Desenvolupament d'interfícies

Unitat 9. Usabilitat, pautes de disseny i accessibilitat. Proves i sistemes de proves.

Índex

1. Usabilitat

En plena era digital, el nombre de continguts proporcionats a través d'aplicacions web que es mostren als usuaris a través d'una interfície, s'ha vist augmentat de manera exponencial.

A més, l'usuari cada vegada compta amb més formació relativa a l'ús de les noves tecnologies. És per això que les aplicacions més recents tenen una aparença i unes funcionalitats que costava pensar fa a penes uns anys.

"Si no ho fas fàcil, els usuaris marxaran del teu web", *Jakob Nielsen*.

Per això, per a l'èxit d'una aplicació o aplicació web, és cada vegada més important proporcionar un accés ràpid als continguts i proporcionar una experiència visual agradable a l'usuari, que el convide a tornar en un futur per a obtindre la informació que necessita. La consecució de les eines d'una aplicació dependrà en gran manera de la satisfacció que es proporcione a l'usuari. Aquesta satisfacció dependrà, al seu torn, d'una sèrie de paràmetres que guarden relació amb la claredat i utilitat dels continguts, amb la qualitat d'aquests, amb un disseny atractiu, etc. Per tant, és fonamental un disseny adequat de la web per a facilitar l'intercanvi d'informació amb l'usuari.

1.1. **Objectius d'ús i estàndards d'usabilitat**

D'acord amb Nilsen, la usabilitat és un concepte relacionat intrínsecament amb la forma en la qual una interfície és presentada a l'usuari, així com la forma en la qual l'usuari la utilitza, tenint en compte alguns paràmetres com la seua senzillesa, claredat, etc.

L'Organització Internacional de Normalització (International Organization for Standardization, ISO), s'encarrega de la creació de normes i estàndards l'objectiu principal dels quals és aconseguir assegurar que serveis i productes presenten uns certs nivells de qualitat, eficiència i seguretat. Segons l'ISO, la usabilitat fa esment a la capacitat d'un programari determinat per a ser comprés, utilitzat i aprés per l'usuari, al mateix temps que li resulta atractiu.

Una aplicació no sols ha de tindre un aspecte atractiu i ser tecnològicament capdavantera, sinó que a més ha de ser fàcilment comprensible per qualsevol usuari, amb l'objectiu de proporcionar la informació que aquest cerca en el menor temps. Per això, es pot afirmar que la usabilitat depén del producte, però també depén de l'usuari, quant a com interactua amb la interfície de l'aplicació en concret i com l'aprecia quant a senzillesa i facilitat d'utilització.

Això es posa de manifest a través de múltiples fets que ocorren diàriament en la relació usuari-interfície. Per exemple, quan per la causa que fora l'experiència de navegació a través d'un determinat portal no és agradable, o bé la informació que proporciona no és clara o útil, és molt probable que l'usuari abandone l'aplicació i no torne en un futur.

Sobre la base de l'anterior, podem afirmar que existeixen paràmetres subjectius (satisfacció d'usuari) i objectius (temps emprat per l'usuari per a aconseguir el seu objectiu, errors

comesos per a aconseguir el que es busca, etc.) per a poder mesurar la usabilitat d'un determinat lloc.

Existeixen diversos estàndards i normes relacionats directament amb la usabilitat i amb l'accessibilitat, que defineixen diferents aspectes relatius a aquesta. S'intenta aconseguir així una uniformitat en els criteris de disseny, ja que no seria lògic que cada dissenyador d'interfícies triarà uns paràmetres d'usabilitat diferents:

- **ISO / IEC 9126**. Es tracta d'un estàndard internacional per a l'avaluació de la qualitat del programari, es presenta dividit en quatre parts: model de la qualitat, mètriques externes, mètriques internes i mètriques de qualitat en ús.
- **ISO / DIS 9241-11**. Es tracta d'una norma que recull els benefici que aporta la mesura de la usabilitat en termes de resultats i satisfacció obtinguts per l'usuari. Aquests beneficis es mesuren pel grau de consecució dels objectius previstos quant a utilització, pels recursos emprats per a aconseguir aquests objectius i pel grau d'acceptació del producte per part de l'usuari.
- **ISO 13407.** L'ISO 13407 proporciona una guia per a aconseguir la qualitat en l'ús mitjançant la incorporació d'activitats de naturalesa iterativa involucrades en el disseny centrat en l'usuari (DCU).
- **ISO 9241 / 151.** Ergonomia de la interacció home-sistema. Part 151: Directrius per a les interfícies d'usuari web. Proporciona directrius sobre el disseny centrat en l'usuari per a les interfícies d'usuari web amb l'objectiu d'augmentar la seua usabilitat.
- **UNE 139803:2004.** Sota el títol de "Aplicacions informàtiques per a persones amb discapacitat. Requisits d'accessibilitat per a continguts en la web", és una norma espanyola, publicada al desembre de 2004, que contempla les especificacions que han de complir els continguts web perquè puguen ser accessibles. Es tracta d'una transposició de les "Pautes d'accessibilitat al contingut en la web" (*WCAG 1.0) desenvolupades per la iniciativa WAI de W3C, però estructurades de manera diferent.
- **UNE 139803:2012**. Donades les diferències entre les WCAG 2.0 i les WCAG 1.0 va sorgir la necessitat d'actualitzar el contingut d'aquesta norma UNEIX perquè els seus requisits siguen concordes amb el contingut de les WCAG 2.0. Així, en 2012 es va actualitzar aquesta norma UNEIX per a adoptar directament les WCAG 2.0. Aquesta norma UNEIX assenyala directament quines parts de WCAG 2.0 es consideren requisits i amb quin nivell de prioritat.

És imprescindible tindre en compte les denominades mesures d'usabilitat, que són una eina clau que permet avaluar la usabilitat quant al desenvolupament d'interfícies es refereix.

Una de les eines més utilitzades són els test d'usabilitat, els quals s'encarreguen d'avaluar des de la facilitat d'ús d'una aplicació per part d'un usuari fins a si la funcionalitat implementada compleix amb la finalitat de l'aplicació. Si el desenvolupament resulta

intuïtiu per a una persona però no compleix les seues expectatives quant a l'objecte de desenvolupament, no estarà complint els criteris d'usabilitat. Els test d'usabilitat s'han de desenvolupar de manera exhaustiva perquè de manera objectiva s'avaluen tots els paràmetres desitjats. Aquests test han de contemplar unes certes mètriques que s'exposen a continuació, reactives a tres importants paràmetres: satisfacció, efectivitat i eficiència:

- **Satisfacció**: el nivell de satisfacció d'un usuari és clau per a l'avaluació de l'aplicació. Les mètriques que es contemplen sota aquest paràmetre són: qualificació de satisfacció de l'usuari sobre l'aplicació, freqüència de reutilització de l'aplicació, qualificació relativa a la facilitat d'aprenentatge o la mesura d'ús voluntari de l'aplicació.
- **Efectivitat**: determina el grau d'èxit d'una aplicació. Aquest paràmetre està estretament lligat també amb la facilitat d'aprenentatge de l'eina. S'han de tindre en compte les següents mètriques per a la seua avaluació: quantitat de tasques rellevants completades en cadascun dels intents, nombre d'accessos a la documentació, al suport i a l'ajuda, quantitat de funcions apreses o quantitat i tipus d'errors tolerats pels usuaris, entre altres.
- **Eficiència**: es defineix de manera relativa al mateix temps que es requereix per a completar una determinada tasca amb el programari implementat. Les mètriques entorn d'aquest atribut es basen en el primer dels intents: temps productiu d'ús, temps per a aprendre el funcionament de la interfície, eficiència relativa al primer intent o errors persistents, entre altres.

1.2. **Els usuaris**

Com ja hem vist, la usabilitat se centra en solucionar i donar resposta a les possibles casuístiques que ha de presentar una interfície per a poder oferir una grata experiència de navegació, siga com fora l'usuari que la utilitze. S'ha de donar, per tant, resposta a gran diversitat de capacitats. Algunes de les característiques més habituals que s'han de tindre en compte són:

- **Capacitats cognitives i perceptives**. Comprensió del llenguatge, capacitat d'aprenentatge i assimilació de conceptes, resolució de problemes.
- **Culturals**. Diversitat lingüística o nivell cultural. Això pot afectar en la interpretació de formats, mesures, títols socials, signes de puntuació, protocols i formalitats.
- **Discapacitats**. Una de les casuístiques més importants que tindre en compte durant el procés de desenvolupament d'un lloc web, quant a la usabilitat i en concret a l'accessibilitat, és l'adequació del programa desenvolupat a les persones amb alguna mena de discapacitat.
- **Tecnològica**. Connexió a Internet, grandàries de pantalla, requisits de memòria i procés. Per a finalitzar aquest apartat, és interessant considerar la següent

classificació, en funció del tipus d'usuari que pot utilitzar una determinada aplicació, la qual cosa permet definir uns certs paràmetres com els permisos d'accés d'una persona o els recursos als quals pot o no accedir.

Tipus d'usuaris en funció dels seus permisos d'accés o la seua finalitat	
Usuari anònim	Un usuari anònim és aquell que navega per la pàgina sense identificar-se com a usuari registrat o sense tindre sessió creada. Per exemple, en el cas d'un banc, un usuari sense registrar podrà accedir a la pàgina d'inici, normalment amb informació bàsica de contingut, però no tindrà accés a la zona privada.
Usuari final registrat	Quan s'implementa aquesta opció, normalment es fa per a disposar d'uns certs privilegis o recordar dades de sessió per a agilitar el procés de navegació.
Usuari beta tester	En el procés del desenvolupament de programari se sol crear un perfil per a un usuari usat com tester, amb la finalitat de realitzar les operacions oportunes per a verificar que l'aplicació funciona segons els requisits del client. Reporten les errades trobades als dissenyadors de l'aplicació que s'encarreguen de solucionar-los abans d'implantar-ho definitivament.

2. Pautes de disseny: Estructura de la interfície d'una aplicació

El disseny de l'estructura d'una interfície no és un fet trivial que haja de deixar-se a l'atzar, sinó que es tindrà en compte un conjunt de pautes de disseny que garantisquen un millor resultat FInal, per exemple, la ubicació de les finestres, el disseny de quadres de missatge i de finestres de diàleg, imatges o icones, entre altres. A continuació, s'exposen les diferents pautes que s'han d'abordar per al disseny d'una interfície.

2.1. **Pautes de disseny per a menús**

Per a la implementació d'un bon disseny, tot menú permetrà una adequada navegació per l'aplicació mostrant totes les condicions i permetent a l'usuari seleccionar les accions mostrades en aquest menú, sempre que s'adeqüen als permisos de l'usuari en aquesta aplicació. Sempre s'ha d'indicar el títol del menú, i quan aquest es mostra a l'usuari ha de contindre les opcions i l'acció associada a cadascuna d'elles, amb l'objectiu de facilitar la selecció de l'opció que més s'ajuste a les seues necessitats.

Una altra de les pautes de disseny més importants és definir una zona concreta en la qual es mostraran les diferents opcions i que no variarà al llarg del disseny, ja que, en cas contrari, l'usuari haurà de buscar en cada nova finestra on ha sigut col·locat el menú disminuint la satisfacció d'ús sobre l'aplicació.

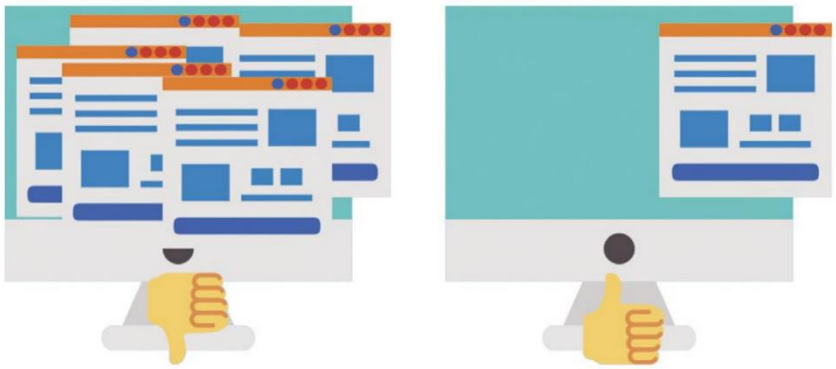
Habitualment es col·loca en la part superior de les aplicacions, però aquesta dependrà del disseny implementat. Els menús "generals" que se situen de manera estàtica en la interfície d'una aplicació solen desplegar-se en forma de cascada, mostrant nous submenús quan s'accedeix a ells. A més d'aquesta mena de menús, també és possible

trobar els denominats "contextuals" o "emergents", que apareixen en seleccionar un objecte concret. En aquesta mena de casos s'han de tindre en compte les següents pautes de disseny:

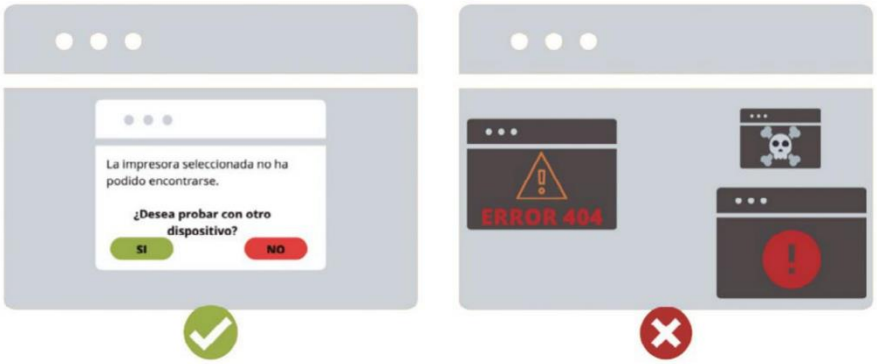
- No utilitzar menús en cascada.
- No mostrar una quantitat excessiva de funcions en aquesta mena de menús. És aconsellable reduir les opcions a un màxim de deu elements.
- Les funcions que es mostren en aquesta mena de menús també han d'estar contingudes en un altre lloc, habitualment en el menú "general" que apareix fixe en l'aplicació.

