longitud exacta de 16 (format típic de targeta)
        etiqueta_missatge.config(text="El número ha de tenir exactament 16 dígitos.", fg="red")
        avis_verificador.place_forget()
        return
    # PAS 1: convertir cada caràcter a int per poder fer operacions
    digits = [int(x) for x in numero_targeta]

    # PAS 2 i 3: multiplicar per 2 els dígitos de posicions senars
    # (1,3,5...) llegint d'esquerra a dreta
    digits_senars = []
    # range(0, len(digits), 2) recorre els índexs 0,2,4,...
    # corresponents a posicions 1,3,5... en comptatge humà
    for idx in range(0, len(digits), 2):
        valor = digits[idx] * 2
        # si el doble és de dues xifres (>=10), es sumen les xifres o
        # es resta 9
        if valor > 9:
            valor = valor - 9
        # afegeix el valor modificat a la llista de senars
        digits_senars.append(valor)

    # PAS 4: recull els dígitos de les posicions parelles
    digits_parells = []
    for idx in range(1, len(digits), 2):
        digits_parells.append(digits[idx])

    # PAS 5: suma totes les parts (tots els dígitos)
    suma_total = sum(digits_senars) + sum(digits_parells)
```

```
# PAS 6: s'aplica mòdul 10. si la suma total és múltiple de 10,  
el número és estructuralment vàlid  
if suma_total % 10 == 0:  
    etiqueta_missatge.config(text="El número de targeta és  
correcte.", fg="green")  
    # si és correcte, es mostra la nota d'avís al peu de pantalla  
    per recordar què significa "correcte"  
    avis_verificador.place(relx=0.5, rely=0.98, anchor='s')  
else:  
    etiqueta_missatge.config(text="El número de targeta NO és  
correcte.", fg="red")  
    # s'amaga la nota si no és vàlid, no té sentit mostrar-la  
    avis_verificador.place_forget()
```

Annex II : Fragment del codi de la funció comprovar en la calculadora

```

def comprovar():
    # reinicia l'índex quan s'esborra el contingut de les caixes
    nonlocal index_actual
    valors = [] # llista temporal amb el contingut de cada caixa
    pos_interrogant = -1 # posició del '?', o -1 si no n'hi ha

    # es recorren les 16 caixes i es recullen els valors
    for i, caixa in enumerate(caixes):
        valor = caixa.get()
        if valor == "?":
            # si ja s'ha trobat un '?', és un error: només pot haver-n'hi 1
            if pos_interrogant != -1:
                etiqueta_resultat.config(text="Només es pot posar un únic '?'")
                return
            pos_interrogant = i
            valors.append("?")
        elif valor.isdigit():
            # es converteixen els valors a enters per poder operar després
            valors.append(int(valor))
        else:
            # qualsevol altre cas (caràcter no numèric i no '?') és invàlid
            etiqueta_resultat.config(text="Hi ha valors no vàlids.")
            return

    # si no hi ha '?', no té sentit calcular res
    if pos_interrogant == -1:
        etiqueta_resultat.config(text="No s'ha introduït cap '?'")
        return

    # es prova cada possible dígit de 0 a 9 per substituir el '?'
    digit_correccte = None
    for x in range(10):
        total = 0
        # es calcula la suma segons l'algoritme de Luhn per al número amb
        # x que substitueix '?'
        for i in range(16):
            # si la posició actual era la del '?', es fa servir x; si no,
            # es fa servir el valor ja recollit
            valor = x if valors[i] == "?" else valors[i]
            # es llegeix d'esquerra a dreta; posicions 1,3,5,... (índex
            # 0,2,4,...) es doblen (x2). posicions parelles (2,4,6,...), no
            # es modifiquen (x1)
            if (i + 1) % 2 == 0:
                total += valor

```

```
        else:
            # posicions senars: doble i si el doble >9 es resta 9
            doble = valor * 2
            if doble > 9:
                doble -= 9
            total += doble
            # s'aplica mòdul 10. si la suma és múltiple de 10, s'ha trobat el
            dígit que fa vàlid el número
            if total % 10 == 0:
                digit_correcte = x
                break

    if digit_correcte is not None:
        # es construeix la representació amb '?' exacta com l'ha introduït
        l'usuari
        numero_amb_interrogant = ""
        for v in valors:
            numero_amb_interrogant += str(v)
        # es construeix el número complet substituint '?' pel dígit trobat
        numero_complet = ""
        for v in valors:
            if v == "?":
                numero_complet += str(digit_correcte)
            else:
                numero_complet += str(v)
        # es mostra el missatge amb el número amb '?' i el dígit trobat i
        el número complet
        etiqueta_resultat.config(text=f"El dígit que has intentat calcular
        al número {numero_amb_interrogant} és: {digit_correcte}.\nNúmero
        complet: {numero_complet}")
        # s'esborra el contingut de totes les caixes per preparar una nova
        entrada i es reinicia l'índex
        for c in caixes:
            c.delete(0, END)
        index_actual = 0
        # es posa el focus a la primera caixa
        caixes[0].focus()
    else:
        # si cap x fa que la suma sigui múltiple de 10, no hi ha solució
        entre 0-9 que faci vàlid el número
        etiqueta_resultat.config(text="Cap valor de 0 a 9 compleix la
        condició.")
```

Annex III : Codi del programa

```
from tkinter import *

# - Funció per canviar de pàgina (mostrar només un frame) -
def mostrar_frame(frame_a_mostrar):
    for f in (frame_inici, frame_verificador, frame_calculador):
        f.pack_forget()
    frame_a_mostrar.pack(expand=True, fill="both")

    if 'boto_inici' in globals():
        if frame_a_mostrar is frame_inici:
            boto_inici.place_forget()
        else:
            boto_inici.place(relx=0, rely=0, anchor='nw', x=10, y=10)

