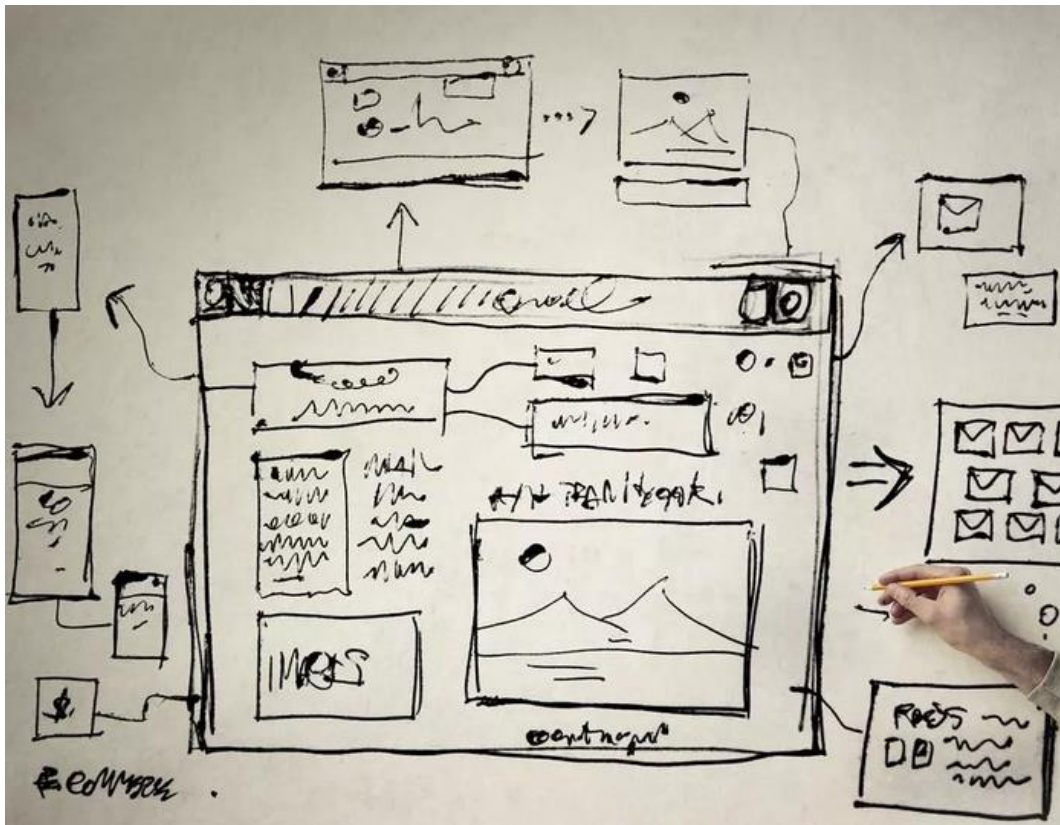


Desenvolupament d'interfícies

Unitat 1. Introducció al disseny d'interfícies



Índex

1.	Introducció	2
2.	Disseny d'una interfície. Disseny gràfic.	2
3.	Programació orientada a esdeveniments.	5
4.	Eines.	6
4.1.	Prototips. Elements clau d'un prototip.	6
4.2.	Aplicacions per al desenvolupament d'interfícies.	8
5.	Recomanacions per al disseny d'una interfície	10
5.1.	Llei de Fitts.	13
6.	El color	14
6.1.	El Sistema RGB	14
6.2.	Matís, saturació i lluentor	16
7.	Patrons de disseny	17
7.1.	Patró Model-Vista-Controlador	17
7.2.	Patró Model-Vista-Model de Vista	19
8.	Estàndards de facto	20
9.	Instal·lació de Visual Studio Community	21
9.1.	Instal·lació de Visual Studio Community	21
9.2.	Creació del primer projecte	26

1. Introducció

En l'actualitat pràcticament la totalitat de les aplicacions existents interaccionen amb l'usuari a través d'una interfície gràfica d'usuari (GUI) derivada, normalment, de la interfície del sistema operatiu sobre el que es vaja a executar l'aplicació: Windows, Linux, entorn Mac...

La facilitat i la riquesa proporcionada pels entorns de desenvolupament per a crear formularis i agregar controls: botons, llistes, marcs, barres de desplaçament... permet que una mateixa aplicació pugui tindre resolta la seua interfície d'usuari de múltiples maneres segons la capacitat artística del dissenyador.

Les interfícies han de ser homogènies i compartir tant detalls de presentació tan evidents com el color de text i fons de la finestra o la grandària de la font com a maneres d'operació com poden ser Tallar i Pegar.

Algunes de les qüestions que s'han de resoldre quan es realitza una interfície són les següents:



Com
organitzar la
finestra?

Quins controls
utilitzar?

Sincronització
entre
components

2. Disseny d'una interfície. Disseny gràfic.

El disseny gràfic consisteix en programar, projectar i realitzar comunicacions visuals d'aplicacions o altre tipus d'eines de programari. A l'actualitat, d'aquesta àrea de desenvolupament s'encarreguen uns professionals determinats, anomenats dissenyador gràfics. Aquests professionals de dotar a aquestes aplicacions de les següents funcions:

- Funció estètica.
- Funció publicitària.
- Funció comunicativa.

Dins del disseny d'interfícies també podem distingir 4 grups d'elements:

- Elements conceptuals
- Elements visuals
- Elements de relació
- Elements pràctics

Per tant, la interfície final serà un conjunt d'elements gràfics distribuïts en un disseny que permeteix una millor presentació i navegació per la aplicació. Si aquests dos factors no es donen a la mateixa vegada i el resultat final de la interfície es òptim, serà fruit de la casualitat.

Observem a la següent imatge dos dissenys, un roïn i altre millor.



Com es pot observar a la imatge, son dues interfícies per a una mateixa aplicació. En la de la esquerra s'observa un disseny "no adequat", els elements no apareixen clarament diferenciats, no es possible llegir amb comoditat els textos (degut als colors empleats), no es veuen bé les imatges, no existeix una coherència a les imatges dels botons, ... En canvi al disseny de la dreta, la navegació de l'usuari es intuïtiva, les imatges dels botons son coherents, els colors son adequats així como la ubicació dels elements, proporcionant així un major grau de satisfacció a l'usuari a l'ús de la aplicació.

Al desenvolupament de qualsevol aplicació i concretament al disseny de la seua interfície, s'han de tindre en compte diferents fases: des de la definició d'objectius que es volen aconseguir al projecte fins disseny visual final, però (evidentment) sense oblidar les especificacions funcionals de l'aplicació.