2.2. **Pautes de disseny per a finestres i quadres de diàleg**

El disseny i creació de finestres és un dels elements clau en el desenvolupament d'una aplicació, per aquesta raó, el seu disseny i posterior implementació ha de realitzar-se tenint en compte alguns aspectes molt importants, com que els usuaris siguen capaços d'obrir i tancar finestres perquè aquestes no s'interposen impedint visualitzar el que apareix en la pantalla a la qual desitgen accedir, també s'han d'habilitar els mecanismes oportuns perquè els usuaris puguen modificar la grandària de les finestres, entre altres. El nombre de finestres ha de triar-se amb especial atenció, no és aconsellable utilitzar un gran nombre d'elles que puguen saturar la pantalla i, per tant, a l'usuari. Però tampoc ha de reduir-se en excés el nombre de finestres, ja que les que siguen utilitzades quedaran saturades de contingut.



Els quadres de diàleg, igual que les finestres, són uns dels elements més importants i que permeten definir i adequar el disseny d'una aplicació a les necessitats dels futurs usuaris. Els quadres de diàleg són els que permeten establir una comunicació activa entre els usuaris i la interfície. A través de caixes de text, habitualment de tipus emergent, s'envien missatges a l'usuari, que haurà d'actuar en conseqüència a aquests. Els missatges han de ser actius i positius, indicant tota informació rellevant que haja de conéixer l'usuari i sense donar per descomptat cap mena d'informació.



2.3. **Pautes de disseny relatives al aspecte**

El disseny relatiu a l'aspecte de la interfície d'usuari posa el focus en els elements essencials: color, font i distribució dels elements. El disseny dels elements visuals ha de facilitar i millorar la usabilitat de l'aplicació, una mala selecció pot portar al fracàs a la millor de les aplicacions.

a) **Icones**: aquest tipus d'elements permet associar un objecte a una acció concreta, dinamitzant així la interacció entre la interfície de l'aplicació i l'usuari, per tant, la premissa principal quant al disseny és que ha de ser representativa de l'acció amb la qual es vincula, sent preferible triar dissenys simples, amb el mateix disseny i no excessivament complexos...

b) **Colors**: com es va analitzar a l'inici d'aquest llibre, l'elecció dels colors resulta decisiva en l'experiència dels usuaris, ja que, entre altres característiques, permet posar el focus d'atenció en aquells elements i funcions més importants. A més, aporten identitat de la marca a la qual es vincula l'aplicació, per exemple, la interfície de l'aplicació per a l'entitat BBVA presenta els colors característics de la marca.

c) **Font**: la tipografia és el tipus de lletra utilitzat per al disseny d'una interfície concreta. És convenient utilitzar una de manera uniforme al llarg de tot el disseny o, almenys, que per als mateixes tasques s'utilitze la mateixa tipografia. Ha de triar-se un tipus de lletra que facilite la lectura als usuaris i, per tant, millore la seua experiència d'ús. Però a més de la tipografia s'han de tindre en compte: la grandària de la font (que ha de ser l'adequada per a la lectura) i el tipus de pantalla i dispositiu on s'utilitzarà l'aplicació, el color i, finalment, l'estil.

2.4. **Pautes de disseny per a elements interactius**

Els elements interactius aporten al disseny de qualsevol interfície un cert comportament dinàmic que permet establir una comunicació activa entre la interfície de l'aplicació i els usuaris d'aquesta. Aquest tipus d'elements són analitzats àmpliament en els capítols inicials d'aquest llibre: botons, checkbox, combobox o radioButton, entre altres.

Les pautes de disseny relatives a la inclusió d'aquesta mena de components s'han de dirigir cap a la millora de la llegibilitat d'aquests, per exemple, en les caixes de text és convenient afegir text explicatiu que ajude l'usuari a conéixer quin tipus de dades s'han d'utilitzar per a completar les caixes de text, així mateix, també s'aconsella que la grandària de les caixes de text s'ajuste el màxim possible a la finestra en la qual són mostrats.

Quant als botons, checkbox o radioButton, és a dir, elements que permeten seleccionar un o diversos valors i enviar-los a l'aplicació per a realitzar les accions oportunes, han de complir algunes pautes de disseny:

1. Els títols han de ser intuïtius.
2. Les accions codificades en cada opció han de quedar el la suficientment comprensibles per a l'usuari.
3. Les opcions han de ser fàcilment distingibles les unes de les altres i, per tant, relativament ràpides de triar i seleccionar.

2.5. **Pautes de disseny per presentació de dades**

Una aplicació ha de presentar diferents conjunts de dades i, a més, l'exposició d'aquests es farà de múltiples formes, per la qual cosa el modelatge d'aquesta representació ha de complir unes certes pautes de disseny en funció de la mena de distribució triada: taules, gràfics … En el cas de les taules, la informació s'ha de mostrar de forma estructurada, ja que resulten clau per a posar èmfasi sobre un conjunt de dades perquè l'usuari els preste especial atenció, ara bé, no és convenient abusar de l'ús d'aquests elements, ja que si es trivialitza el seu ús l'usuari no els pararà atenció en el futur. Algunes de les pautes de disseny principals relatives a les taules són:

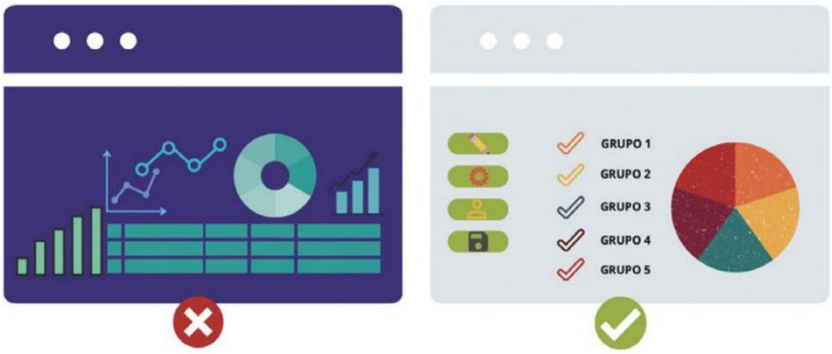
1. Utilitzar etiquetes en files i columnes, clares i concises.
2. Incloure un títol de la taula la longitud de la qual siga inferior a dues línies de text.
3. Utilitzar encapçalats de fila o columna per a resumir el contingut de la fila o columna.

Finalment, també serà possible utilitzar gràfics, però igual que en el cas de les taules, l'ús d'aquesta mena d'elements no ha de ser abusiu. Algunes de les pautes de disseny són:

1. Adequar la grandària dels gràfics a les dimensions de la pantalla.

2. No abusar del nombre de gràfics.

3. Seleccionar una paleta de color que permeta diferenciar les dades, a més, és aconsellable utilitzar una llegenda fàcil d'identificar al gràfic.



3. Accesibilitat

El concepte d'accessibilitat està relacionat d'una manera molt directa amb el d'usabilitat. En aquest cas, l'accessibilitat es pot definir com la possibilitat d'accés a una determinada aplicació, enfront de la usabilitat que es refereix a la facilitat d'ús. Per tant, és evident que una aplicació ha de ser accessible abans que usable.

L'accés a una determinada aplicació ha de facilitar-se per a tots els usuaris potencials, més enllà de les limitacions tècniques de cada usuari (programari, maquinari, etc.) o de les limitacions individuals de cadascun (discapacitats, domini d'un determinat idioma, etc.). D'aquesta manera, una aplicació accessible ha de tindre en compte la gran diversitat de potencials usuaris que pot arribar a tindre.

Encara que l'accessibilitat es troba especialment dirigida cap al desenvolupament d'aplicacions web, és convenient conéixer alguns dels seus conceptes i normes més importants, ja que en molts casos el disseny d'una aplicació serà extrapolable a diferents àmbits.

3.1. **El consorci World Wide Web (W3C)**

El consorci World Wide Web (W3C) és una comunitat internacional on les organitzacions membre s'encarreguen del desenvolupament d'estàndards que asseguren el creixement i l'accés a la web. Va ser creada en 1994 amb un conjunt d'objectius que permeteren desenvolupar tecnologies interoperables.

Dins d'aquest marc es fa necessari desenvolupar estratègies, directrius i recursos per a garantir l'accés per igual a la web, és així com apareix la Web Accessibility Initiative o Iniciativa per a l'Accessibilitat a la Web (WAI). Aquesta iniciativa va desenvolupar les Directrius d'Accessibilitat per al Contingut Web 2.0, més conegut com WCAG 2.0, on es recullen pautes i tècniques que permeten oferir solucions accessibles per al programari i contingut web.

Aquest conjunt de pautes va ser aprovat sota l'estàndard internacional ISO/IEC 40500:2012. Al juny de 2018 va ser substituït per la versió WCAG 2.1.

Els quatre principis que regulen aquest funcionament són que el disseny ha de ser perceptible, operable, comprensible i robust.

Tal com es recull en la Recomanació del W3C de l'11 de desembre de 2008 per a Pautes d'Accessibilitat per al Contingut Web (WCAG) 2.0, perquè una pàgina web siga conforme amb les WCAG 2.0, han de satisfer-se tots els requisits de conformitat següents:

1. **Nivell de conformitat.** Es recull un conjunt de criteris de conformitat, redactats en forma d'enunciats verificables sobre el contingut web, i que són utilitzats per a verificar l'adequació a l'accessibilitat d'un lloc web. Es distingeixen tres nivells de conformitat que defineixen el grau d'accessibilitat, un d'ells s'ha de satisfer per complet.
    - **Nivell A**: per a aconseguir conformitat amb el nivell A (el mínim), la pàgina web satisfà tots els criteris de conformitat del nivell A, o proporciona una versió alternativa conforme. S'han de satisfer 25 criteris.
    - **Nivell AA**: per a aconseguir conformitat amb el nivell AA, la pàgina web satisfà tots els criteris de conformitat dels nivells A i AA, o es proporciona una versió alternativa conforme al nivell AA. S'han de satisfer 13 criteris.
    - **Nivell AAA**: per a aconseguir conformitat amb el nivell AAA, la pàgina web satisfà tots els criteris de conformitat dels nivells A, AA i AAA, o proporciona una versió alternativa conforme al nivell AAA. S'han de satisfer 23 criteris.
2. **Pàgines completes**. La conformitat s'aplica a pàgines web completes, i no es pot aconseguir si s'exclou una part de la pàgina.
3. **Processos complets**. Quan una pàgina web és part d'una sèrie de pàgines web que presenten un procés, totes les pàgines en aqueix procés han de ser conformes amb el nivell especificat o un superior.
4. **Ús de tecnologies exclusivament segons mètodes que siguen compatibles amb l'accessibilitat**. Per a satisfer els criteris de conformitat només es depén d'aquells usos de les tecnologies que siguen compatibles amb l'accessibilitat.
5. **Sense interferència**. Si les tecnologies s'usen d'una forma que no és compatible amb l'accessibilitat, o està usada d'una forma que no compleix els requisits de conformitat, no ha d'impedir als usuaris accedir al contingut de la resta de la pàgina.

3.2. **Tipus de discapacitats**

Com hem vist en l'apartat anterior, existeixen diverses casuístiques d'usuaris en funció de les seues capacitats cognitives, tecnològiques o culturals. Un altre dels punts que han de tindre en compte són les discapacitats: en aquest punt veurem algunes de les més importants, relatives al que el disseny d'interfícies es refereix. Parlem de discapacitats visuals, motrius i auditives. En el següent apartat es veuran diferents tipus de tecnologies aplicades a l'accessibilitat.

A. Discapacitat visual

Es defineix com a discapacitat visual la dificultat que presenten algunes persones per a participar en activitats pròpies de la vida quotidiana, que sorgeix a conseqüència de la interacció entre una dificultat específica relacionada amb una

disminució o pèrdua de les funcions visuals i les barreres presents en el context en què es desembolica la persona. Es distingeix entre ceguesa o deficiència visual: en el primer cas es produeix una limitació total de la funció visual, mentre que en el segon no arriba a ser total, per tant, les mesures que s'han de prendre en cada cas són diferents. Per exemple, per a persones sense una ceguesa total, una possible mesura seria el disseny d'una versió de la interfície amb les grandàries més grans que en la versió original.