# FINESTRA PRINCIPAL
finestra = Tk()
finestra.title("Programa Part Pràctica TdR - Berta Ventura Güell")
finestra.state('zoomed')

# FRAME: PANTALLA D'INICI
frame_inici = Frame(finestra)

# - Contenedor central per centrar els botons -
contenedor_inici = Frame(frame_inici)
contenedor_inici.place(relx=0.5, rely=0.5, anchor='center')

# - Funcions -
def anar_a_verificar():
    mostrar_frame(frame_verificador)

def anar_a_calcular():
    mostrar_frame(frame_calculador)

# - Botons -
boto_verificar= Button(contenedor_inici, text="VERIFICAR EL NÚMERO D'UNA  
TARGETA DE CRÈDIT", font=("Calibri 14"), width=50, height=2,  
command=anar_a_verificar, bg="#b6d7a8")

boto_calcular = Button(contenedor_inici, text="CALCULAR UN DÍGIT D'UN  
NÚMERO DE TARGETA DE CRÈDIT", font=("Calibri 14"), width=50, height=2,  
command=anar_a_calcular, bg="#f4b6c2")

# - Posició botons -
```

```

boto_verificar.grid(row=0, column=0, padx=20, pady=10)
boto_calcular.grid(row=1, column=0, padx=20, pady=10)

# - Botó sortir -
boto_sortir = Button(frame_inici, text="✖ Sortir", font=("Calibri 12"),
width=10, height=1, bg="indian red", command=finestra.destroy)
boto_sortir.place(relx=1, rely=0, anchor='ne', x=-10, y=10)

# FRAME: VERIFICADOR
frame_verificador = Frame(finestra)

def construir_verificador():
    if getattr(frame_verificador, "_construït", False):
        return
    frame_verificador._construït = True

i = 0

# Títol
boto_titol_ver = Button(frame_verificador, text="VERIFICAR EL NÚMERO
D'UNA TARGETA DE CRÈDIT", font=("Calibri 14"), width=40, height=1,
bg="#b6d7a8", state=DISABLED, disabledforeground="black",
relief="flat")
boto_titol_ver.place(relx=0.5, rely=0.08, anchor='center')

# Text explicatiu sota el títol
text_exp_ver= Label(frame_verificador, text="Introdueix els 16 dígits
del número de targeta de crèdit i comprova si aquest és vàlid.",
font=("Calibri 12"), justify="center")
text_exp_ver.place(relx=0.5, rely=0.14, anchor='center')

contenedor = Frame(frame_verificador)
contenedor.place(relx=0.5, rely=0.5, anchor='center')

text1 = Entry(contenedor, font=("Calibri 12"), justify="center")
text1.grid(row=0, column=0, columnspan=5, padx=5, pady=10)

etiqueta_missatge = Message(contenedor, text="", font=("Calibri 12"),
width=170, justify="center")
etiqueta_missatge.grid(row=1, column=0, columnspan=5, rowspan=2,
padx=5, pady=8)

# Nota d'avís al peu
avis_verificador = Label(frame_verificador, text="⚠ Que sigui
'correcte' vol dir que compleix l'algoritme de Luhn.\n" "Això no

```

```
implica que la targeta existeixi realment.", font=("Calibri 10"),
fg="gray40", justify="center")

# - Funcions -
def bloquejar_teclat(event):
    return "break"

text1.bind("<Key>", bloquejar_teclat)

def click_boto(valor):
    nonlocal i
    if i < 16:
        text1.insert(i, str(valor))
        i += 1

def esborrar():
    nonlocal i
    if i > 0:
        text1.delete(i - 1)
        i -= 1

def resset():
    nonlocal i
    text1.delete(0, END)
    i = 0
    etiqueta_missatge.config(text="")
    avis_verificador.place_forget()

def comprovar():
    numero_targeta = text1.get()

    if not numero_targeta.isdigit():
        etiqueta_missatge.config(text="Només pots introduir dígit.",
fg="red")
        avis_verificador.place_forget()
        return

    if len(numero_targeta) != 16:
        etiqueta_missatge.config(text="El número ha de tenir exactament
16 dígit.", fg="red")
        avis_verificador.place_forget()
        return

    digits = [int(x) for x in numero_targeta]

    digits_senars = []
    for idx in range(0, len(digits), 2):
```

```

        valor = digits[idx] *
        if valor > 9:
            valor = valor - 9
        digits_senars.append(valor)

    digits_parells = []
    for idx in range(1, len(digits), 2):
        digits_parells.append(digits[idx])

    suma_total = sum(digits_senars) + sum(digits_parells)

    if suma_total % 10 == 0:
        etiqueta_missatge.config(text="El número de targeta és
correcte.", fg="green")
        avis_verificador.place(relx=0.5, rely=0.98, anchor='s')
    else:
        etiqueta_missatge.config(text="El número de targeta NO és
correcte.", fg="red")
        avis_verificador.place_forget()