En aquesta imatge, es defineixen totes aquestes fases que s'han de tindre molt presents durant el desenvolupament d'una interfície.



Després d'identificar les tasques que ha de realitzar l'aplicació, es definiran els objectes i les accions: esta forma d'actuar es similar la definició de classes a un disseny orientat a objectes. El més adequat és primer fer un modelat textual de l'escenari de l'aplicació, diferenciant entre substantius i verbs. Els primers representen els objectes i elements y els segons les accions sobre els anteriors (igual que al un disseny orientat a objectes). Dins dels objectes es habitual diferenciar diferents tipus d'objectes:

- **Objectes origen:** Elements de l'aplicació que poden ser desplaçats a altre lloc, deixant-los sobre un objecte destí o interferint sobre un objecte destí.
- **Objectes destí:** elements que reben objectes origen o son interferits per objectes origen.
- **Objectes d'aplicació:** elements de l'aplicació que permeten interaccions de l'usuari sobre ell.

Un exemple d'objectes origen i destí podria ser una icona que representa un document (objecte origen) i és arrossegada sobre una icona en forma d'impressora (objecte destí) per que siga imprès.

Un objecte d'aplicació seria, per exemple, un text informatiu sobre l'ús de l'aplicació.

Un dels aspectes més importants a tindre en compte al moment de dissenyar una interfícies és el tipus de pantalla on es mostrarà. Per aquest motiu s'ha de realitzar de forma exhaustiva el disseny a aplicar, la col·locació de les icones i objectes, la forma de interacció, la ubicació i grandària dels textos, ...

Una vegada fet el disseny, s'ha de fer una revisió del mateix per a descobrir les possibles problemes i corregir-los.

Per últim hem revisar altres aspectes molt importants a l'aplicació amb l'objectiu de millorar tot el que siga possible l'experiència de l'usuari:

Temps de resposta del sistema: El temps de resposta fa referència a dos conceptes clau: la duració i la variabilitat del mateix. El primer està molt clar, és el temps que triga el sistema en completar una acció concreta. Ha de ser adequat, si és molt curt l'usuari pot pensar que no ha fet bé l'acció i és molt llarg per fer l'usuari no torne a fer l'acció en un futur. Aquest temps ha de ser constant (variabilitat) i així l'usuari l'associarà a un funcionament correcte i adequat de l'acció dins de l'aplicació.

Ajuda a l'usuari: A la actualitat els dos tipus d'ajuda més habituals son: les **ajudes contextuais o integrades** en l'aplicació que apareixen de forma automàtica durant l'ús de l'aplicació (suggeriments o guies al començar una tasca); i les ajudes complementaries en forma de manuals dins de l'aplicació o disponibles a la web del desenrotllador de l'aplicació.

Identificació o etiqueta d'ordres o accions: L'aplicació ha de tindre una nomenclatura clara associada a cada acció, es a dir, que el element textual o visual que representa a una determinada funció siga una paraula o imatge adequada. També es recomanable associar combinacions de tecles o tecles de funció a les accions o tasques més comuns.

3. Programació orientada a esdeveniments.

La visió de la programació tradicional canvia, ja que les aplicacions passen de seguir una seqüència a orientar-se **en funció dels esdeveniments o senyals que l'usuari o el sistema proporcionen a l'aplicació**. Es proporcionen eines per al dibuix de la interfície mitjançant controls o components que se situen en la plantilla com: botons, etiquetes de text, botons d'opció, etc. **Posteriorment s'enllacen les accions associades a cadascun dels controls com a resposta a un senyal o esdeveniment**. Per exemple, en pitjar el botó de guardar s'hauran de seguir els passos adequats perquè el fitxer es guarde en el lloc indicat per l'usuari.

Un esdeveniment és qualsevol acció reconeguda per un control o formulari. Per exemple el fer un clic amb el ratolí, estrényer una determinada combinació de tecles o simplement que el sistema ha arribat a una determinada hora. Com a conseqüència, l'objecte afectat respon a aquest esdeveniment executant codi en el lloc corresponent.

Cadascun dels objectes que constitueixen la interfície de l'aplicació té la possibilitat de respondre a un conjunt d'esdeveniments ja establits (per exemple, un clic, un doble clic, la càrrega de l'objecte en memòria, etc.). Ara és el programador el que decideix a quins respondrà cada objecte i quin serà el codi que s'executarà en cada cas. Aquest **codi constitueix l'anomenat procediment d'esdeveniment per a aqueix objecte i aqueix esdeveniment particular**.

Quan s'inicia l'execució de l'aplicació es carrega i es presenta el formulari inicial; a continuació, l'aplicació queda en espera que es produïska un esdeveniment. Aquest pot ser provocat per l'usuari, pel mateix sistema, o fins i tot per alguna acció del codi.

Seguidament es produirà la corresponent resposta a aqueix esdeveniment executant-se el procediment d'esdeveniment associat, si existeix. Una altra vegada l'aplicació quedarà en espera que es produïska un nou esdeveniment. D'aquesta manera, **mitjançant l'encadenament d'esdeveniments, el programa s'executa.**

És necessari considerar el conjunt de successos o esdeveniments que es puguin donar quan l'usuari interactua amb la nostra aplicació, ja que en funció dels esdeveniments programats el programa es comportarà d'una forma o una altra. Per això, ha de tindre en compte un conjunt d'hipòtesis de partida sobre l'estat de la interfície i del que pot ocórrer sobre ella. A més, com ja s'ha comentat anteriorment, el mateix codi pot generar esdeveniments: per exemple, la càrrega d'un formulari en memòria dona lloc a l'esdeveniment Lloeu, el canvi del contingut d'un quadre de text provoca l'esdeveniment Change, etc.

4. Eines.

El primer pas per al disseny d'una interfície és fer un esborrany o prototips. Una bona versió prèvia o prototipus millorarà la velocitat de desenvolupament de l'aplicació. Existeixen molt tipus de eines que ens permetran dur a terme la construcció de prototipus que inclouran menús, finestres, quadres de diàleg, desplegable, tractament de errades, etc.

Altres dels punts més importants del desenvolupament de un bon disseny es la avaluació del mateix, la forma en la que es realitza aquesta avaluació pot ser determinant. La forma més adequada de fer aquesta avaluació seria així:

1. Després de que l'usuari haja provat l'aplicació emplenarà un formulari en el que apareixeran diverses qüestions quantitatives sobre el grau de satisfacció de l'aplicació i el temps empleat per aprendre el funcionament de l'aplicació (si ha tingut que mirar molt l'ajuda, a tingut que retrocedir a sovint, ...).
2. Amb les dades recollides es pot comprovar si l'aplicació compleix els requisits inicials.
3. Si l'anàlisi anterior no compleix amb les expectatives inicials, serà necessari revisar les especificacions inicials i solucionar els problemes relacionades amb errades o defectes. Aquest procés es repetirà fins que es compleixen els requisits de disseny de la interfície.