A l'hora de dissenyar la interfície, és desitjable que el disseny de l'aplicació o aplicació web accessible se centre en evitar les següents barreres d'accés per a les persones amb discapacitat visual:

1. **Impossibilitat d'accedir al contingut, o d'operar amb l'aplicació, des del teclat de l'ordinador.** Per a garantir l'accessibilitat d'una aplicació és necessari contemplar la funcionalitat i operativitat total a través del teclat de l'ordinador. Un dels casos més comuns és l'enviament d'un formulari a través d'un botó que només pot ser activat mitjançant el ratolí o un menú que només es desplega en passar el punter del ratolí sobre ell, per a aquests casos ha d'existir la possibilitat d'assignar tecles d'accés a qualsevol enllaç o control de formulari que proporcione una drecera mitjançant el teclat. Ara bé, és important tindre en compte que la creació d'aquestes dreceres no xoc amb els ja existents en qualsevol sistema; serà difícil trobar un ampli ventall de dreceres de tecles disponibles, per la qual cosa es recomana utilitzar tecles d'accés ràpid a elements de la pàgina web molt freqüents o de difícil localització.

2. **Absència de textos alternatius per als elements no textuals**. S'han de proporcionar accessos complementaris als elements multimèdia, ja que la seua absència pot provocar que els usuaris no puguen accedir al contingut. En el cas dels usuaris amb discapacitat visual, en utilitzar vídeos serà convenient que es descriga el que es mostra, habitualment a través d'un text que és reproduït per àudio, aconseguint així que aquests usuaris tinguen una experiència satisfactòria en utilitzar l'aplicació.

3. **Formularis i taules de dades complexes i difícils d'interpretar correctament**. Quan es creen formularis és especialment necessari cuidar uns certs aspectes d'accessibilitat per a poder garantir una correcta interpretació dels continguts als usuaris de lectors de pantalla.
   Un dels elements utilitzats pels usuaris amb discapacitat visual són els lectors de pantalla, que consisteixen en un programari que "llig i explica" el que hi ha en la pantalla. En el cas dels formularis, el programari ha de ser capaç d'identificar clarament de quina mena de control es tracta, d'indicar el seu estat i d'anunciar l'etiqueta corresponent a aquest control. Alguns dels més coneguts per a l'ús de contingut web són:

- *BrowseAloud*. Lector de pantalla destinat específicament a llegir el contingut de les pàgines web. Està disponible per a Windows i per a Mac.
- *WebAnyware*. Lector de pantalla en la web.
- *vozMe*. Explica com incorporar el component en Wordpress, Bloguer o qualsevol altra web. Permet triar entre una veu masculina o femenina i també permet descarregar un fitxer MP3 amb l'àudio.

4. **Utilització inadequada d'elements estructurals en les pàgines o falta d'estructuració en els seus continguts:** absència d'encapçalats de secció, definicions de llistes, agrupacions de controls.
   La manera en què s'estructura el contingut d'una pàgina web és fonamental en el que es refereix a la seua accessibilitat. És molt important identificar correctament els elements d'estructura bàsics com encapçalats, llestes, paràgrafs, taules de dades, etc.

   - *Llocs amb pobre contrast de color o amb informació basada en el color.*
   - *Presència de captchas en els quals no s'aporta solució accessible*. En el cas de les imatges de text visualment distorsionades, que s'usen com a mecanismes de control destinats a distingir a un humà d'una màquina o programa d'ordinador, s'han de proporcionar diferents mètodes alternatius per a accedir a la informació, adaptats a diferents capacitats sensorials.

   L'evolució tecnològica ha creat i adaptat diversos dispositius per a contribuir al fet que qualsevol usuari amb una discapacitat visual siga capaç d'utilitzar un lloc web d'igual forma que el faria si la seua capacitat visual fora la mateixa que la d'un usuari sense aquesta dificultat. En aquest cas, se centra en el desenvolupament de dispositius d'eixida, com succeeix amb les pantalles o les impressores.
   Al llarg d'aquest punt hem parlat dels lectors de pantalles, indicats per a les persones amb ceguesa total. Per a aquelles que tenen una dificultat visual parcial existeixen els ampliadors de pantalles, això són programes que permeten ampliar el text i les imatges, fins i tot existeixen navegadors especials que lligen les pàgines web i reconeixen la veu de l'usuari per a navegar per elles.

B. Discapacitat auditiva

La discapacitat auditiva es defineix com el dèficit total o parcial en la percepció del so, que s'avalua pel grau de pèrdua de l'audició en cada oïda. Es poden distingir entre les persones hipoacústiques, que són les que presenten una deficiència

parcial, és a dir, compten una resta auditiva que poden millorar a través d'audiòfons. I, d'altra banda, les persones sordes, les quals presenten una deficiència total.
La principal barrera que es destaca en aquest cas és l'incipient utilitze elements audiovisuals en la web (vídeos, animacions, sons, etc.). Perquè aquests puguen ser accessibles per a persones amb discapacitat auditiva hauran de presentar una alternativa de text.

C. Discapacitat motora

Una persona amb una discapacitat motora és aquella que pateix d'una manera duradora i sovint crònica una afecció més o menys greu de l'aparell locomotor que suposa una limitació de les seues activitats en relació amb la mitjana de la població. S'ha de conéixer també que aquesta cobreix tots els trastorns que poden causar deterioració parcial o total de les habilitats motores, incloent la part superior i/o inferior del cos.
A conseqüència d'això, en termes relatius al disseny podem trobar persones amb dificultats per a dur a terme algunes tasques informàtiques, com utilitzar un ratolí, moure un punter, utilitzar una pantalla tàctil, prémer dues tecles al mateix temps o mantindre premuda una tecla, fins i tot poden ser incapaces d'utilitzar un teclat i no poder introduir dades.
Les principals barreres d'accessibilitat que es poden produir en el disseny d'una interfície són:

- *Impossibilitat d'interaccionar adequadament amb la pàgina des del teclat o altres dispositius d'entrada.* Per a facilitar la interacció es recomana usar dispositius d'entrada dissenyats especialment per a ser utilitzats amb l'ordinador en general i la navegació en particular . Per exemple, teclats amb tecles grans, teclat convencional amb cobertor, ratolí de bola gran o trackball, ratolí de botó, entre altres.
- *Enllaços gràfics i altres elements accionables que no estan etiquetats correctament i no són accessibles als reconeixedors de veu*. De nou, igual que ocorre amb les discapacitats visuals en el cas dels lectors, en aquest seran de rellevant importància els reconeixedors de veu, a través dels quals els usuaris seran capaços de realitzar les operacions desitjades, ara bé, per a això el disseny dels elements han de ser fàcilment accionables a través d'aquesta mena de dispositius.

Igual que en el primer cas, el desenvolupament de dispositius d'eixida ha millorat l'accessibilitat, en aquest cas, serà el programari o dispositius d'entrada els que recullen la nostra atenció. Per exemple:

- *Programari de reconeixement de veu*. Programari indicat per a aquells casos en els quals no es pot fer ús del teclat o del ratolí.
- *Trackball*. Consisteix en un dispositiu similar a un ratolí, però que no requereix del desplaçament d'aquest per a funcionar, sinó que el desplaçament es duu a terme utilitzant una bola que en girar permet el desplaçament del ratolí per la pantalla.

- *Càmeres web*. A través del reconeixement facial, permeten traduir el moviment de la cara o els ulls en moviment del ratolí per la pantalla.

4. Anàlisi i verificació de la usabilitat

És fonamental tindre la capacitat d'analitzar i verificar quin grau d'usabilitat té una determinada aplicació i la seua interfície sobre la base dels objectius i necessitats dels usuaris. Aquesta informació permetrà realitzar possibles millores o modificacions en el producte, a fi d'augmentar la usabilitat d'aquest.

Aquest procés d'anàlisi i verificació de la usabilitat se sol dur a terme en la fase d'avaluació d'un projecte.

Existeixen diversos mètodes que s'utilitzen comunament per a aquesta mena de tasques. A continuació, es defineixen alguns dels mètodes més àmpliament utilitzats en el desenvolupament d'aplicacions web.

4.1. **Mètode per inspecció. Avaluació heurística**

Aquest tipus de mètode es duu a terme per professionals experts en usabilitat, que es dediquen a analitzar de manera completa, identificant possibles problemes que és necessari corregir. La base d'aquest mètode és tant la pròpia experiència dels experts que avaluen el lloc com els codis de bones pràctiques o guies existents per a detectar uns certs principis relacionats amb la usabilitat.

Alguns d'aquests principis poden ser els següents:

- Compliment de directrius d'accessibilitat.
- Utilització del mateix llenguatge entre aplicació i usuari.
- Informació aportada a l'usuari per part del sistema sobre el procés que està duent a terme, és a dir, sobre el que està succeint. Un exemple típic d'aquest fet és quan un determinat usuari vol accedir a un contingut audiovisual que ofereix el portal, i el portal l'informa que està procedint a la càrrega en buffer del contingut.
- Fomentar el control de la interfície de l'aplicació per part de l'usuari. Per exemple, és important que l'usuari tinga la capacitat de poder fer marxa enrere en alguna acció que haja dut a terme, ja que pot tractar-se d'un error. Així mateix, és important facilitar i orientar a l'usuari per a poder resoldre un error determinat, per exemple , a l'hora de completar un camp d'un formulari.
- Incloure documentació d'ajuda que puga ser consultada en un moment per l'usuari.
- Estructura adequada de la informació mostrada en l'aplicació.
- Inclusió d'elements multimèdia i adequació dels mateixos a la temàtica del lloc.
- Qualitat adequada del contingut quant a llenguatge i redacció es refereix.

El gran avantatge d'aquest mètode respecte a uns altres és que es pot desenvolupar d'una manera senzilla, ràpida i eficaç en un termini breu de temps, encara que el seu cost pot ser major.

4.2. **Mètode de test amb usuaris**

Al contrari que en el mètode per inspecció, aquest mètode es basa en l'anàlisi d'una aplicació a través d'un grup d'usuaris reals, de manera que aquests usuaris puguen detectar problemes d'utilització o bé plantejar opcions de millora.

És important tindre en compte que aquest mètode pot implementar-se una vegada que l'aplicació ja es troba en la fase de producció perquè els usuaris tinguen una visió completa de la funcionalitat de l'aplicació, però també pot dur-se a terme durant les fases de disseny i implementació d'aquesta. Aquest fet és molt important, ja que és menys costós corregir problemes durant aquestes fases del disseny que fer-lo en la fase de producció.

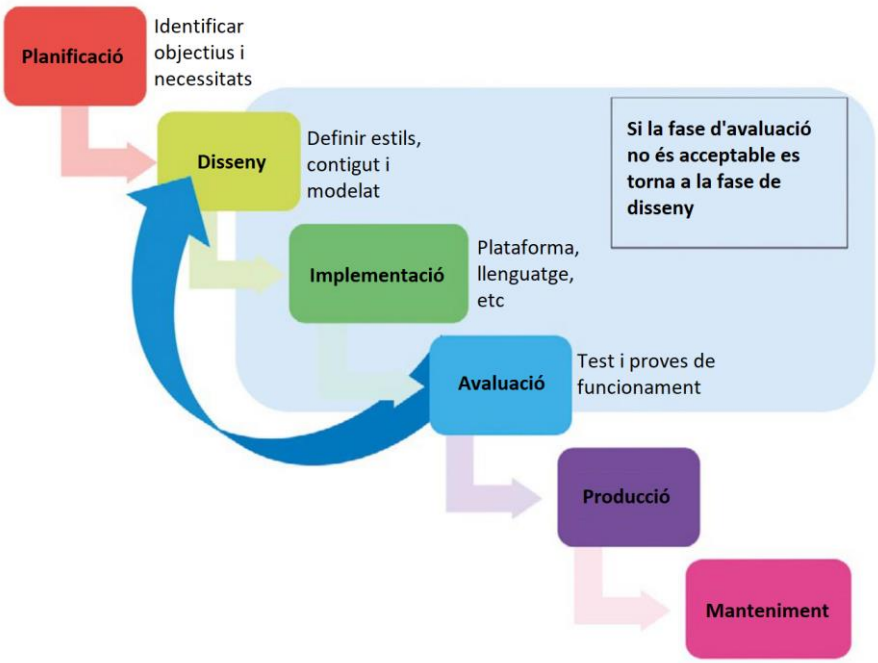
Aquest no té per què considerar-se com un mètode contraposat al mètode d'inspecció, sinó que tots dos poden complementar-se en l'anàlisi de la usabilitat.

Així mateix, el gran avantatge d'aquesta mena de mètodes és que els resultats són més fiables i, a més, s'aprofundeix en el descobriment d'errors de disseny relacionats amb la usabilitat.

No obstant això, cal tindre en compte l'elevat cost que pot suposar, ja que per a dur-lo a terme serà necessari disposar de diversos usuaris, així com situar-los en una zona adequada per a fer la seua tasca, prestant-los les eines necessàries per a això.

5. Anàlisi i verificació del procés de desenrotllament d'interfícies

A l'hora d'implementar un determinat projecte, s'han d'establir unes fases o processos per a facilitar la consecució de l'objectiu. És a dir, és necessari aplicar una sèrie de procediments, tècniques i mètodes específics per a l'objecte que es persegueix. D'aquesta manera, per a la realització d'una interfície i la seua aplicació s'estableixen les següents fases:



5.1. **Fase de planificació**

Tot projecte, siga de l'àmbit que siga, ha de planificar-se amb anterioritat al seu inici, independentment de l'objectiu que perseguisca. En aquesta fase és necessari fixar principalment els objectius que es pretenen aconseguir, a més dels mitjans a través dels quals s'arribarà a aquests.