# - Botons -
boto1 = Button(contenedor, text="1", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(1), bg="grey86")
boto2 = Button(contenedor, text="2", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(2), bg="grey86")
boto3 = Button(contenedor, text="3", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(3), bg="grey86")
boto4 = Button(contenedor, text="4", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(4), bg="grey86")
boto5 = Button(contenedor, text="5", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(5), bg="grey86")
boto6 = Button(contenedor, text="6", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(6), bg="grey86")
boto7 = Button(contenedor, text="7", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(7), bg="grey86")
boto8 = Button(contenedor, text="8", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(8), bg="grey86")
boto9 = Button(contenedor, text="9", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(9), bg="grey86")
boto0 = Button(contenedor, text="0", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(0), bg="grey86")

boto_esborrar = Button(contenedor, text="✕", font=("Calibri 12"),
width=5, height=2, command=lambda: esborrar(), bg="azure3")
boto_ac = Button(contenedor, text="AC", font=("Calibri 12"), width=5,
height=2, command=lambda: resset(), bg="rosy brown")

```

```

boto_calcular = Button(contenedor, text="COMPROVAR", font=("Calibri
12"), width=20, height=1, command=lambda: comprovar(), bg="yellow3")

# - Posició botons -
boto1.grid(row=3, column=1, padx=1, pady=1)
boto2.grid(row=3, column=2, padx=1, pady=1)
boto3.grid(row=3, column=3, padx=1, pady=1)

boto4.grid(row=4, column=1, padx=5, pady=5)
boto5.grid(row=4, column=2, padx=5, pady=5)
boto6.grid(row=4, column=3, padx=5, pady=5)

boto7.grid(row=5, column=1, padx=5, pady=5)
boto8.grid(row=5, column=2, padx=5, pady=5)
boto9.grid(row=5, column=3, padx=5, pady=5)

boto0.grid(row=6, column=2, padx=5, pady=5)
boto_esborrar.grid(row=6, column=1, padx=5, pady=5)
boto_ac.grid(row=6, column=3, padx=5, pady=5)

boto_calcular.grid(row=7, column=1, columnspan=3, padx=5, pady=8)

# Construir interfície verificador
construir_verificador()

# FRAME: CALCULADORA DE DÍGITS.

frame_calculador = Frame(finestra)

def construir_calculadora():
    if getattr(frame_calculador, "_construït", False):
        return
    frame_calculador._construït = True

# Títol
boto_titol_calc = Button(frame_calculador, text="CALCULAR UN DÍGIT
D'UN NÚMERO DE TARGETA DE CRÈDIT", font=("Calibri 14"), width=50,
height=1, bg="#f4b6c2", state=DISABLED, disabledforeground="black",
relief="flat")
boto_titol_calc.place(relx=0.5, rely=0.08, anchor='center')

# Text explicatiu sota el títol
text_exp_calc = Label(frame_calculador, text="Introdueix el número de
targeta amb un sol '?' on falta el dígit que es vol calcular.\n" "Es
calcularà el dígit que falta perquè el número de targeta sigui

```

```

vàlid.", font=("Calibri 12"), justify="center")
text_exp_calc.place(relx=0.5, rely=0.15, anchor='center')

contenedor = Frame(frame_calculador)
contenedor.place(relx=0.5, rely=0.5, anchor="center")

caixes = []
index_actual = 0

# - Funcions -
def bloquejar_teclat(event):
    return "break"

for i in range(16):
    entrada = Entry(contenedor, width=5, font=("Calibri 14"),
                    justify="center", bg="white")
    entrada.grid(row=0, column=i, padx=2, pady=10, ipady=12)
    entrada.bind("<Key>", bloquejar_teclat)
    caixes.append(entrada)

def click_boto(valor):
    nonlocal index_actual
    if index_actual < len(caixes):
        if valor == "?":
            if any(caixa.get() == "?" for caixa in caixes):
                etiqueta_resultat.config(text="Només es pot posar un únic '?'")
                return
            if caixes[index_actual].get() == "":
                caixes[index_actual].insert(0, str(valor))
                index_actual += 1
                if index_actual < len(caixes):
                    caixes[index_actual].focus()
                etiqueta_resultat.config(text="")

def esborrar():
    nonlocal index_actual
    if index_actual > 0:
        index_actual -= 1
        caixes[index_actual].delete(0, END)
        caixes[index_actual].focus()

# Etiqueta de resultat
etiqueta_resultat = Label(contenedor, text="", font=("Calibri 12"),
                           fg="blue")
etiqueta_resultat.grid(row=5, column=0, columnspan=16, pady=(5,0))

```

```
def comprovar():
    nonlocal index_actual
    valors = []
    pos_interrogant = -1

    for i, caixa in enumerate(caixes):
        valor = caixa.get()
        if valor == "?":
            if pos_interrogant != -1:
                etiqueta_resultat.config(text="Només es pot posar un  
únic '?'")
                return
            pos_interrogant = i
            valors.append("?")
        elif valor.isdigit():
            valors.append(int(valor))
        else:
            etiqueta_resultat.config(text="Hi ha valors no vàlids.")
            return

    if pos_interrogant == -1:
        etiqueta_resultat.config(text="No s'ha introduït cap '?'")
        return

    digit_correcte = None
    for x in range(10):
        total = 0
        for i in range(16):
            valor = x if valors[i] == "?" else valors[i]
            if (i + 1) % 2 == 0:
                total += valor
            else:
                doble = valor * 2
                if doble > 9:
                    doble -= 9
                total += doble
        if total % 10 == 0:
            digit_correcte = x
            break

    if digit_correcte is not None:
        numero_amb_interrogant = ""
        for v in valors:
            numero_amb_interrogant += str(v)
```

```

numero_complet = ""
for v in valors:
    if v == "?":
        numero_complet += str(digit_correcte)
    else:
        numero_complet += str(v)

etiqueta_resultat.config(text=f"El dígit que has intentat
calcular al número {numero_amb_interrogant} és:
{digit_correcte}.\nNúmero complet: {numero_complet}")

for c in caixes:
    c.delete(0, END)
index_actual = 0
caixes[0].focus()

else:
    etiqueta_resultat.config(text="Cap valor de 0 a 9 compleix la
condició.")