Les eines disponibles per desenvolupament d'interfícies poden establir-se en una primera classificació entre comercials i lliures. Principalment trobarien dos grups, Microsoft per a les primeres i altres per a les segones.

Altra classificació seria en funció del llenguatge i/o dispositiu final, encara que actualment la majoria d'eines permeten crear interfícies per a una gran quantitat de dispositius.

4.1. Prototips. Elements clau d'un prototip.

Un prototip consisteix en una mena de esborrany o disseny inicial per a fer-se una idea de la aplicació final que s'obtindrà. El prototip permet comprovar el resultat dels diferents dissenys finals, comprovar algunes funcionalitats i realitzar tests d'usabilitat. Normalment a un prototip no hi ha programació lògica, únicament un disseny visual i simulacions de

funcionament. D'aquesta manera s'estalvia temps, esforços i diners ja que serà més senzill fer canvis sobre aquests dissenys previs que sobre l'aplicació final.

Aplicar la tècnica de prototips al disseny d'interfícies es fonamental, ja que ens permet crear una aplicació visual però buida de codi, a la que es pot observar els menús, elements visuals i altres parts del disseny. Aquest prototip s'anirà refinant amb la retroalimentació del client final i quan estiga llista es començarà a escriure el codi de dins de l'aplicació.

Els avantatges més destacables de l'ús de prototips son:

- **Millora la velocitat de desenvolupament:** Es més eficient fer canvis al disseny d'un prototip, ajustant menús, elements, colors, etc. abans de tindre un disseny definitiu. A més al aparèixer color i tipografies el client no es fixarà en el disseny base sinó en el prototip (esquelet de l'aplicació), que és l'objectiu del disseny d'interfícies i del prototip como a primer pas de desenvolupament.
- **Fa que el client estiga compromès:** El client forma part del procés d'evolució del prototip, opinant, proposant canvis i arribant junt al dissenyadors a un esquelet/disseny final de l'aplicació. Es prioritari involucrar al client en aquesta fase.

No existeix un tipus únic de prototips, hi ha molts. Podem trobar esquemes de pàgines, wireframes, mockups, esborranys, sketches, diagrames, ... Però principalment podem distingir en tres tipus principals:

- **Sketching:** Es dibuixa tota la interfícies de l'aplicació en paper, identificant els processos i la relació entre pantalles. Aquesta tècnica es la que se sol aplicar a la fase inicial per establir una jerarquia i ordre de continguts, però sense molt de detall.
- **Wireframing:** Amb aquesta tècnica ja es dibuixa amb més detalls. Apareixen els elements que inicien accions, la disposició física dels elements a les pantalles (es pot fer en paper o en digital). En aquesta fase ja s'envia al client un esborrany i el client el valida aportant comentaris i possibles canvis de disseny (disposició d'element, ordre de pantalles o processos, ...)
- **Prototipat:** S'utilitza per arribar ja a un disseny final al que les diferents pantalles ja interactuen entre si (únicament digital). Aquest ja és un pas final de disseny, ja que no s'avalua únicament el disseny i l'ordre i organització dels elements, sinó també la interacció amb l'usuari (menús, botons, formularis, icones, desplegable, llistes, ...). Aquest prototip s'utilitza també per a fer proves reals amb usuaris i així arribar la producte final amb major seguretat. En aquesta fase es necessitaran diferents versions/prototips de l'aplicació fins arribar al producte final. Podem trobar 3 classes de prototips basats en la seua funcionalitat:
 - a) Horitzontal: Modela moltes característiques generals (nombre de pantalles, opcions bàsiques, ...) de l'aplicació però sense entra en detall. S'utilitza a les primeres etapes de disseny.
 - b) Vertical: Amb poques característiques, però amb molt més de detall en algunes d'elles. Per exemple un prototip que mostra amb molt detall les formes

d'identificació a l'aplicació però entra en detall a la resta de característiques o pantalles.

- c) Diagonal: És una barreja dels dos anteriors. Fins a un cert nivell presenta les característiques i a partir d'aquest mostra les de tipus vertical.

4.2. Aplicacions per al desenvolupament d'interfícies.

El disseny del prototip, en qualsevol de les seues fases, s'ha de basar en els següents aspectes:

- Identificar els elements que formen de cada una de les pantalles o finestres de l'aplicació.
- Distribuir els elements en la interfície sense que provoquen saturació, però amb suficient informació i la interacció amb l'usuari siga correcta.
- Organitzar la jerarquia d'element, el seu ordre i la disposició dels mateixos.
- Un disseny adequat per tal d'aprofitar correctament l'espai de pantalla en funció del dispositiu.
- Patrons de disseny per tal que l'estandardització d'interfícies siga viable.
- Mantindre els aspectes tècnics d'usabilitat i accessibilitat.



A les interfícies gràfiques sempre es possible diferenciar de forma clara dues àrees. L'àrea comuna a totes les finestres/pantalles de l'aplicació (informació de l'usuari connectat, botó de desconnexió, ...) i d'altra banda el contingut que varia entre finestres/pantalles.




Dins del disseny d'interfícies existeix el concepte de **jerarquia visual**. Aquest concepte es refereix a la disposició d'elements dins del disseny, la qual es molt important. Cal tindre un criteri a l'hora de situar els diferents elements a la pantalla. El criteri habitual es col·locar els elements de esquerra a dreta i de dalt a baix, establint així una jerarquia de major a menor importància (el més important va dalt a l'esquerra i van perdent importància els elements que queden baix a la dreta)

El més habitual al disseny d'interfícies es utilitzar eines basades en components visuals. Aquestes eines tenen nombroses avantatges (son eines WYSIWYG, permeten l'alineació de components de forma fàcil, arrossegar components, ...), però una de les més importants es la possibilitat de empaquetar els components creats per que puguin ser reutilitzats en altres desenvolupaments. Finalment el que estén fent utilitzant aquesta característica es reutilitzar codi, aspecte molt important en la creació de noves aplicacions.