De manera més concreta, en el cas d'una aplicació s'hauran de tindre en compte detalls com els recursos professionals i tècnics necessaris, tipus d'emmagatzematge, costos. A més, serà necessari establir les bases dels continguts que tindrà el lloc i de com es disposaran els mateixos per a proporcionar una experiència grata a l'usuari.

D'altra banda, en aquesta fase és especialment crític que s'obtinga la major quantitat possible d'informació sobre l'usuari, ja siga a través d'enquestes, entrevistes, reunions, etc., per a identificar necessitats, objectius, comportaments.

5.2. **Fase de disseny**

Durant aquesta fase s'implementarà l'aspecte i funcionalitats que tindrà l'aplicació, tenint sempre en compte la importància de la usabilitat, així com tota la informació obtinguda durant la fase de planificació.

És fonamental que durant aquesta fase el dissenyador valore els tipus d'usuari que pot tindre l'aplicació per a considerar les seues necessitats, limitacions i habilitats, i adaptar el desenvolupament de l'aplicació a aquestes.

En el disseny d'aplicacions amb diverses pantalles i elements de navegació és molt aconsellable realitzar de manera prèvia un esquema de continguts i organització del lloc, identificant enllaços entre menús o finestres. També serà aconsellable documentar de

manera adequada totes les accions que es realitzen per a facilitar la seua comprensió a dissenyadors externs que puguen requerir aqueixa informació arribat el cas.

5.3. **Fase d'implementació**

Durant aquesta fase, es realitzarà el disseny plantejat en la fase anterior, obtenint-se d'aquesta manera les primeres versions operatives de l'aplicació. És fonamental en aquest cas l'haver considerat adequadament els llenguatges de programació que s'utilitzaran, així com les plataformes de desenvolupament.

És molt aconsellable tindre l'opció de realitzar versions prèvies o prototips de l'aplicació per a poder analitzar sobre les mateixes l'experiència de l'usuari quant a usabilitat i accessibilitat es refereix.

5.4. **Fase d'avaluació**

Es tracta sens dubte d'una de les etapes més crítiques del procés, atés que sobre la mateixa es prendrà la decisió de passar l'aplicació a producció o, per contra, quins aspectes han de modificar-se i com es durà a terme. Per això, es tracta d'una fase a partir de la qual es pot tornar a la fase de disseny o bé avançar a la fase de posada en producció.

5.5. **Fase de producció**

Com el seu propi nom indica, durant aquesta fase es duu a terme la posada a la disposició dels usuaris de l'aplicació, una vegada ha sigut comprovat i analitzat el seu funcionament en les etapes anteriors del procés.

5.6. **Fase de manteniment i seguiment**

Es tracta de l'última fase del procés, encara que ha de tractar-se com una fase crítica. Es tracta d'una fase fonamental, pel fet que la tecnologia evoluciona contínuament, i amb ella les necessitats i possibilitats dels usuaris, per la qual cosa és molt important adaptar i millorar el lloc a partir d'això.

6. El projecte de desenvolupament

La divisió en fases d'un projecte permetrà organitzar tot el procés que comporta el desenvolupament d'un nou producte, ja es físic o digital. És a dir, la divisió en fases aportarà un ordre lògic a la gestió del projecte, simplificant la gestió d'aquest en tasques més xicotetes i manejables.

6.1. **Fases d'un projecte de desenvolupament**

En qualsevol projecte de desenvolupament (també, lògicament, en el desenvolupament d'aplicacions) existeix una sèrie de fases típiques que poden distingir-se, segons s'indica en el següent gràfic:

En el desenvolupament d'aquestes fases és important considerar un protocol de proves en cadascuna que ajude a detectar possibles problemes i inconvenients, abans del pas a la següent fase. És un procés d'especial rellevància en qualsevol projecte, però especialment delicat en l'àmbit del desenvolupament d'aplicacions. La raó és senzilla: detectar un problema de planificació o disseny en la fase d'avaluació o producció pot suposar un augment de costos considerable, en comparació a la seua detecció en fases més primerenques.

De forma general, podem diferenciar les proves que es realitzen en cadascuna de les fases o mòduls del desenvolupament del programa, de les realitzades sobre el funcionament general de l'aplicació. És evident que aquestes últimes es fan una vegada estan finalitzats i integrats tots els mòduls del sistema, típicament en la fase d'avaluació.

És molt important dissenyar un protocol de proves exhaustiu, que permeta garantir el correcte funcionament de l'aplicació i, per tant, l'èxit d'aquesta. Al llarg d'aquest tema vorem els nombrosos tipus de proves existents.

6.2. **Objectius principals del sistema de proves**

Com es va indicar anteriorment, les fases de proves són indispensables per al correcte desenvolupament de qualsevol aplicació, independentment del seu àmbit. És fonamental implementar i dissenyar un protocol de proves correcte i exhaustiu, en funció de cada cas en concret.

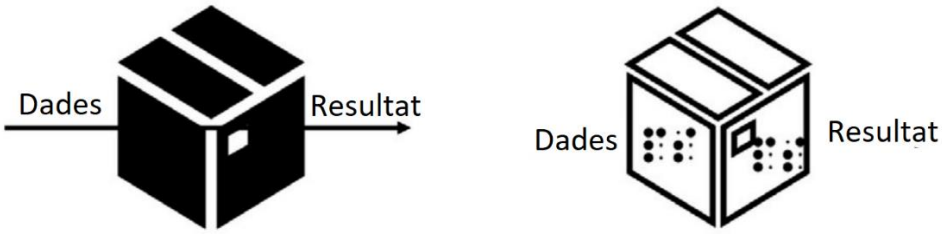
Són múltiples els avantatges que un bon sistema de proves aporta al desenvolupament d'una aplicació:

- Permeten validar la manera en què el programari funciona, comparant-lo amb les especificacions i objectius de partida.
- Permeten detectar i corregir errors en el programari en fases més primerenques, facilitant així la seua depuració al llarg del desenvolupament.
- Permet provar i avaluar l'aplicació en diferents escenaris i situacions.

6.3. **Proves de caixa negra i caixa blanca**

Pràcticament la totalitat dels desenvolupaments d'aplicacions de programari es basen en dos grups de proves amb diferents característiques entre si:

a) **Proves de caixa negra:** en les quals s'avalua l'aplicació des d'un punt de vista extern, és a dir, sense preocupar-nos del "interior". Són les habituals per a la prova d'interfícies.
b) **Proves de caixa blanca:** es basen en l'avaluació del codi intern del programari. Un bon disseny per a aquestes proves implica l'avaluació de tots els possibles camins que s'han implementat en el disseny d'un programa.



6.4. **Depuració de codi**

Com a resultat de realitzar proves sobre un desenvolupament programari, serà necessari en molts casos depurar el codi, en funció dels resultats d'aquestes proves i de l'objectiu final de l'aplicació. El procediment habitual sol ser el mostrat en la figura.

En el cas de la depuració de codi, l'objectiu principal és detectar i corregir errors en aquest, en molts casos a partir de les proves realitzades en cadascuna de les fases del projecte. En relació amb el desenvolupament de codi, poden aparéixer bàsicament tres tipus d'errors:

- **Errors de compilació**. En gran part dels casos ocorren a causa de la sintaxi, que varia en funció del llenguatge de programació que s'utilitze.
- **Errors d'execució.** Solen aparéixer quan existeixen operacions incorrectes, que no són permeses. Habitualment, el sistema generarà un missatge d'error indicant aquest motiu.
- **Errors de lògica.** Estan motivats per un error en el disseny del programa, ja que el resultat que es produeix no és l'esperat. És més complicat de detectar i, per tant, de corregir, ja que en aquest cas el programa s'executa de manera normal, sense llançar cap error..

De fet, la majoria de les aplicacions que s'utilitzen per al desenvolupament de codi (com a Eclipse o NetBeans) inclouen eines de depuració que ajuden a detectar errors.

El procés de depuració de codi va íntimament lligat als processos de proves. Segons els resultats obtinguts, s'intenta localitzar l'error en el codi, tindre en compte el mateix per a

implementar la correcció necessària (el que és la depuració de codi pròpiament dita) i, finalment, tornar a provar de nou el programa.

7. Tipus de proves

De manera addicional a la classificació de proves com de caixa negra o de caixa blanca, és possible plantejar altres tipus de classificacions, segons s'indica a continuació, segons la mena de funcionalitat que s'avalue en cada cas.

7.1. **Proves unitaries**

Aquest primer tipus de proves, les proves unitàries, són les utilitzades per a avaluar funcionalitats concretes, examinant tots els camins possibles implementats en el desenvolupament d'un algorisme, funció o classe. Per tant, podem dir que una prova unitària és aquella que permet comprovar el funcionament d'un dels mòduls que formen el programa. Després d'avaluar el funcionament unitari de l'aplicació, es procedeix amb les proves en les quals s'engloben la resta de mòduls.

7.2. **Proves d'integració**

Les anomenades proves d'integració s'utilitzen amb l'objectiu d'aportar una garantia relacionada amb el funcionament adequat de l'aplicació, una vegada s'han integrat tots els mòduls que componen la mateixa, d'ací el nom d'aquestes proves.

Aquest tipus de proves ofereixen la possibilitat d'analitzar el correcte funcionament de tots els mòduls d'una forma conjunta, ja que és possible que s'hagen aplicat altres tipus de proves sobre aquests mòduls, però de manera separada. Per tant, la importància d'aquesta mena de proves a l'entorn del desenvolupament d'aplicacions és fonamental, ja que és habitual que una aplicació d'una certa complexitat estiga composta per diversos mòduls.

Podem distingir dos tipus concrets:

a) **Proves d'integració ascendent.** Com el seu propi nom indica, aquest tipus de proves consisteixen a avaluar en primer lloc els nivells o mòduls més baixos de l'aplicació, per a anar pujant en els mateixos de manera gradual..
   Aquestes proves es realitzen de manera grupal, per al que és necessari una figura (denominada "controlador") que s'encarregue de coordinar aquestes proves.
b) **Proves d'integració descendent.** En aquest cas, les proves es realitzaran de manera inversa respecte a les proves d'integració ascendent, és a dir, es comença des del mòdul principal i es va descendint gradualment.

7.3. **Proves de regressió**

Les anomenades proves de regressió es caracteritzen per ser un conjunt de proves que s'havien executat anteriorment sobre el sistema, i que s'utilitzen per a buscar evidències relacionades amb modificacions i/o canvis que s'haja pogut produir sobre el codi del

programa, amb la finalitat de detectar si han pogut ocasionar nous errors o fallades, que anteriorment no havien aparegut.

Entre elles existeixen diversos tipus, amb característiques diferenciades entre si:

a) **Proves de regressió locals**: tenen la fi de trobar errors que hagen sigut ocasionats per canvis o modificacions recents en el codi.
b) **Proves de regressió desemmascarades o al descobert:** en aquest cas, aquestes proves tenen lloc quan la realització de canvis ocasiona problemes que no tenen cap relació amb aquests, però que s'han detectat arran de la seua inclusió.
c) **Proves de regressió remota o a distància:** estan relacionades amb l'aparició de problemes ocasionats en integrar les diferents parts d'una aplicació, considerant-se que aquestes parts per separat no llançaven aqueix mateix problema.

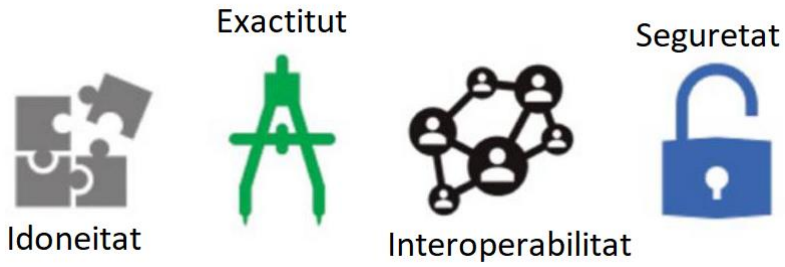
Com es pot deduir d'aquesta mena de proves, són fonamentals per a poder validar de manera correcta tots els canvis que es facen en el codi de l'aplicació.

7.4. **Proves funcionals**

Les anomenades proves funcionals es caracteritzen per avaluar al complet les funcionalitats que s'indiquen en l'especificació de l'aplicació o eina desenvolupada, concretament en relació amb els seus requisits de disseny.

Igual que ocorre amb la resta de les proves que es realitzen, és molt important que el procés es documente de manera completa, amb la finalitat de facilitar la seua anàlisi i comprensió.

Així mateix, la normativa **ISO 25010** especifica una sèrie de característiques que han de complir aquestes proves funcionals per a assegurar que estiguen correctament definides. Aquestes característiques es recullen al gràfic següent i suposen un conjunt de característiques que seria desitjable complir.



Aquestes proves han de ser realitzades per personal amb experiència en el seu desenvolupament perquè els resultats puguen ser interpretats de manera adequada i correcta. Així mateix, és molt important que es realitzen diverses propostes de millora si es cree convenient, o fins i tot canvis i/o modificacions si es cree necessari o si els resultats obtinguts no foren satisfactoris.