caixes[0].focus()

# - Botons -
boto1 = Button(contenedor, text="1", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(1), bg="grey86")
boto2 = Button(contenedor, text="2", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(2), bg="grey86")
boto3 = Button(contenedor, text="3", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(3), bg="grey86")
boto4 = Button(contenedor, text="4", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(4), bg="grey86")
boto5 = Button(contenedor, text="5", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(5), bg="grey86")
boto6 = Button(contenedor, text="6", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(6), bg="grey86")
boto7 = Button(contenedor, text="7", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(7), bg="grey86")
boto8 = Button(contenedor, text="8", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(8), bg="grey86")
boto9 = Button(contenedor, text="9", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(9), bg="grey86")
boto0 = Button(contenedor, text="0", font=("Calibri 12"), width=5,
height=2, command=lambda: click_boto(0), bg="grey86")
boto_esborrar = Button(contenedor, text="✕", font=("Calibri 12"),
width=5, height=2, command=lambda: esborrar(), bg="azure3")
boto_blanc = Button(contenedor, text="?", font=("Calibri 12"),

```

```
width=5, height=2, command=lambda: click_boto("?"), bg="grey86")
boto_calcular = Button(contenedor, text="CALCULAR", font=("Calibri
12"), width=44, height=1, command=lambda: comprovar(), bg="yellow3")
```

- Posició botons -

```
boto0.grid(row=2, column=5, padx=2, pady=2)
boto1.grid(row=2, column=6, padx=2, pady=2)
boto2.grid(row=2, column=7, padx=2, pady=2)
boto3.grid(row=2, column=8, padx=2, pady=2)
boto4.grid(row=2, column=9, padx=2, pady=2)
boto5.grid(row=2, column=10, padx=2, pady=2)
```

```
boto6.grid(row=3, column=5, padx=2, pady=2)
boto7.grid(row=3, column=6, padx=2, pady=2)
boto8.grid(row=3, column=7, padx=2, pady=2)
boto9.grid(row=3, column=8, padx=2, pady=2)
boto_blanc.grid(row=3, column=9, padx=2, pady=2)
boto_esborrar.grid(row=3, column=10, padx=2, pady=2)
```

```
boto_calcular.grid(row=4, column=5, columnspan=6, padx=5, pady=8)
```

Nota d'avís

```
nota_calc = Label(frame_calculador, text="⚠ El dígit calculat fa
que el número de targeta complert sigui vàlid segons l'algoritme de
Luhn.\n" "Això no garanteix que sigui el número d'una targeta
existent.", font=("Calibri 10"), fg="gray40", justify="center")
nota_calc.place(relx=0.5, rely=0.98, anchor='s')
```

Construir la interfície de la calculadora

```
construir_calculadora()
```

Botó d'inici

```
boto_inici = Button(finestra, text="🏠 Inici", font=("Calibri 12"),
width=7, height=1, bg="lightblue", command=lambda:
mostrar_frame(frame_inici))
```

Mostrar d'entrada la pantalla d'inici