A la actualitat existeixen un munt d'eines per desenrotllar interfícies. Les més conegudes i utilitzades son:

- **Visual Studio.** Una de les característiques més rellevants d'aquest IDE es que suporta l'ús de diferents llenguatges. En aquest IDE es poden escriure aplicacions en C#, F#, Razor, HTML5, CSS, JavaScript, Typescript, XAML i XML. Aquest entorn (al igual que la majoria d'ells) inclou les funcionalitats de autocompletat de codi (s'escriu més ràpid i s'eviten errades de sintaxi), execució de codi pas a pas (es poden crear punts d'interrupció, veure l'estat de variables i col·leccions, ...). També suporta l'ús de repositoris com GIT, Github i Azure DevOps. L'ús d'aquests repositoris està molt extensa ja que facilita el control dels canvis al desenvolupament de les aplicacions, les còpies de seguretat i el treball col·laboratiu (més d'una persona participa al desenvolupament de l'aplicació).
- **Monodevelop.** Este IDE es libre y gratuito, incorpora todas las funcionalidades propias de un editor de texto además de otras que permiten depurar y gestionar proyectos. Entre sus principales ventajas se encuentran que permite trabajar con algunos de los lenguajes más demandados en la actualidad, como son C#, Java, .NET y Python. Pertenece a Unity, motor de videojuegos multiplataforma por excelencia.
- **Glade.** Permite la creación de interfaces gráficas de usuario. Es muy utilizada en entornos XML. Permite el desarrollo de interfaces gráficas basadas en lenguaje C, C++, C#, Java, Python.
- **NetBeans.** Herramienta de código abierto. Se trata de uno de los entornos de desarrollo más utilizados en cuanto al desarrollo de interfaces a través de lenguaje de programación en Java. Este IDE permite extender el entorno con un gran número de módulos que agrupan clases de Java que permiten interactuar con las API de NetBeans.
- **Eclipse.** Entorno de desarrollo de código abierto y multiplataforma. Dispone de la funcionalidad Graphical Layout que nos permite visualizar el contenido en vista de diseño y desarrollar componentes visuales de una forma rápida e intuitiva. Cabe destacar su componente Palette, un panel que permite crear botones, cuadros de texto, cuadrículas, insertar imágenes.

Eina	Tipus de llicència	Llenguatges suportats	Lloc web oficial
 Visual Studio	Propietària (existeix una versió amb limitacions)	C#, HTML, Javascript, XML	https://visualstudio.microsoft.com
	Lliure	C#, Java, .NET, Python	https://www.monodevelop.com/

Monodevelop			
 Glade	Lliure	C++, C#, Java, Python	https://glade.gnome.org/
 Eclipse	Lliure	Java, HTML, PHP, Python	https://www.eclipse.org/eclipse/
 Netbeans	Lliure	Java, C++, Python	https://netbeans.apache.org/

Comparativa de les eines de creació d'interfícies

5. Recomanacions per al disseny d'una interfície

Els usuaris volen interfícies que cobrisquen les seues necessitats i els faciliten la informació que necessiten. Volen tindre la sensació de conduir l'ordinador, no que l'ordinador els conduísca a ells, en tot cas que els ajuden.

La principal recomanació a tindre en compte és **crear l'aplicació des d'un punt de vista d'usuari**, és a dir, pensar en el maneig de l'aplicació i la seua funcionalitat per a albirar l'organització dels elements o controls en la pantalla.

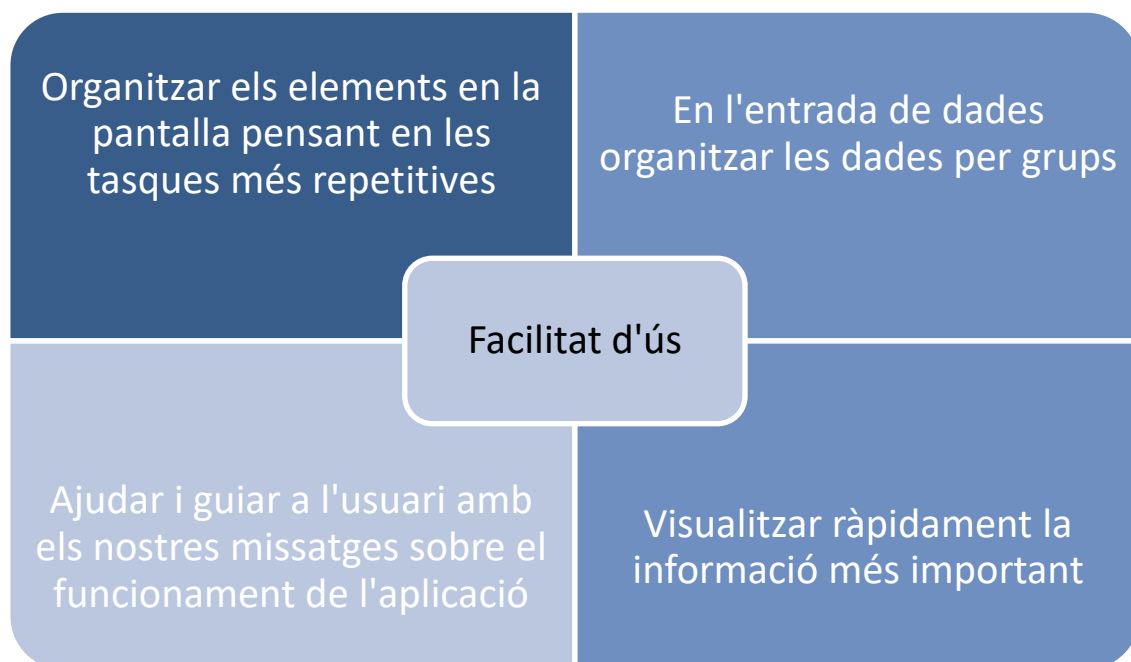
Per a aconseguir aquesta sensació existeixen les següents normes genèriques per a la realització de les interfícies gràfiques. A continuació es descriuen algunes d'elles:

- **Donar el control a l'usuari.** Definir les maneres d'interacció de manera que no oblige al fet que l'usuari realitze accions innecessàries i no desitjades. Per exemple, si en un processador de textos se selecciona el corrector ortogràfic, l'usuari romandrà en aquesta manera si desitja continuar i tindrà la possibilitat d'entrar i eixir de la manera fàcilment.
- **Ocultar a l'usuari els detalls tècnics.** L'usuari no té perquè conèixer les funcions internes del sistema com la gestió d'arxius o la gestió de memòria.
- **Llegibilitat.** Perquè l'IU afavorisca la usabilitat del sistema de programari, la informació que s'exhibisca en ella ha de ser fàcil de situar i llegir. Per a aconseguir obtenir aquest resultat s'han de tindre en compte algunes consideracions com: el text que aparega en l'IU hauria de tindre un alt contrast, s'ha d'utilitzar combinacions de colors com el text en negre sobre fons blanc o groc suau. La grandària de les fonts ha de ser prou gran com per a poder ser llegit en monitors estàndard. S'ha de ressaltar allò que és important, com per exemple una dada mal introduïda o ressaltar els números negatius en roig.