7.5. **Proves no funcionals: de capacitat, rendiment, ús de recursos i seguretat**

Les proves funcionals es basen en analitzar com es comporta l'aplicació a nivell intern. A diferència d'aquestes, les proves no funcionals solen utilitzar-se amb l'objectiu d'avaluar com es comporta l'aplicació a nivell extern.

Segons el que hem estudiat al llarg d'aquest capítol, es pot afirmar que les proves funcionals són proves de caixa blanca, mentre que les proves no funcionals són proves de caixa negra.

A continuació, es recullen les principals proves no funcionals, al costat de les seues característiques més destacades:

a) **Prova de capacitat.** El seu ús sol estar lligat a avaluar el comportament de l'aplicació davant una situació d'estrés. És a dir, s'utilitzen per a comprovar la resposta del sistema davant l'augment de les peticions o càrrega de treball que rep, aplicant-se una situació d'estrés sobre aquest.
b) **Prova de rendiment.** L'objectiu d'aquesta mena de proves és comprovar com es comporta l'aplicació des del punt de vista de la seua eficiència, tenint en compte paràmetres especialment importants com el temps de respostes i la velocitat de processament. Són proves molt importants per a decidir si és necessari o no optimitzar processos (i, per tant, codi).
c) **Prova d'estrés.** Solen estar relacionades amb les proves de capacitat, ja que es basen a llançar diverses peticions al sistema, amb l'excepció que en aquest cas el que es pretén avaluar és com es recupera l'aplicació davant una situació de sobrecàrrega, més que la resposta que es dona.
d) **Prova de volum.** També se solen desenvolupar de manera paral·lela a les anteriors, ja que persegueixen avaluar el sistema des del punt de vista de la seua capacitat per a processar grans volums de dades.
e) **Proves de seguretat.** Corresponen a una mena de proves d'una gran importància, ja que tenen com a objectiu final aportar una garantia sobre la integritat de les dades de l'aplicació. Així mateix, avaluen diferents mecanismes o formes de protecció per a l'aplicació, amb l'objectiu de millorar la seua robustesa i seguretat.

7.6. **Proves manuals**

Les proves manuals es caracteritzen per ser executades per part de l'encarregat de desenvolupar el codi, amb l'objectiu de poder provar el seu funcionament i, en cas de ser necessari, modificar la seua implementació.

És a dir, per a aquestes proves és el mateix desenvolupador programari el que les executa, avaluant per si mateix si la resposta de l'aplicació a una determinada entrada és la correcta o no, en funció del codi implementat amb anterioritat.

Per a la seua execució no s'empra una eina determinada, sinó que depén de cada programador. Habitualment, els entorns de desenvolupament més habituals (com a Eclipse o NetBeans) incorporen eines i/o funcionalitats per a facilitar la realització d'aquesta mena de proves.

7.7. **Proves automàtiques**

Contràriament a les proves manuals, les proves automàtiques requereixen l'ús d'eines de proves per a poder executar-se. En tots dos casos, es tracta de proves complementàries entre si i que tenen una gran importància per a aconseguir un desenvolupament programari correcte.

Amb les proves automàtiques, la seua execució sol ser més ràpida que en el cas de les manuals, ja que, al cap i la fi, aquestes últimes les executa una persona. A més, permeten comprovar el funcionament de l'aplicació davant diverses variacions en les dades, i repetir les proves d'una manera ràpida i senzilla.

Se solen utilitzar per a la realització de proves de regressió, ja que aconsegueixen una optimització del procediment.

El número i tipus d'eines existents en el mercat per a dur a terme aquest tipus de proves és molt ampli, per la qual cosa l'ús de l'una o l'altra dependrà de l'objectiu final, de la mena de prova, de les preferències del desenvolupador i, finalment, de l'aplicació que s'està desenvolupant. A continuació, es destaquen algunes de les més utilitzades en l'àmbit del desenvolupament d'aplicacions programari:

- **JMeter**: és una eina desenvolupada per la prestigiosa companyia Apatxe, que facilita la realització de proves de rendiment i de càrrega sobre l'aplicació.
- **Bugzilla**: es tracta d'una aplicació executada en línia, que s'usa per a realitzar el seguiment d'errors en els mòduls del programari en cadascuna de les seues versions.
- **JUnit**: es tracta d'un grup de llibreries desenvolupades a Java i que s'utilitzen per a realitzar proves unitàries sobre aplicacions que han sigut desenvolupades en aquest llenguatge.

- **Cucumber**: és una eina de programari lliure, que facilita realitzar proves d'acceptació sobre aplicacions web. S'utilitza Ruby com a llenguatge per a generar scripts.
- **Selenium**: es tracta d'un grup d'eines que s'utilitzen habitualment per a realitzar proves d'aplicació sobre desenvolupaments d'aplicacions web.

7.8. **Proves d'usuari**

Les proves d'usuari, tal com el seu propi nom indica, són executades per un o diversos usuaris reals del sistema o aplicació que s'està desenvolupant, que no tenen per què tindre coneixements de programació. S'utilitzen de manera complementària a altres tipus de proves que són executades per experts, ja que és possible que aquests últims no hagen tingut en compte unes certes situacions que poden donar-se en un entorn real d'ús de

l'aplicació. És a dir, són proves basades en l'experiència de l'usuari en la realització de les seues tasques.

Encara que no existeix un procediment fixe, pot resultar òptim que participen en aquesta mena de proves almenys quinze usuaris, ja que és lògic pensar que quants més usuaris proven l'aplicació, més situacions d'error podran detectar-se, encara que tampoc és viable que tots els possibles usuaris de l'aplicació participen en les proves. En qualsevol cas, les proves s'executen segons un guió previ, la qual cosa suposa una diferència substancial respecte a les proves alfa i beta. També pot resultar aconsellable que participen en les mateixes diferents tipus de perfils, per a provar al seu torn diferents funcionalitats i procediments.

7.9. **Proves d'acceptació**

Les anomenades proves d'acceptació s'executen sobre una aplicació determinada per a corroborar que el seu funcionament compleix amb els requeriments inicials de disseny, és a dir, el seu funcionament és l'esperat en el moment del disseny de l'aplicació. És evident que es tracta d'una mena de proves d'especial rellevància per a l'èxit de l'aplicació.

Aquestes comprovacions es realitzen des del prisma del rendiment de l'aplicació, així com des de la seua funcionalitat, atés que en tots dos casos es plantegen requeriments en la fase de disseny del desenvolupament. Evidentment, l'aplicació resultant ha de satisfer les necessitats plantejades quant a funcionament, però també davant temps de resposta, comportament, estabilitat, etc., amb l'objectiu de que l'experiència d'ús siga grata i adequada.

Algunes altres característiques d'aquesta mena de proves són les següents:

- És habitual que aquestes proves siguen definides pel client per al qual s'està desenvolupant l'aplicació. És evident que el grau de satisfacció sobre el producte dependrà de les expectatives existents sobre aquest.
- Solen executar-se de manera prèvia a la implantació definitiva de l'aplicació.
- Igual que ocorre amb altres tipus de proves, és molt important que es documenten de manera correcta, recollint-se totes les evidències possibles sobre aquestes.

7.10. **Desenvolupament del pla de proves**

Com és evident, el procediment de proves de qualsevol desenvolupament programari està compost per proves molt diverses, amb característiques diferents entre si. Com hem estudiat, l'aplicació de tota aquesta bateria de proves és fonamental per a garantir el correcte funcionament de l'aplicació i, per tant, tindre més possibilitats d'èxit.

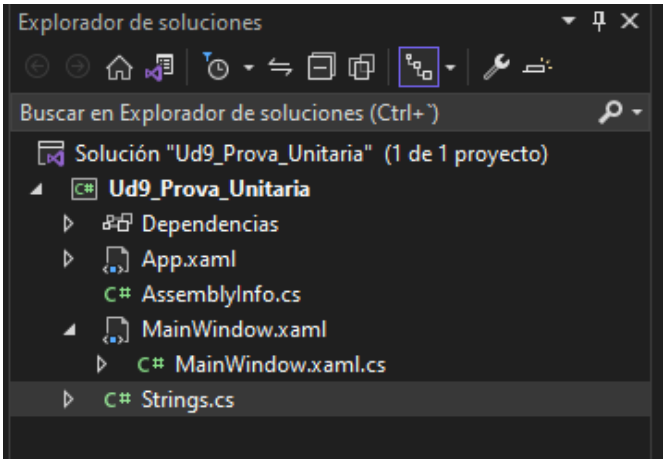
Habitualment, el flux que se segueix per a executar aquestes proves és el que es mostra en la figura següent, encara que aquest fet dependrà de cada cas concret:

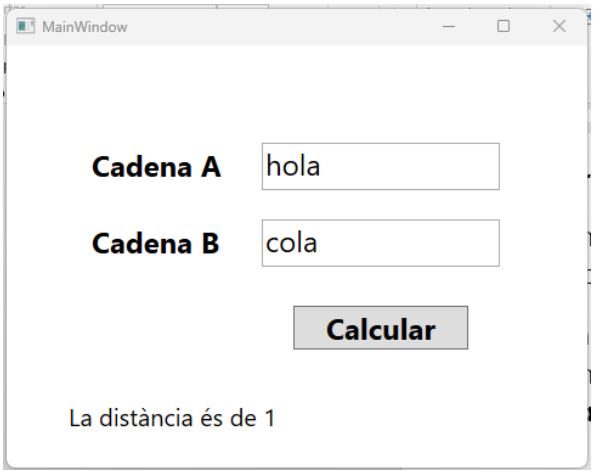
7.11. **Proves unitàries amb VSC**

A continuació veurem com podem fer proves unitàries (en quest cas, provar el correcte funcionament del mètode d'una classe) dins de Visual Studio.

Partirem d'una solució amb una classe anomenada **StringsUtils** i dins d'ella un mètode anomenat **GetHammingDistance** que calcula els canvis necessaris (cada canvi és reemplaçar un caràcter) per a convertir una cadena en altra (les dues han de ser de la mateixa llargària).

Vegem primer unes captures de la solució comentada:





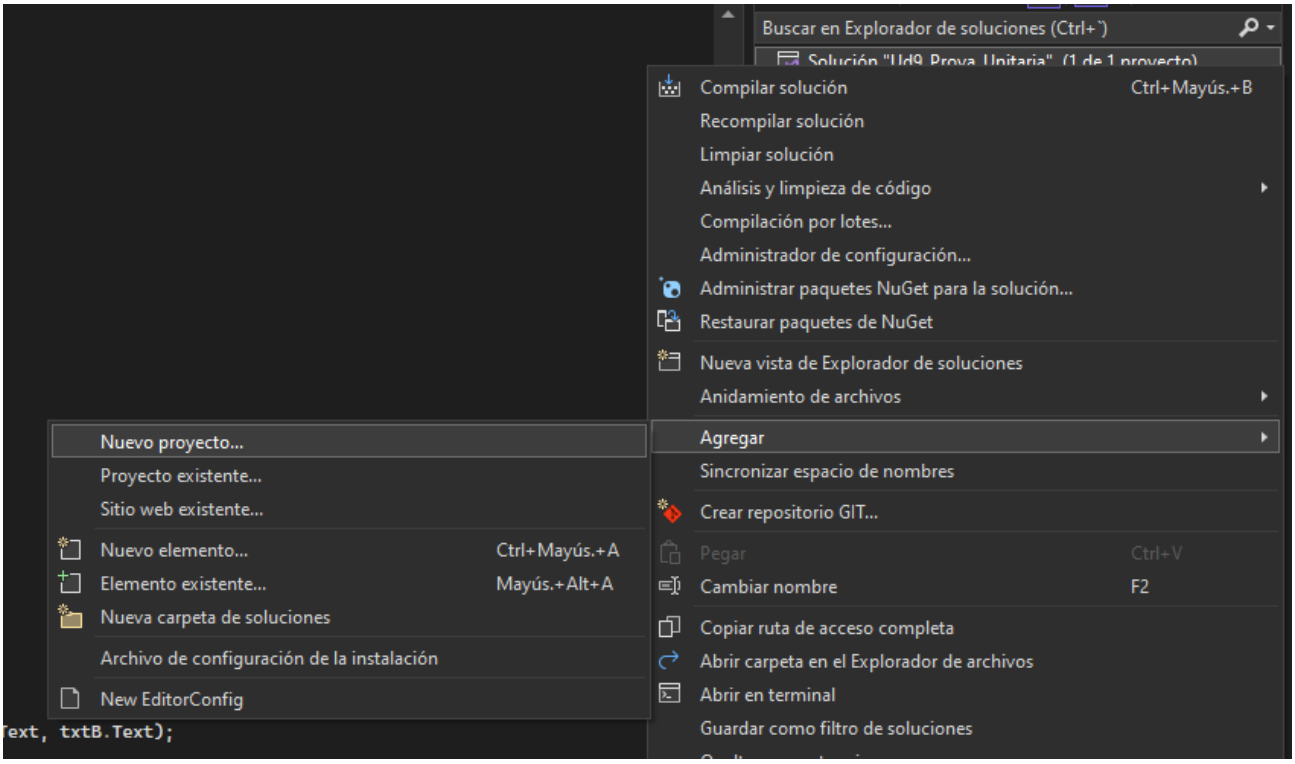
El codi de la classe:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ud9_Prova_Unitaria
{
    1 referencia
    public class StringsUtils
    {
        1 referencia
        public static int GetHammingDistance(string s, string t)
        {
            if (s.Length != t.Length)
            {
                throw new Exception("Les cadenes han de tindre la mateixa llargaria");
            }