```
mostrar_frame(frame_inici)
finestra.mainloop()
```

Annex IV : Captures del funcionament de la pantalla del verificador

Programa Part Pràctica TDR - Berta Ventura Güell

VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT

Introdueix els 16 dígit del número de targeta de crèdit i comprova si aquest és vàlid.

86570

El número ha de tenir exactament 16 dígit.

1	2	3
4	5	6
7	8	9
<	0	AC

COMPROVAR

Figura 9. Captura de la pantalla del verificador cas 1. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un nombre incomplet, de menys de 16 xifres.

Programa Part Pràctica TDR - Berta Ventura Güell

VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT

Introdueix els 16 dígit del número de targeta de crèdit i comprova si aquest és vàlid.

5796428097536978

El número de targeta NO és correcte.

1	2	3
4	5	6
7	8	9
<	0	AC

COMPROVAR

Figura 10. Captura de la pantalla del verificador cas 2. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un nombre de targeta inventat, incorrecte.

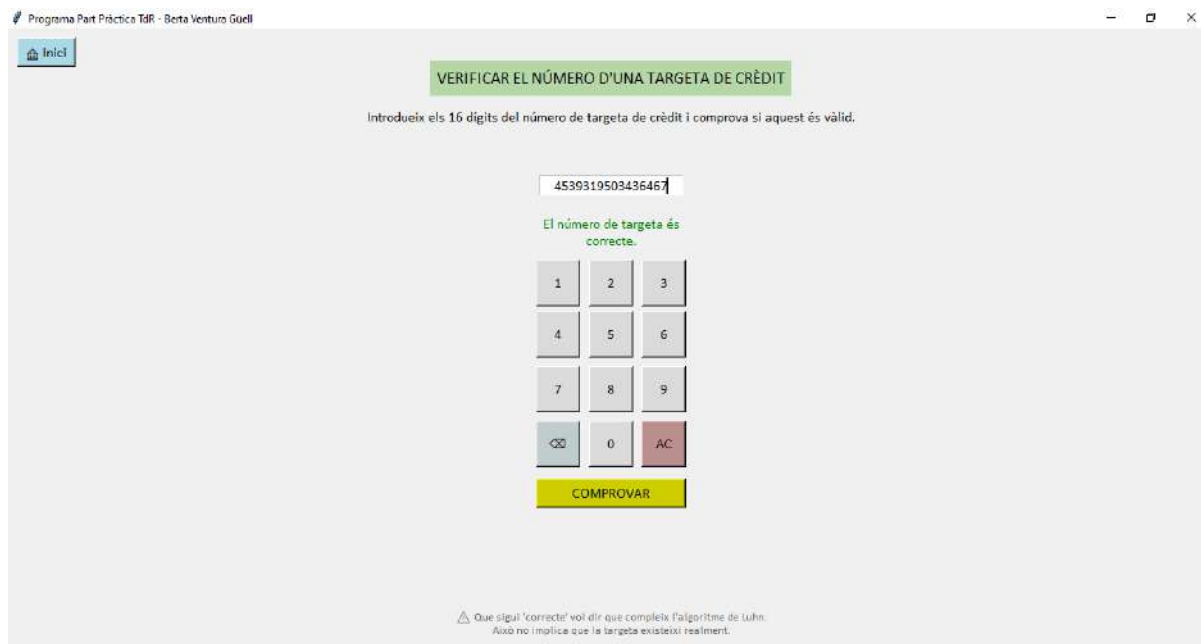


Figura 11. Captura de la pantalla del verificador cas 3. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un nombre de targeta real correcte.

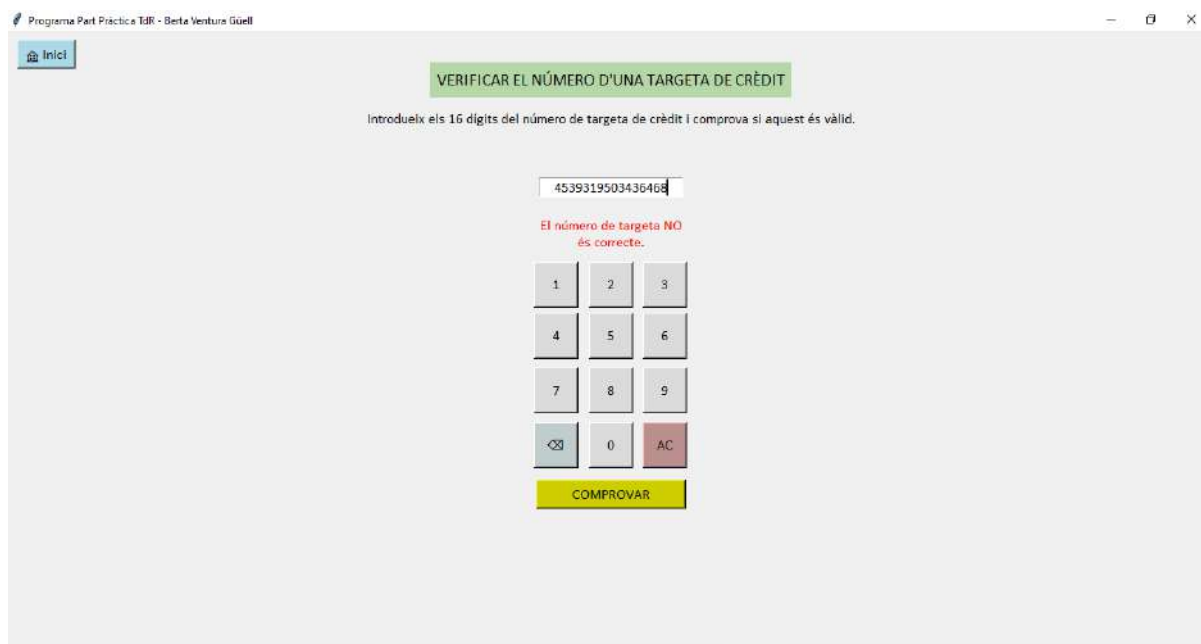


Figura 12. Captura de la pantalla del verificador cas 4. (Elaboració pròpia, 2025)

Nota: quan s'introdueix el nombre de targeta correcte, però amb la substitució d'un sol dígit; i aquest esdevé incorrecte.

Programa Part Práctica TdR - Berta Ventura Güell

VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT

Introdueix els 16 dígit del número de targeta de crèdit i comprova si aquest és vàlid.

4530319593436467

El número de targeta és correcte.

1	2	3
4	5	6
7	8	9
<>	0	AC

COMPROVAR

⚠ Que sigui 'correcte' vol dir que compleix l'algorisme de Luhn. Això no implica que la targeta existeixi realment.

Figura 13. Captura de la pantalla del verificador cas 5. (Elaboració pròpia, 2025)

Nota: quan s'introdueix el nombre de targeta correcte, però amb un intercanvi de posició dels dígit 0 i 9, un en posició senar i un altre en posició parella (consecutius o no); el número continua essent correcte perquè, tot i l'error, aquest compleix l'algorisme de Luhn.

Programa Part Práctica TdR - Berta Ventura Güell

VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT

Introdueix els 16 dígit del número de targeta de crèdit i comprova si aquest és vàlid.

4530319503436476

El número de targeta NO és correcte.

1	2	3
4	5	6
7	8	9
<>	0	AC

COMPROVAR

Figura 14. Captura de la pantalla del verificador cas 6. (Elaboració pròpia, 2025)

Nota: quan s'introdueix el nombre de targeta correcte, però amb un intercanvi de posició de dos dígitos diferents a 0 i 9, un en posició senar i l'altre en posició parella (consecutius o no); el número esdevé incorrecte.

Programa Part Práctica TdR - Berta Ventura Güell

INICI

VERIFICAR EL NÚMERO D'UNA TARGETA DE CRÈDIT

Introdueix els 16 dígit del número de targeta de crèdit i comprova si aquest és vàlid.

4539319503436764

El número de targeta és correcte.

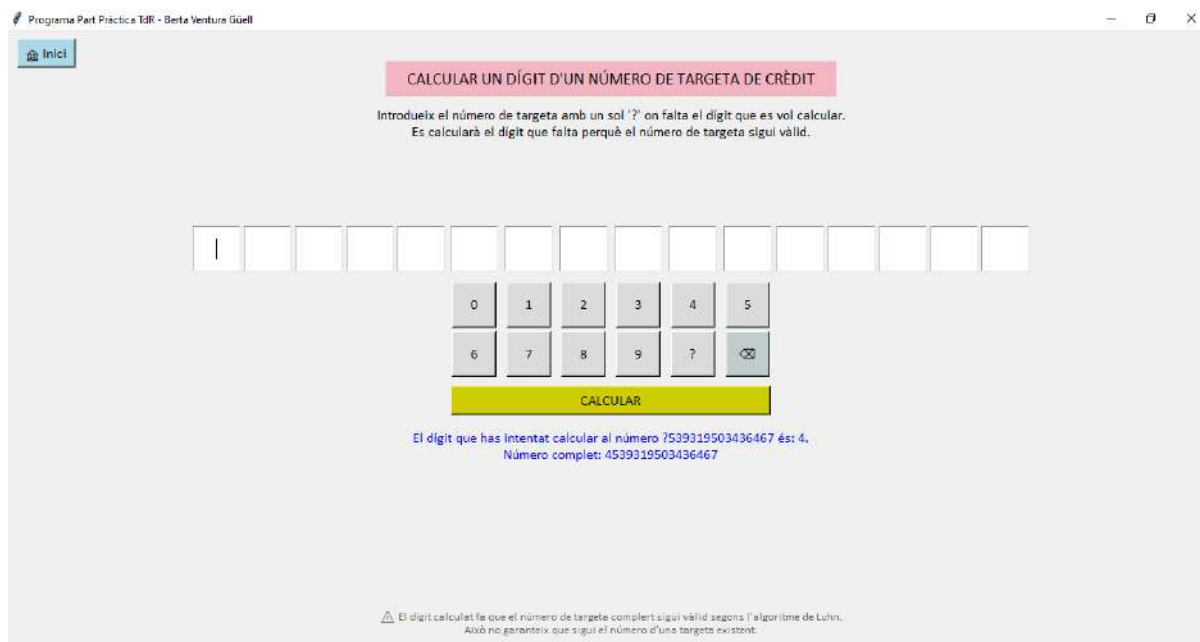
1	2	3
4	5	6
7	8	9
✖	0	AC

COMPROVAR

⚠ Que sigui 'correcte' vol dir que compleix l'algorisme de Luhn. Això no implica que la targeta existeixi realment.

Figura 15. Captura de la pantalla del verificador cas 7. (Elaboració pròpia, 2025)