- **Permetre que la interacció de l'usuari es pugui interrompre i desfer**, és a dir, que les accions siguin reversibles.
- **Reduir la càrrega memorística de l'usuari**, com més haja de recordar un usuari, més propensa a errors serà la seua interacció amb el sistema. Aquesta és la raó per la qual una interfície d'usuari ben dissenyada no posarà a prova la memòria de l'usuari.
- **Establir valors per defecte útils**. Evita canvis innecessaris si els valors per defecte són els que més s'utilitzen.
- **Agrupar en panells les dades que tenen una característica comuna**. Per exemple, si en un formulari es demana informació d'una persona podem distingir entre la informació personal i professional, cada part estaria separada en panells diferents.
- **Adaptar-se a usuaris amb diferents nivells d'habilitat**. La nostra aplicació serà utilitzada per diferents persones amb un grau de coneixement sobre la mateixa dispar, per això haurà d'adaptar-se al grau de coneixement de l'usuari. Per a usuaris neòfits resulta interessant la utilització d'assistents que ens ajuden a fer les tasques més comunes, mentre que per als usuaris més experts podem implementar dreceres de teclat o barres d'eines que simplifiquen l'execució de les tasques.
- **Construir una interfície consistent**: La interfície haurà de presentar la informació de manera consistent, això implica que tota la informació visual estiga organitzada d'acord amb el disseny estàndard que es manté en totes les presentacions de pantalles de l'entorn gràfic que s'haja triat per a l'aplicació. Mantindre la continuïtat dins d'i entre productes. La gent porta malament el canvi radical de les interfícies Quanta gent utilitza Windows XP perquè diu que no s'aclareix amb Windows 7?
- **El temps de resposta ha de ser** xicotet i en cas d'operacions complexes mostrar algun tipus de progrés per a informar l'usuari.
- **Ajuda**: disposar de facilitats d'ajuda en línia que faciliten l'ús del programa. Es tracta d'una de les tasques que més treball porten però que l'usuari agraeix més al principi.
- **Entrada de dades**: és important que es minimitze el nombre d'accions que l'usuari ha de realitzar. A més s'ha de mantindre un ordre lògic en l'entrada de dades, és a dir, que el desplaçament del focus després de completar un camp tinga cert sentit i ordre i no es desplace de manera aleatòria per tot el formulari.
- **Visualització de dades**: utilitzar un format de dades que permeti una assimilació ràpida de la informació i que no varie d'un element a un altre.
- **Validació de l'entrada de dades de l'usuari**. Resulta convenient validar molts dels camps del formulari d'introducció per a evitar errors no contemplats en el programa. Un de les principals maneres d'atacar un programa per part d'un hacker és precisament aquesta, és a dir, buscar una introducció de dades que provoquen un desbordament de pila i ens permeten accedir al sistema en manera supervisor.

- **Maneig d'errors i visualització de missatges d'error.** És fonamental l'ús d'excepcions en la programació actual, si es produeix un error per una cosa no contemplada en el programa o per una fallada en el sistema, s'haurà de capturar aqueixa excepció i facilitar la seua resolució. **Quina imatge dona un programa que es tanca en produir-se un error?**
- **Gestionar correctament els missatges d'error, facilitant a l'usuari la seua lectura i comprensió:** referència explícita del problema i suggeriment de solucions. S'ha d'evitar donar informació tècnica sobre els errors, els missatges han de ser clars i apuntar a la resolució d'aquest.
- **Logs.** A l'usuari no s'ha de donar informació tècnica, no obstant això a nosaltres ens resulta de vital importància tindre informació sobre l'error, ja que d'aquesta manera podrem tindre un punt de partida per a resoldre'l. Per això resulta de vital importància en el maneig de les excepcions la utilització d'un sistema de logs que guarden la informació rellevant d'una aplicació, similar a un visor de successos de Windows.
- **Documentació:** comentar el codi i documentar les solucions a determinats problemes resulta de vital importància, ja que quan es realitza una aplicació ens recordem de les coses, però què succeeix 6 mesos després, com diu el refrany Més val un llapis curt que una memòria llarga.

A continuació es mostra una figura que ens ajudarà a millorar les nostres interfícies gràfiques:



5.1. Llei de Fitts.

En l'actualitat, i més amb l'aparició dels dispositius tàctils, es fa molt recalcament en la grandària de les etiquetes o botons que conformen una interfície GUI. Amb l'aparició dels dispositius mòbils tàctils i les seues reduïdes dimensions aquesta llei ha cobrat especial importància.

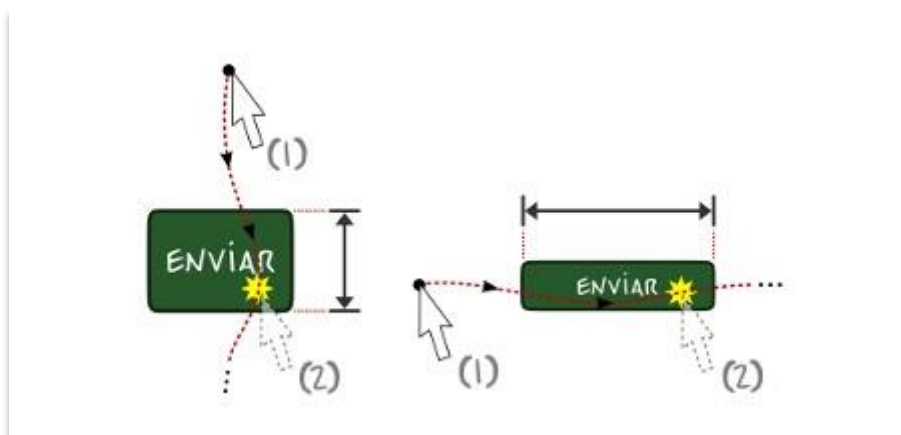
La llei de Fitts és un model del moviment humà, que prediu el temps necessari per a moure's ràpidament des d'una posició inicial fins a una zona destí final com una funció de la distància fins a l'objectiu i la grandària d'aquest.

Es tracta d'un model matemàtic que ens ajuda a quantificar la dificultat que comporta l'ús d'una interfície, ja siga en diferents sectors de la població (joves, ancians, ...) o en condicions físiques diferents com puga ser sota de l'aigua.