            int distance =
            s.ToCharArray()
            .Zip(t.ToCharArray(), (c1, c2) => new { c1, c2 })
            .Count(m => m.c1 != m.c2);

            return distance;
        }
    }
}
```

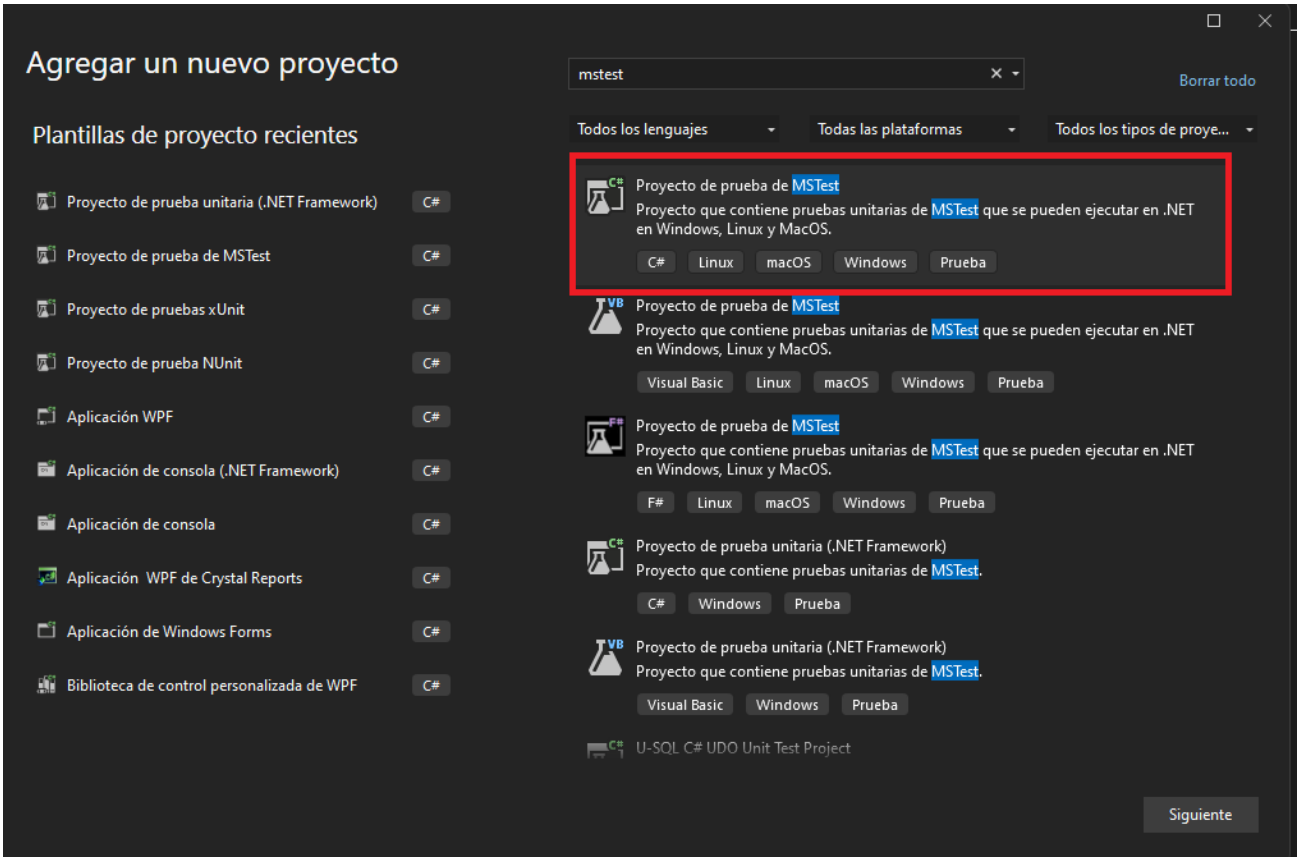
I com la utilitzem:

```
namespace Ud9_Prova_Unitaria
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    2 referencias
    public partial class MainWindow : Window
    {
        0 referencias
        public MainWindow()
        {
            InitializeComponent();
        }

        1 referencia
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                lblResultat.Content = "La distància és de " + StringsUtils.GetHammingDistance(txtA.Text, txtB.Text);
            }
            catch (Exception ex)
            {
                lblResultat.Content = ex.Message;
            }
        }
    }
}
```

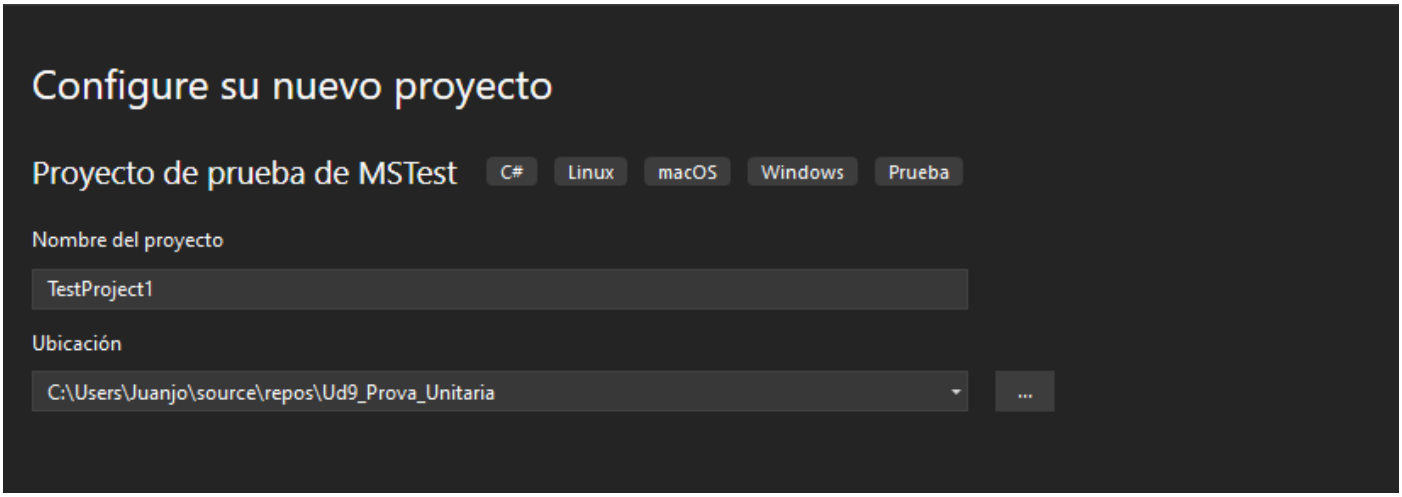
El primer que hem de fer es afegir altre projecte a la solució, serà un projecte de tipus "Projecte de prova unitària". Per a fer això polsem amb el botó dret del ratolí sobre la solució i triem l'opció **"Agregar"→"Nou Projecte":**



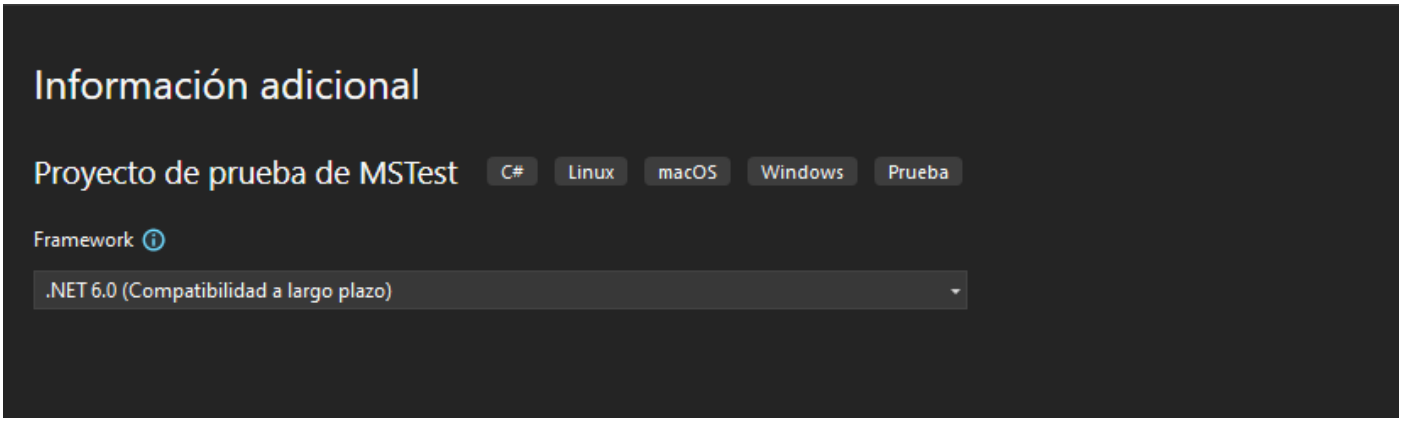
Filtrem les plantilles amb la paraula "mstest" i triem la plantilla del projecte amb C# (existeixen altres plantilles igualment vàlides, però nosaltres començarem amb aquesta):



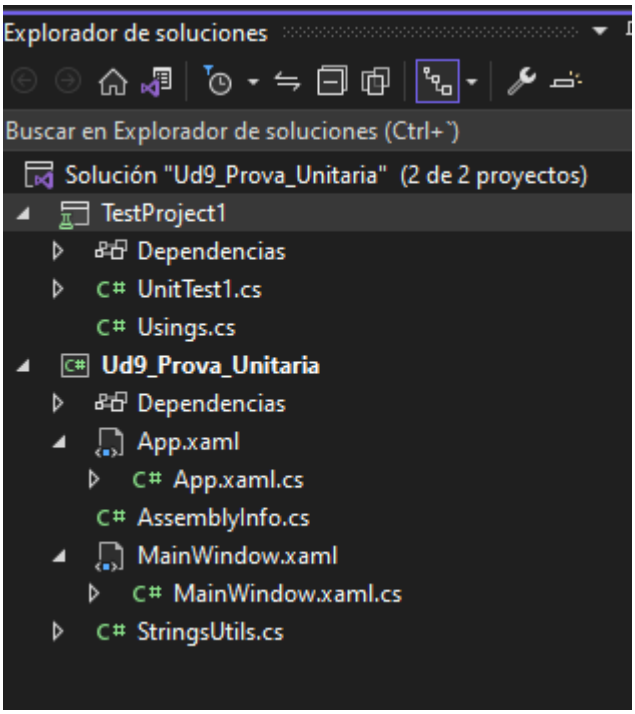
Polsem **Següent**:



Polsem **Següent** (podem canviar el nom al projecte):



I finalment polsem **Crear**. Ja tenim el projecte de proves creat:

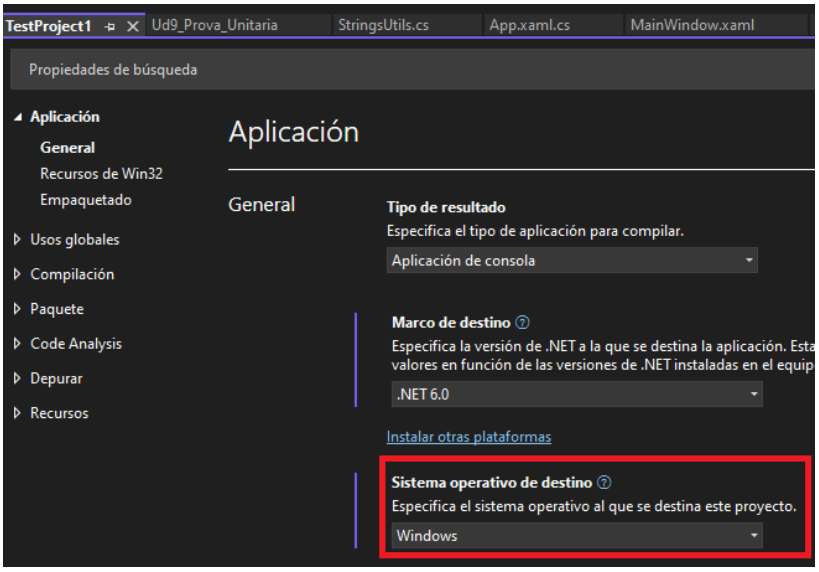


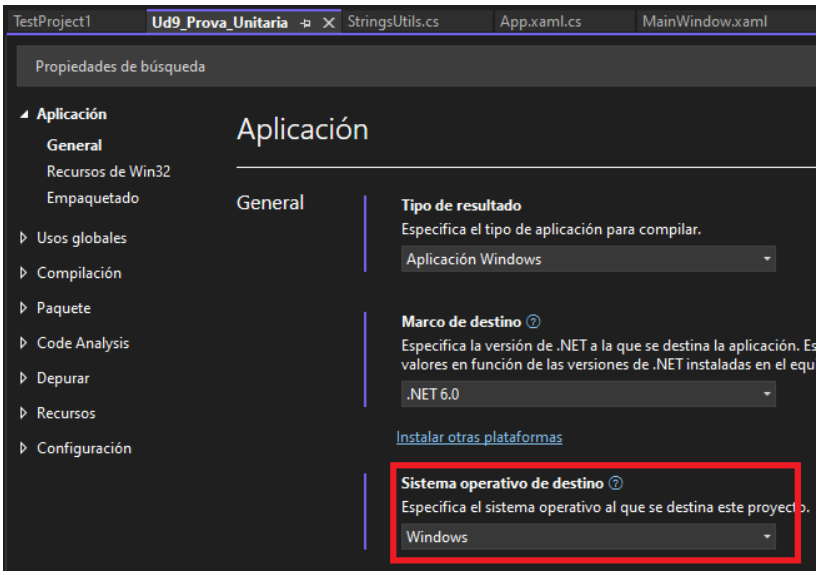
Observem el contingut del fitxer **UnitTest1.cs**