Nota: quan s'introdueix el nombre de targeta correcte, però amb un intercanvi de dos dígit en posicions parells o senars (les dues); el número continua essent vàlid perquè, tot i l'error, aquest compleix l'algorisme de Luhn.

Annex V : Captures del funcionament de la pantalla de la calculadora**Figura 16.** Captura de la pantalla de la calculadora de dígit cas 1. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el primer dígit, el qual el calcula correctament.

**Figura 17.** Captura de la pantalla de la calculadora de dígit cas 2. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el segon dígit, el qual el calcula correctament.

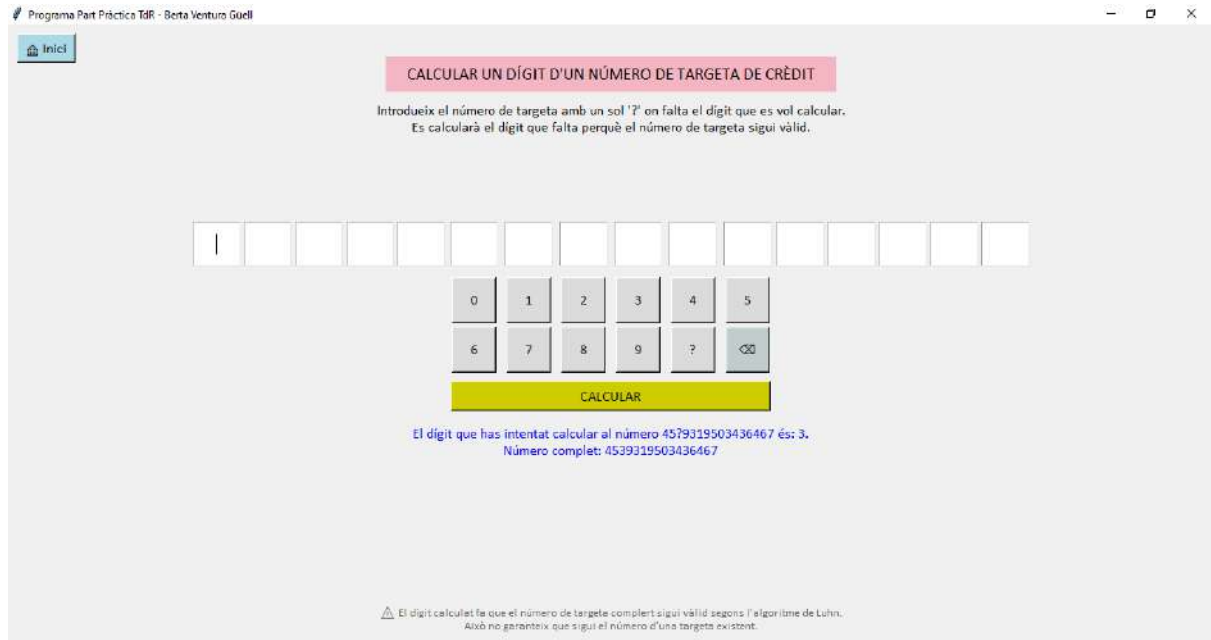


Figura 18. Captura de la pantalla de la calculadora de dígit cas 3. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el tercer dígit, el qual el calcula correctament.



Figura 19. Captura de la pantalla de la calculadora de dígit cas 4. (Elaboració pròpia, 2025) **Nota:** quan s'introdueix un número de targeta correcte i s'intenta calcular el quart dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

Input field: | | | | | | | | | | | | | | | |

Buttons: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ?,

CALCULAR

El dígit que has intentat calcular al número 4539719503436467 és: 8.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algoritme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 20. Captura de la pantalla de la calculadora de dígit cas 5. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el cinquè dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

Input field: | | | | | | | | | | | | | | | |

Buttons: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ?,

CALCULAR

El dígit que has intentat calcular al número 4539379503436467 és: 1.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algoritme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 21. Captura de la pantalla de la calculadora de dígit cas 6. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el sisè dígit, el qual el calcula correctament.

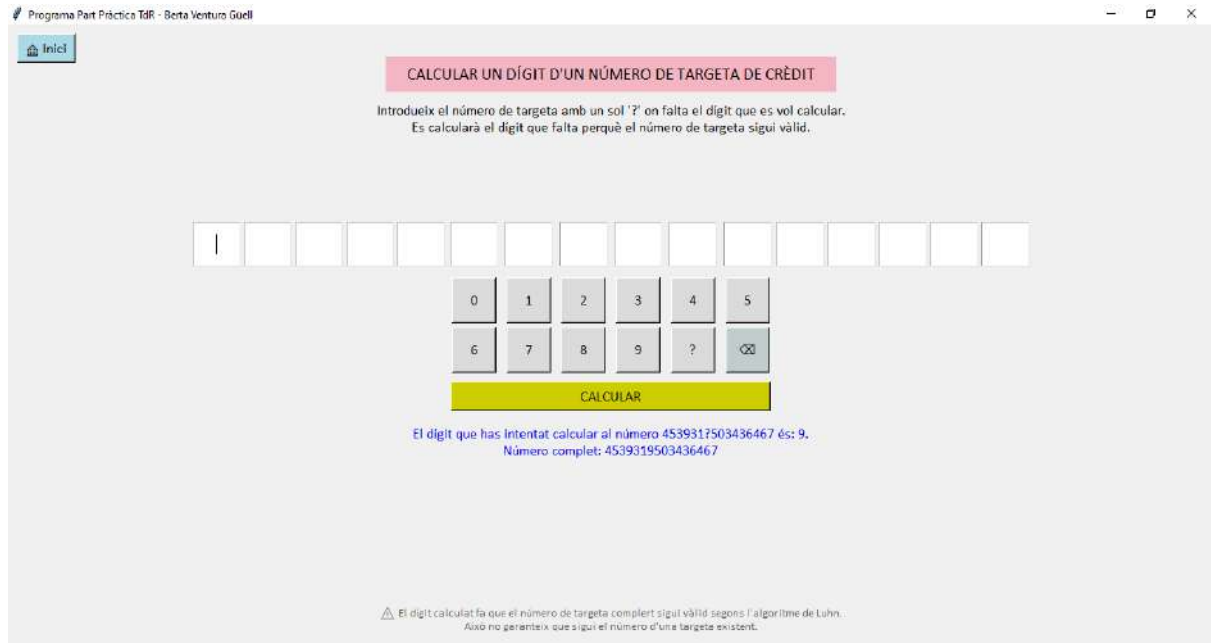


Figura 22. Captura de la pantalla de la calculadora de dígit cas 7. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el setè dígit, el qual el calcula correctament.



Figura 23. Captura de la pantalla de la calculadora de dígit cas 8. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el vuitè dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

INICI

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 45393195?3436467 és: 0.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algoritme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 24. Captura de la pantalla de la calculadora de díigits cas 9. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el novè dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

INICI

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 453931950?436467 és: 3.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algoritme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 25. Captura de la pantalla de la calculadora de díigits cas 10. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el desè dígit, el qual el calcula correctament.

Programa Pràctica TdR - Berta Ventura Güell

INICI

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 4539319503?36467 és: 4.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 26. Captura de la pantalla de la calculadora de dígit cas 11. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular l'onzè dígit, el qual el calcula correctament.