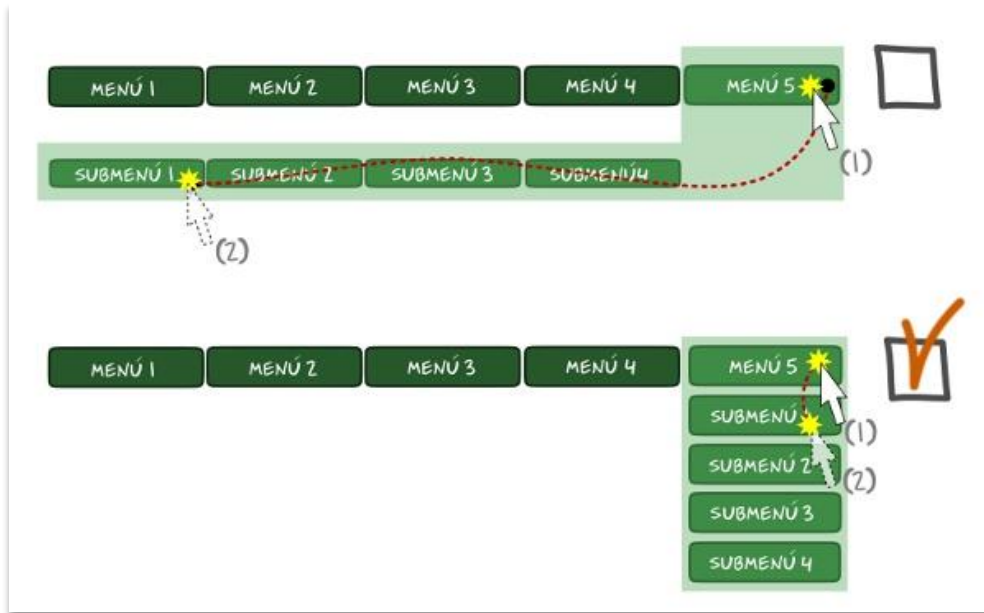
La llei de Fitts realment ens diu una cosa obvia: *"el temps necessari per a aconseguir un enllaç amb el punter és una funció de la distància i de l'ample de l'enllaç al llarg de l'eix de desplaçament del cursor"*. Més simple encara: si l'usuari ha de recórrer un espai en la pantalla amb el punter per a fer clic en un botó, **com més lluny estiga el punt d'origen i més xicotet siga el botó, més temps necessitarà**.

Tan sols grans companyies, com Apple per als seus productes, poden permetre's valorar les seues interfícies amb aquesta llei, però sí que ens pot ajudar a millorar les nostres interfícies amb alguns consells:

- Les accions més habituals o més importants per als usuaris han d'estar el més a prop possible de la posició prèvia del cursor i tindre una superfície clicable major. Exemple comú de mala pràctica: en completar l'últim camp en un procés de registre, ens trobem que el botó enviar o acceptar està més lluny d'aqueix punt que altres botons del formulari i la seua grandària és exactament el mateix (malament!).
- Si augmentem la grandària dels enllaços, hem de tindre en compte quina és la direcció de moviment del cursor més probable quan aquests s'usen, ja que el que compte és el recorregut útil sobre l'eix de desplaçament del cursor. Si el moviment és predominantment vertical, per exemple, serà l'altura del botó, i no el seu ample, la mesura a potenciar.



- En els menús jeràrquics, assegura't que les opcions que apareixen queden prop de la posició original i que no es canvia la direcció del desplaçament.



- Els menús popup o contextuais poden ser usats més ràpidament que els pull-down, en estalviar desplaçament l'usuari.
- La fórmula ens diu que cada increment en la grandària de l'objecte comporta una millora progressivament menor (un matemàtic ens diria que això es deu a la naturalesa logarítmica de la funció). La traducció és que fer enllaços grans ajuda, però **fer-los enormes no aporta cap benefici addicional**.
- Potser no pots (o no vulgues) fer més gran un objecte, però possiblement sí que pots **situar-lo més a prop**.

Podem veure una xicoteta demostració de la llei de Fitts en el següent en la web <http://fww.few.vu.nl/hci/interactive/fitts/>

6. El color

L'ull humà sol és capaç de percebre els denominats colors additius; a través de la combinació d'aquests li és possible obtenir la resta dels colors, això són: blau (B), vermell (R) i verda (G). Per tant és important tindre en compte els color emprats als dissenys, **una aplicació massa "colorida" no serà agradable a l'usuari**. També pot ser interessant triar els colors adequats en funció del tipus o sector de l'aplicació.

A continuació analitzarem en què consisteix el sistema de representació RGB, així com les propietats principals del color, que modifiquen i redefeixen el sistema de color base.

6.1. El Sistema RGB

De la mateixa forma, un ordinador serà capaç d'obtenir la representació de tots els colors utilitzant el Sistema RGB, o cosa que és el mateix, el Sistema Xarxa-Green-Blue. Indicant la

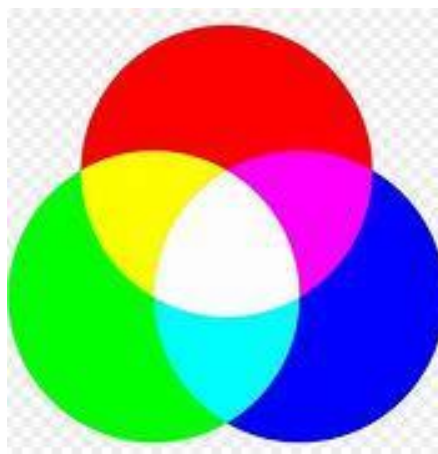
proporció de cadascun d'ells dins de la combinació d'aquests tres, donarà lloc a tota la paleta de colors coneguda.

Per a representar cada color de manera que pugui ser traduït per l'ordinador s'utilitzen 8 bits per a codificar cadascun dels colors additius, és a dir, s'estableix la proporció de cada color que formarà part de la combinació de tres.

L'escala monocromàtica d'un color tindrà 256 (2 elevat a 8) valors.

A l'hora de representar cadascun dels colors, és possible utilitzar tant el sistema de numeració decimal (0 a 255) com l'hexadecimal, on cadascun dels dígit es codifica amb 8 bits binaris que, agrupats en blocs de 4 bits, ens retorna el valor corresponent en hexadecimal.

El nombre de combinacions de colors es calcula multiplicant el nombre màxim de graus en l'escala monocromàtica de cada color, $256 \times 256 \times 256$, la qual cosa ens dona 16 777 216 colors.



Cercle de colors additius al Sistema RGB

Per exemple: El color groc estaria format així:

Color	Decimal	Binari	Hexadecimal
Roig	255	1111 1111	FF
Verd	255	1111 1111	FF
Blau	0	0000 0000	00

D'aquesta forma el color groc quedaria expressat en hexadecimal com **FFFF00**. Fixat que al dibuix anterior el color groc està únicament format pels colors roig i blau.