```
namespace TestProject1
{
    [TestClass]
    0 referencias
    public class UnitTest1
    {
        [TestMethod]
        0 referencias
        public void TestMethod1()
        {
        }
    }
}
```

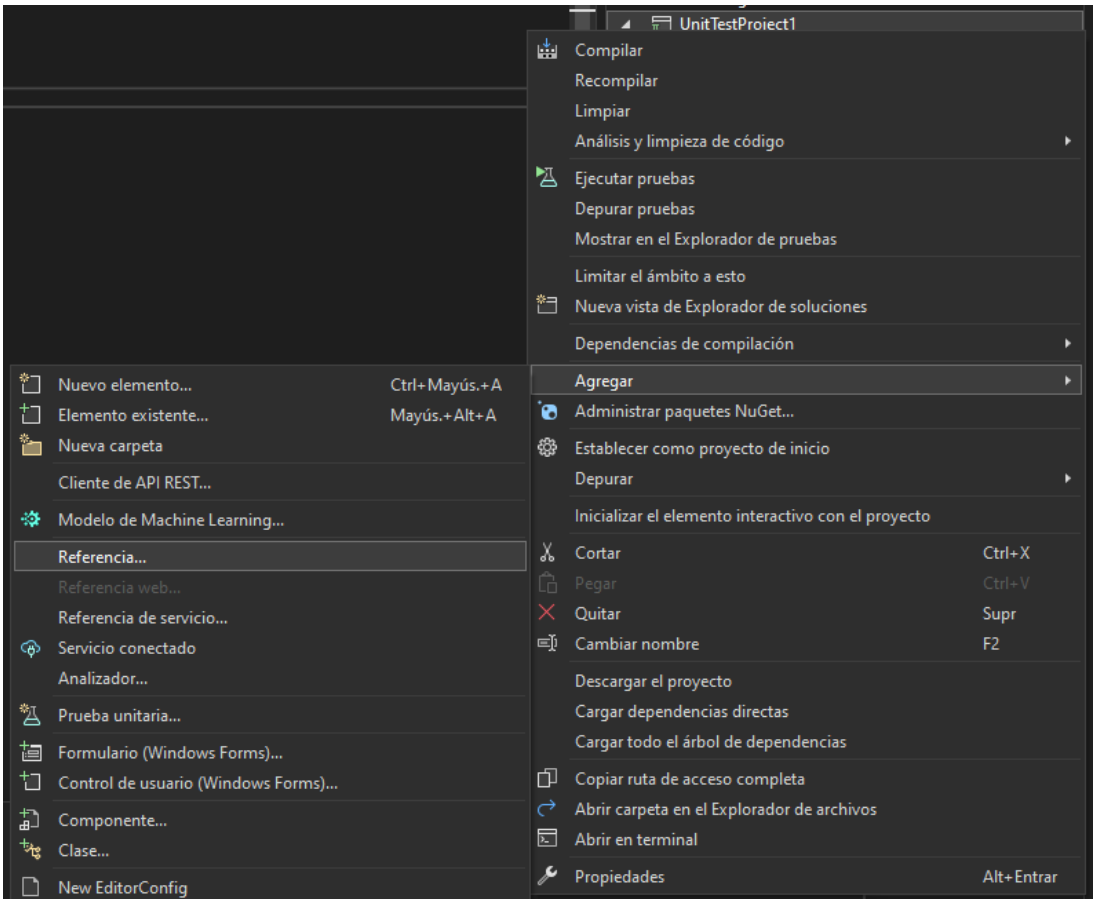
Els atributs **TestClass** indica que la classe és una classe per a realitzar proves i l'atribut **TestMethod** indica que els mètodes següents estan destinat a fer proves sobre alguna classe. Dissenyarem un parell d'ells.

Abans de continuar hem de comprovar que la plataforma destí dels dos projectes (tan el de l'aplicació com el de proves) tenen la mateixa **Sistema Operatiu de destí**. Al nostre cas. **Windows.** Això hem de comprovar-ho i configurar-ho a les propietats de cada projecte:

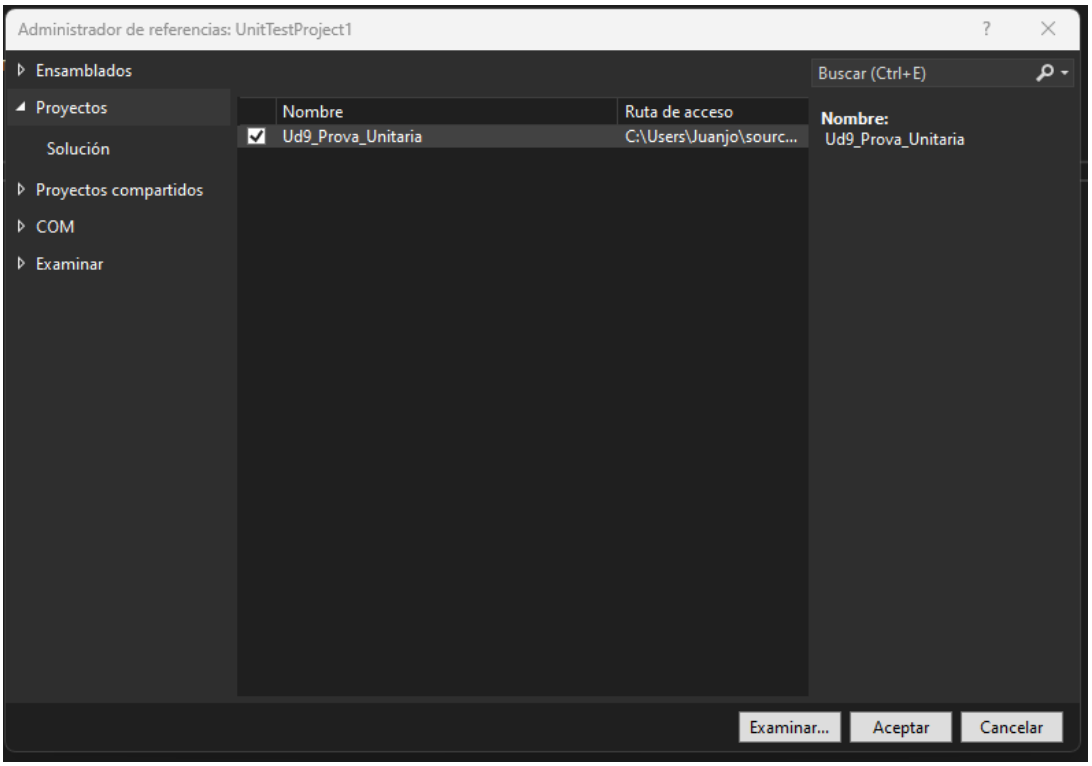




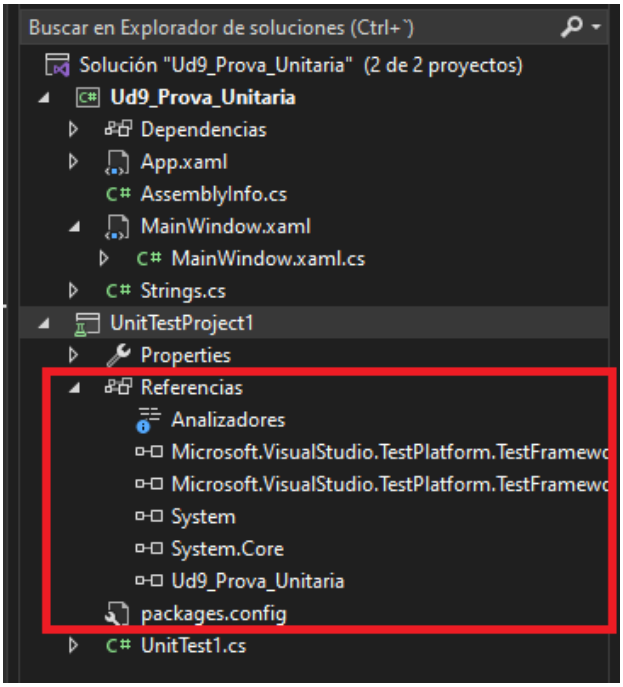
El següent pas és afegir una referència a la classe de desitgem provar. Per a fer això hem de polsar amb el botó dret del ratolí sobre el projecte de proves i triar l'opció **Afegir→Referència**:



Busquem el projecte que desitgem provar i polsem **Acceptar**:



Si observem les referències del projecte de prova podem observar la classe del projecte referenciat:



Ja ho tenim tot, ara començarem a fer el primer mètode prova per a provar el mètode **GetHamingDistance** de l'aplicació. Tornem al fitxer Unitest1.cs del projecte de prova i farem alguns canvis:

```
using Ud9_Prova_Unitaria;

namespace TestProject1
{
    [TestClass]
    0 referencias
    public class UnitTest1
    {
        [TestMethod]
        0 referencias
        public void TestMethod1()
        {
            //Preparació de la prova
            string cadena1 = "caracola";
            string cadena2 = "parabola";
            int resultatEsperat = 2;

            //Execució de la prova
            var resultat=StringsUtils.GetHammingDistance(cadena1, cadena2);