Programa Pràctica TdR - Berta Ventura Güell

INICI

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 45393195034?6467 és: 3.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 27. Captura de la pantalla de la calculadora de dígit cas 12. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el dotzè dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 4539319503437467 és: 6.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 28. Captura de la pantalla de la calculadora de dígets cas 13. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el tretzè dígit, el qual el calcula correctament.

Programa Part Práctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 4539319503436767 és: 4.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 29. Captura de la pantalla de la calculadora de dígets cas 14. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el catorzè dígit, el qual el calcula correctament.

Programa Pràctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 4539319503436477 és: 6.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 30. Captura de la pantalla de la calculadora de dígit cas 15. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular el quinçè dígit, el qual el calcula correctament.

Programa Pràctica TdR - Berta Ventura Güell

CALCULAR UN DÍGIT D'UN NÚMERO DE TARGETA DE CRÈDIT

Introdueix el número de targeta amb un sol '?' on falta el dígit que es vol calcular.
Es calcularà el dígit que falta perquè el número de targeta sigui vàlid.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

0 1 2 3 4 5
6 7 8 9 ?

CALCULAR

El dígit que has intentat calcular al número 4539319503436467 és: 7.
Número complet: 4539319503436467

⚠ El dígit calculat fa que el número de targeta complert sigui vàlid segons l'algorisme de Luhn.
Això no garanteix que sigui el número d'una targeta existent.

Figura 31. Captura de la pantalla de la calculadora de dígit cas 16. (Elaboració pròpia, 2025)

Nota: quan s'introdueix un número de targeta correcte i s'intenta calcular l'últim dígit (dígit de control), el qual el calcula correctament.

Figura 32. Captura de la pantalla de la calculadora de dígit cas 17. (Elaboració pròpia, 2025)

Nota: quan s'intenta introduir més d'un '?'. Es mostra el missatge avisant de l'únic ús d'aquest.

Figura 33. Captura de la pantalla de la calculadora de dígit cas 18. (Elaboració pròpia, 2025)

Nota: quan no s'ha introduït cap '?' i es mostra el missatge avisant-ho.

Annex VI : Entrada de l'algoritme de Luhn a la Viquipèdia**AMPLIACIÓ**

[explicació original]

Pel que fa a la detecció d'errors, l'algoritme de Luhn permet detectar de manera molt eficaç un ampli ventall d'errors, però no tots. Detecta de manera completa qualsevol substitució d'un sol dígit, gràcies a l'aplicació de l'aritmètica modular, que assegura que qualsevol canvi alteri el residu final. També és capaç d'identificar gairebé totes les transposicions de dígit consecutius, amb l'única excepció d'algunes combinacions molt concretes (com el canvi de 09 a 90 o viceversa). En canvi, les transposicions de dígit situats en posicions d'igual paritat (és a dir, les dues posicions parelles o les dues senars) no són detectades, ja que els càlculs aplicats a aquests dígit no generen variacions en la suma final.

Exemple pràctic: com es valida una targeta?