Es important conèixer aquesta forma d'interpretar els colors, es la que utilitzarem per a donar color a qualsevol component d'una interfície.

6.2. Matís, saturació i lluentor

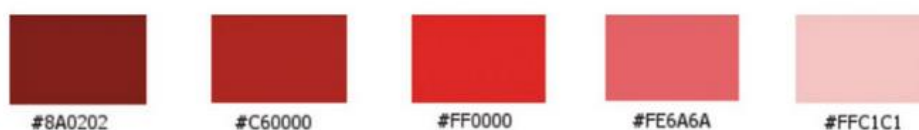
A més del grau en l'escala monocromàtica de cadascun dels colors del sistema RGB, els colors presenten tres propietats que permeten distingir-se els uns dels altres, aquestes són: el matís, la saturació i la lluentor. Aquestes propietats ens permeten definir els colors com a cromàtics, complementaris o pròxims, així com definir el contrast de color:

- **Matís:** Atribut que permet distingir un color d'un altre. Els tres matisos primaris són els colors additius, verd, vermell i blau; la resta de colors s'obté barrejant aquests tres. El matís permet definir dos colors com a complementaris quan està un enfront de l'altre en el cercle cromàtic.
- **Saturació:** Aquest atribut defineix la intensitat d'un color. Pot relacionar-se amb l'amplada de banda de llum que estem visualitzant, per tant, queda condicionat pel nivell de gris present en un color: com més gran sigui el nivell de gris, menys saturat serà un color, i serà menys intens.
- **Lluentor:** Atribut que defineix la quantitat de llum d'un color. Representa el fosc (si se li afegeix negre) o clar (si se li afegeix blanc) que és un color respecte del seu patró, és a dir, respecte del color pur sense modificar la lluentor. En una composició de colors en disseny gràfic, com més brillant sigui un color, més a prop sembla estar.



Cercle cromàtic de matisos

Observa a les següents imatges del color roig. Fixat com canvia el pes dels colors verd i blau i com afecten al color roig final.



Escala de color amb matís roig canviant els seus valors saturació

7. Patrons de disseny

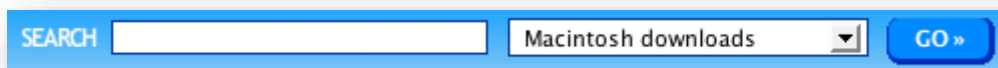
Els consells anteriors són molt importants i cal tindre'ls molt en compte a l'hora de realitzar les interfícies gràfiques, no obstant això queden molt abstractes i de primeres no s'entenen molt bé.

Els patrons de disseny s'utilitzen en programació en multitud de situacions i bàsicament són **l'esquelet de les solucions a problemes comuns en el desenvolupament de programari**.

En altres paraules, brinden una solució ja provada i documentada a problemes de desenvolupament de programari que estan subjectes a contextos similars.

Existeixen multitud de patrons per a problemes diferents, però ens centrarem en els anomenats patrons GUI que ens ajudaran a resoldre de manera efectiva situacions comunes a l'hora de programar aplicacions gràfiques.

Bàsicament aquest tipus de patrons ens ajuden a visualitzar la forma en la qual s'han resolt diferents funcionalitats comunes a molts programes. Per exemple, en la nostra aplicació realitzarem una cerca de registres en una base de dades. Podem implementar-ho de diferents maneres però podem seguir un patró com el següent:



D'aquesta manera podem millorar les nostres aplicacions mitjançant exemples de coses que ja s'han fet i funcionen, sense comptar amb la facilitat d'ús perquè els usuaris ja estan acostumats a això.

En el següent lloc web podem trobar patrons GUI per a diferents funcionalitats: <http://www.welie.com/patterns/showpattern.php?patternid=search>

7.1. Patró Model-Vista-Controlador

El desenvolupament d'aplicacions gràfiques presenta diverses coincidències, bé siguin d'escriptori o basades en web, que ens permeten dissenyar un model de comportament de les mateixes de manera genèrica.

És a dir, el que es pretén és tindre unes directrius de disseny que ens permeten agilitar la creació de les nostres aplicacions.

El patró més estès per a la realització d'interfícies gràfiques és **model-vista-controlador** que pretén la separació de la presentació, de les dades i dels esdeveniments que es produeixen.

Aquest model consta de tres components:

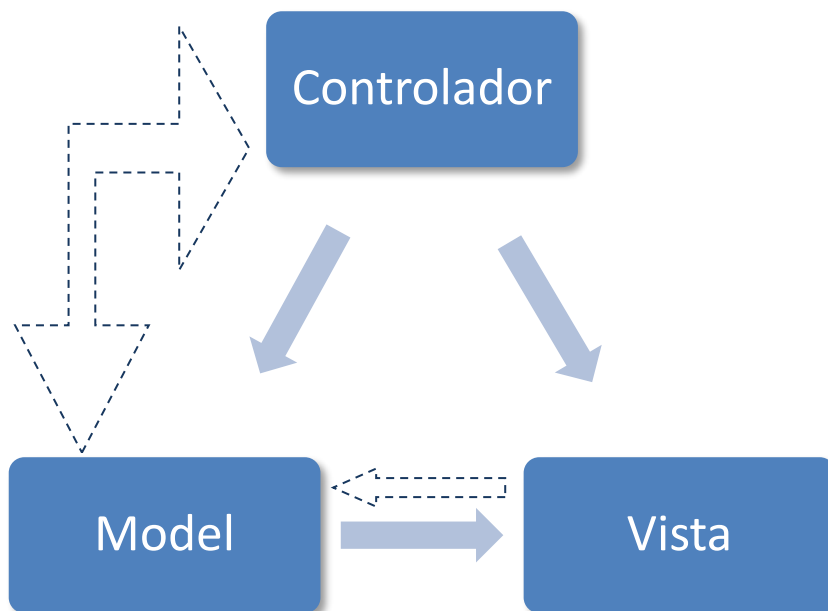
- **Model:** és l'encarregat d'emmagatzemar les dades, normalment resideix en memòria, sinó tots almenys una part.

- **Vista:** mostra les dades del model, emmagatzemat en memòria, a l'usuari final.
- **Controlador:** és l'encarregat de gestionar les relacions entre el model i la vista. Maneja els esdeveniments que es produeixen en la vista i actualitza el model en funció d'ells.

Posarem un exemple de funcionament per a comprendre millor el model:

Suposem que hem desenvolupat un programa que conté una taula on visualitzem les dades d'una sentència SQL que hem realitzat. **El model conté les dades en memòria i la vista els mostra a l'usuari.** L'usuari decideix canviar una dada que considera erroni, després de fer-lo pitja el botó d'Acceptar , això desencadena un esdeveniment en el sistema, el controlador el rep i indica al model que ha de canviar una dada.