            //Comprovació
            Assert.AreEqual(resultatEsperat, resultat);
        }
    }
}
```

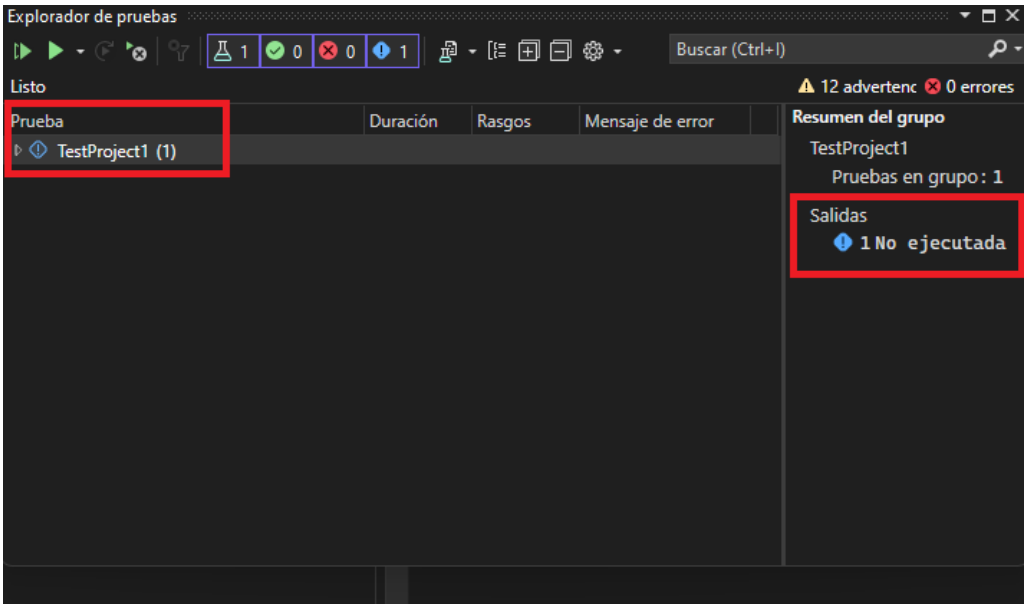
El primer que fem al codi és indicar (amb **using**) que volem utilitzar les classes del projecte referenciat abans.

A continuació comencem a crear el primer mètode proves. Observa que diferenciem tres parts dins del mètode TestMethod1:

- **Preparació**: Preparem la prova. En aquest cas definim dos cadenes i el resultat esperat (en aquest cas 2)

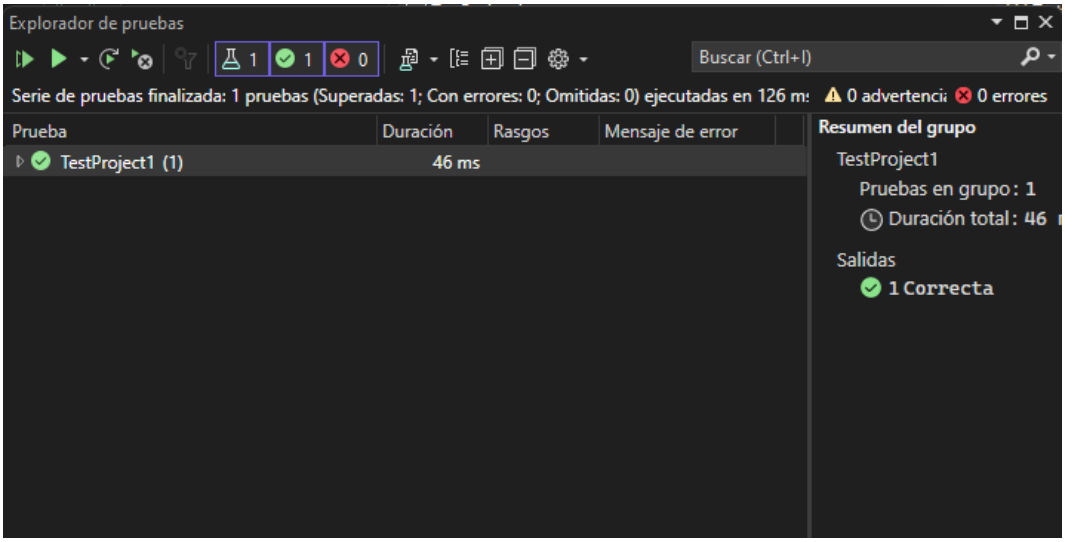
- **Execució**: Executem la prova i guardem el resultat.
- **Comprovació**: Amb la classe **Assert** podem fer múltiples comprovacions. En aquest cas comparem si el resultat esperat i el obtingut son iguals.

Per a executar el test hem d'obrir una finestra anomenada Explorador de Proves (podem obrir-la des de el menú **Veure→Explorador de Proves**):



En aquesta finestra observem el mètode de prova creat (de moment únicament tenim un) i que encara no s'ha executat.

Per a executar-lo podem seleccionar el mètode de prova i polsar el triangle verd (play):

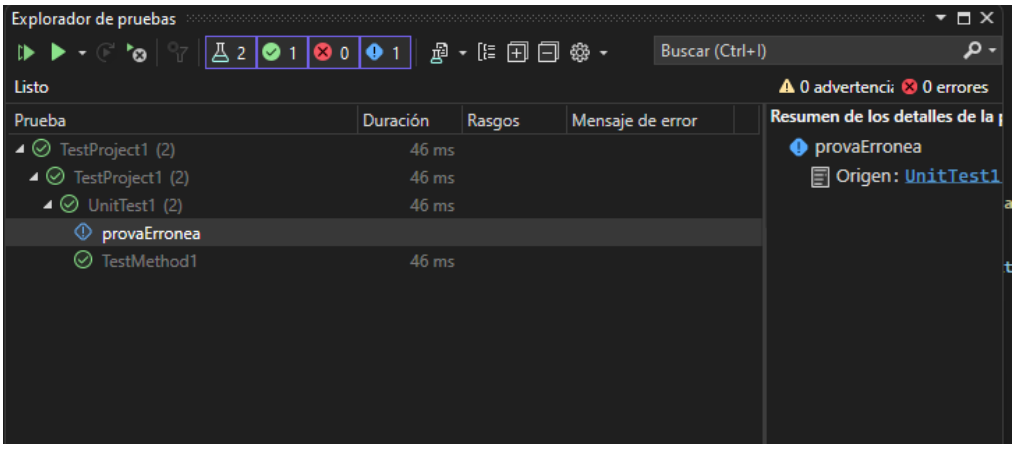


Quan termina el color del mètode proves canvia a ver i a la dreta s'indica el resultat de la prova (en aquest cas ha sigut correcta, efectivament els canvis entre la paraula caracola i parabola son 2).

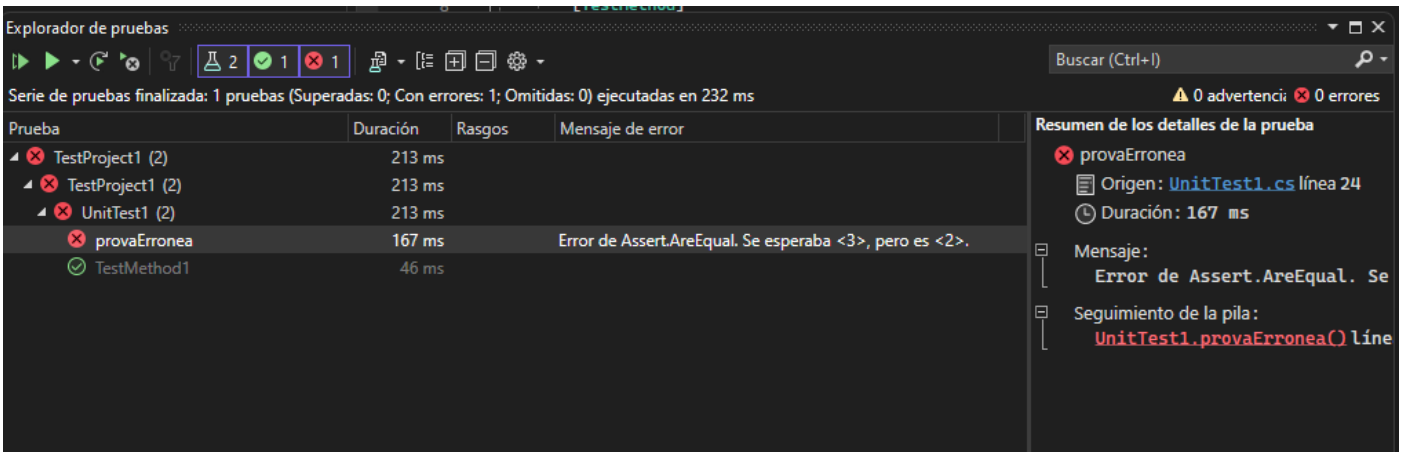
Vegem que passa quan el test no te èxit. Creem ara un nou mètode de prova amb dues cadenes, forçant que el resultat esperat no és el resultat que torna el mètode de l'aplicació. Observa el següent codi:

```
[TestMethod]
0 referencias
public void provaErronea()
{
    //Preparació de la prova
    string cadena1 = "caracola";
    string cadena2 = "parabola";
    int resultatEsperat = 3;

    //Execució de la prova
    var resultat = StringsUtils.GetHammingDistance(cadena1, cadena2);

    //Comprovació
    Assert.AreEqual(resultatEsperat, resultat);
}
```

Ara tenim altre mètode amb les mateixes cadenes però resultat esperat diferent. Obrim l'Explorador de Proves i despleguem les proves dins del projecte de proves:



Vegem que ara tenim una prova pendent d'executar. La seleccionem i la executem:

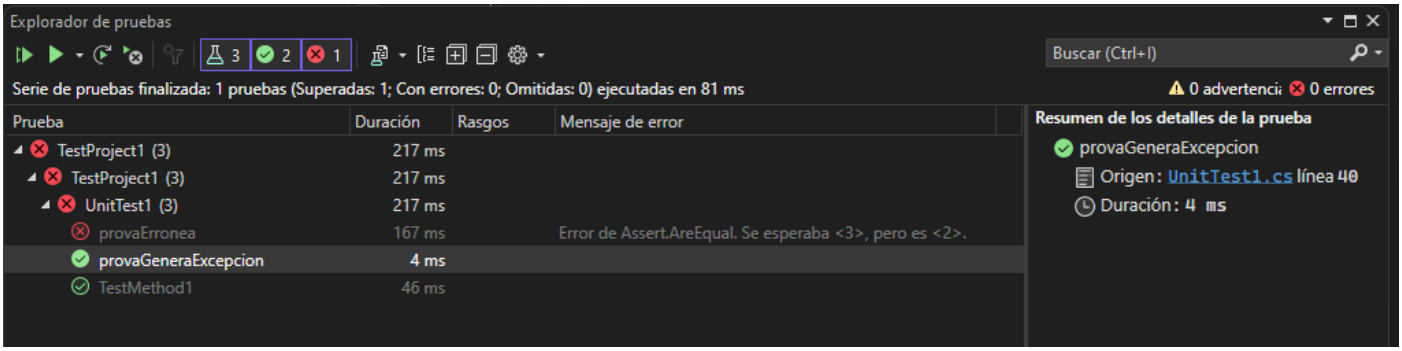


I com esperàvem la prova no termina amb èxit. A més ens informa del problema amb el **Assert**, el resultat obtingut i el resultat esperat.

Per acabar, també podem crear un mètode de prova per a comprovar que un mètode llança una excepció davant d'un valor no esperat o no vàlid. En aquest cas passarem una de les dos cadenes amb valor **null**. Observa el següent codi:

```
[TestMethod]
[ExpectedException(typeof(NullReferenceException))]
0 referencias
public void provaGeneraExcepcion()
{
    //Preparació de la prova
    string cadena1 = "caracola";
    string cadena2 = null;
    int resultatEsperat = 3;

    //Execució de la prova
    var resultat = StringsUtils.GetHammingDistance(cadena1, cadena2);

}
```

Com s'aprecia en la imatge, ara afegim un atribut (ExpectedException) per a indicar que el que s'espera del mètode és que llance una excepció degut a que una de les dos cadenes es nul·la. També pots observar que per aquest tipus de prova no cal fer cap comprovació amb un **Assert**. Executem la prova:



I podem comprovar que el mètode de prova ha tingut èxit.

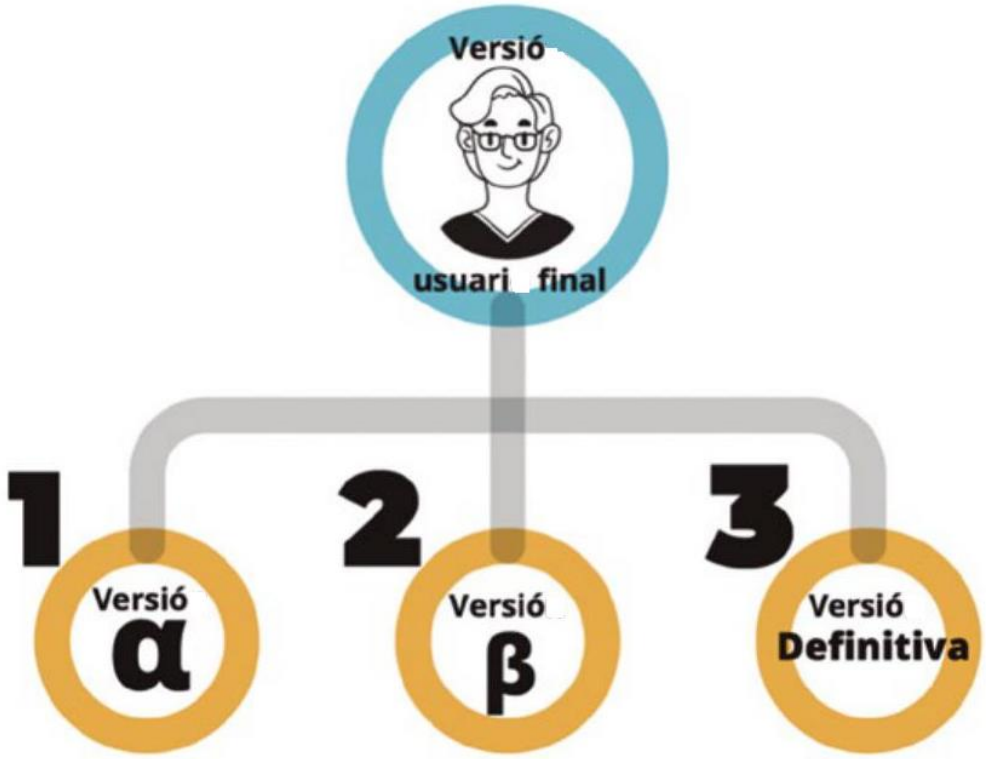
8. Versions Alfa i Beta

Aquestes proves s'utilitzen per a detectar errors que només poden ser descoberts per part de l'usuari final, i que no poden ser detectats per totes les proves realitzades de manera prèvia a la posada en producció de l'aplicació. Els tipus de proves estudiats fins ara en el tema es basen en la mirada del programador, de l'expert o també de l'usuari, però segons unes pautes o passos preestablerts.

Per tant, és molt habitual que es creen versions de proves de l'aplicació resultant, que són les denominades versions alfa i beta, l'objectiu de les quals és recollir possibles errors que no hagen sigut detectats anteriorment, i que només poden recollir-se quan es testa l'aplicació per usuaris reals, en els diferents entorns i condicions en els quals es vaja a utilitzar la mateixa.

Es tracta de conceptes que se solen utilitzar també en altres camps diferents del desenvolupament programari, com en la fabricació d'altres productes o béns.

En la següent imatge es resumeixen les diferents versions que sol tindre una aplicació:



8.1. **Versió ALFA**

Es coneix com a versió alfa d'una aplicació a la primera versió d'aquesta, que serà provada inicialment per un grup de persones, l'objectiu de les quals és simular l'ús que li donaria el client o usuari final. Evidentment, es tracta d'una versió que encara no està del tot finalitzada per a poder implantar-se en producció, sinó que ha de ser prèviament depurada i avaluada.

Habitualment s'executen en les mateixes oficines del client final per a reproduir més fidelment l'escenari real d'ús.

8.2. **Versió BETA**

Es tracta d'una versió pràcticament final de l'aplicació. De fet, es tracta de la primera versió del desenvolupament que serà provada directament pel client o usuaris finals d'aquesta, fet que en si mateix és la principal diferència respecte a les proves alfa.

És molt comú que es realitzen en un entorn d'aplicació el més semblant al real per a evitar que el mateix condicione el resultat del seu ús.

Els usuaris que proven aquestes versions es coneixen com a "beta tester".