El número imprès a la part frontal d'una targeta de crèdit o debit no és aleatori. Aquest número, anomenat PAN (Primary Account Number), permet identificar l'entitat emissora i el titular del compte, i incorpora un dígit de control que es calcula mitjançant l'algoritme de Luhn, especialment dissenyat per detectar errors de transcripció habituals.

L'algoritme s'aplica mitjançant una sèrie de passos molt concrets. Per entendre millor com funciona, s'explica tot seguit de forma detallada fent servir un exemple: la targeta amb número 4539 3195 0343 6467. Cal tenir present que l'últim dígit del número, que en aquest cas és el 7, és el dígit de control i serveix per verificar si tota la resta del número és correcta.

Així doncs, es posa per exemple la validació d'un número de targeta.

Pas 1: Primer de tot, es pren el número de la targeta i es llegeix d'esquerra a dreta. D'aquest, s'agafaran els dígit de les posicions senars, és a dir, un dígit sí i un no, que són els que es multiplicaran per dos en el següent pas. En aquest cas el número seria aquest: 4539 3195 0343 6467; per tant, els dígit que cal agafar són els següents: 4, 3, 3, 9, 0, 4, 6, 6.

Pas 2: Seguidament, s'agafen els dígit que s'han pres anteriorment, els de les posicions senars, i es multipliquen per dos. Com a resultat, queda: 8, 6, 6, 18, 0, 8, 12, 12.

Pas 3: Ara, s'agafen els resultats de les multiplicacions realitzades en el pas anterior: 8, 6, 6, 18, 0, 8, 12, 12. Si algun resultat té dues xifres (és més gran que 9), se sumen les xifres del resultat per reduir-lo a una sola xifra, o bé se li'n resta 9. Per tant, els dígit modificats són: 8, 6, 6, 9, 0, 8, 3, 3.

Pas 4: A continuació, cal agafar els dígit que no s'han tocat, és a dir, els dígit que es troben a les posicions parelles del número inicial (inclòs el dígit de control): 4539 3195 0343 6467 i s'han de sumar juntament amb els dígit modificats del pas anterior.

Pas 5: Finalment cal aplicar l'aritmètica modular amb mòdul 10. El número de targeta serà vàlid si la suma total feta anteriorment en mòdul 10 dona residu 0, és a dir, quan $x \equiv 0 \pmod{10}$, essent x la suma total. Dit d'una altra manera, per poder confirmar la validesa del número de targeta, cal que la suma total després de seguir tots els passos anteriors, sigui múltiple de 10.

[S'ha creat una taula de l'exemple de validació d'un número de targeta amb l'aplicació de l'algoritme de Luhn.]

En aquest cas $80 \equiv 0 \pmod{10}$, cosa que significa que 80 és divisible per 10, per això el residu dona 0. Per tant, el número 4539 3195 0343 6467 és vàlid segons l'algoritme de Luhn.

Ara bé, si el dígit de control fos diferent, per exemple un 8 en lloc d'un 7, la suma total seria 81, i no seria vàlida. Només quan la suma total és exactament un múltiple de 10 (com per exemple 70, 80 o 90), el número de la targeta es considera vàlid estructuralment. Aquest procés matemàtic permet detectar errors immediatament després que el número sigui introduït. Aquesta comprovació es fa automàticament cada vegada que s'usa una targeta, ja sigui per fer una compra, una transferència o una operació bancària. És per això que l'algoritme de Luhn continua sent una eina fonamental en la seguretat i verificació del sistema financer modern.

ENTRADA A LA VIQUIPÈDIA

Per tal de poder visualitzar aquest text directament a l'entrada a la Viquipèdia s'ha d'accedir al següent enllaç:

https://ca.wikipedia.org/wiki/Algoritme_de_Luhn#Exemple_pr%C3%A0ctic:_com_es_valida_una_targeta