D'aquesta manera funciona el model MVC, a continuació es mostra una figura que explica:



Avantatges

- Clara separació entre interfície, lògica de negoci i de presentació, que a més provoca part dels avantatges següents.
- Senzillesa per a crear diferents representacions de les mateixes dades.
- Facilitat per a la realització de proves unitàries dels components.
- Reutilització dels components.
- Simplicitat en el manteniment dels sistemes.
- Facilitat per a desenvolupar prototips ràpids.
- Els desenvolupaments solen ser més escalables, és a dir, són fàcilment ampliables.

Desavantatges

- Haver de cenyir-se a una estructura predefinida, la qual cosa a vegades pot incrementar la complexitat del sistema. Hi ha problemes que són més difícils de resoldre respectant el patró MVC.
- La corba d'aprenentatge per als nous desenvolupadors s'estima major que la de models més simples com a formularis web.
- La distribució de components obliga a crear i mantindre un major nombre de fitxers.

Posem un altre exemple:

- **Model:** un exemple de la vida real d'un Model seria una classe anomenada Client, la qual té les mateixes propietats d'una taula client en la meua base de dades.
- **Controlador:** un Controlador seria el Controlador Client, generalment les classes Controladores porten el sufix "Controlador", així que en el nostre cas es cridaria ClientesControlador. El controlador portaria les accions que nosaltres podem realitzar en un client com per exemple, agregar, esborrar, modificar, agregar ordre, etc.
- **Vista:** la Vista és el més fàcil d'entendre, simplement és la nostra pàgina html. A través de l'acció del Controlador especifiquem al fet que vista volem enviar el resultat de l'acció del Controlador. En alguns casos és necessari passar informació a la Vista des del Controlador, això s'aconsegueix fàcilment en el codi de l'acció.

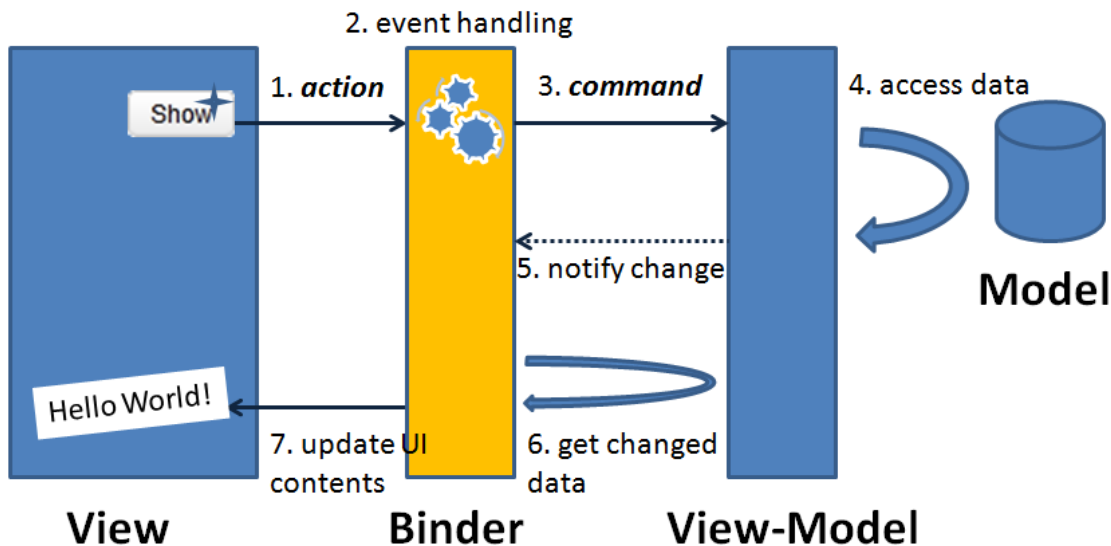
Cace destacar que la major part de les eines de desenvolupament incorporen en les classes de la vista gran part o tot el processament d'esdeveniments. Amb el que el controlador queda semi-ocult dins de la vista.

7.2. Patró Model-Vista-Model de Vista

Igual que el patró anterior el que pretén aquest és dividir l'aplicació en capes el més independents possibles per a facilitar el manteniment i la reusabilitat.

És molt similar a l'anterior, de fet es basa en ell, no obstant això afegeix una capa més per a mantindre sincronitzats les dades entre la vista i el model.

En la imatge següent es pot veure l'esquema de blocs del patró.



Com més endavant veurem en els exemples d'altres unitats, està capa fa el treball de sincronització de les dades de manera automàtica sense necessitat de fer-ho en el controlador.

- **Model:** representen les dades que manipula l'aplicació. Normalment trets de la base de dades.
- **Vista:** conté els elements de la interfície d'usuari. Defineix l'estructura, el disseny i l'aparença del que l'usuari observa en la pantalla.
- **Model de vista:** és la capa intermèdia que ofereix les dades a la vista i respon a les interaccions de la mateixa amb esdeveniments que canvien les dades. Per a aconseguir-ho utilitza la tècnica del **binding**.

8. Estàndards de facto

Com diu el refrany “*cada mestre amb el seu llibret*”, en la indústria informàtica no anava a resultar menys. Existeixen uns estàndards de facto que proporcionen diferents fabricants de programari o maquinari perquè les aplicacions segueixen els criteris anteriorment descrits.

Per exemple, Apple té la seua pròpia guia d'estil que estableix una sèrie de normes a complir per al desenvolupament d'aplicacions informàtiques.

També resulten interessants les **normes CUA** desenvolupades per IBM és un estàndard d'interfícies d'usuari per als sistemes operatius i programes informàtics. Va ser publicat per primera vegada en 1987 com a part de la seua arquitectura de l'aplicació de sistemes. El CUA conté normes per al funcionament d'elements com ara quadres de diàleg, menús i dreceres de teclat que han arribat a ser tan influent que són implementades hui dia per molts programadors que mai han llegit la CUA.

Alguns exemples de combinacions de tecles que utilitzen la majoria dels programes i sistemes operatius com pot ser invocar l'ajuda en estrényer F1, realitzar una cerca en estrényer F3 i així un llarg etcètera de funcionalitats que es donen com un estàndard van ser introduïdes per aquestes normes.

Podem trobar una descripció d'elles en la següent url [normes CUA](#)

Hi ha hagut molts exemples del que es diu **Look and Feel**, és a dir, els colors utilitzats, formes dels elements, distància mínima entre ells, layouts utilitzats i el comportament dels elements dinàmics de la interfície.

Per exemple, Apple va ser una de les primeres companyies a definir tot aquest tipus de situacions.

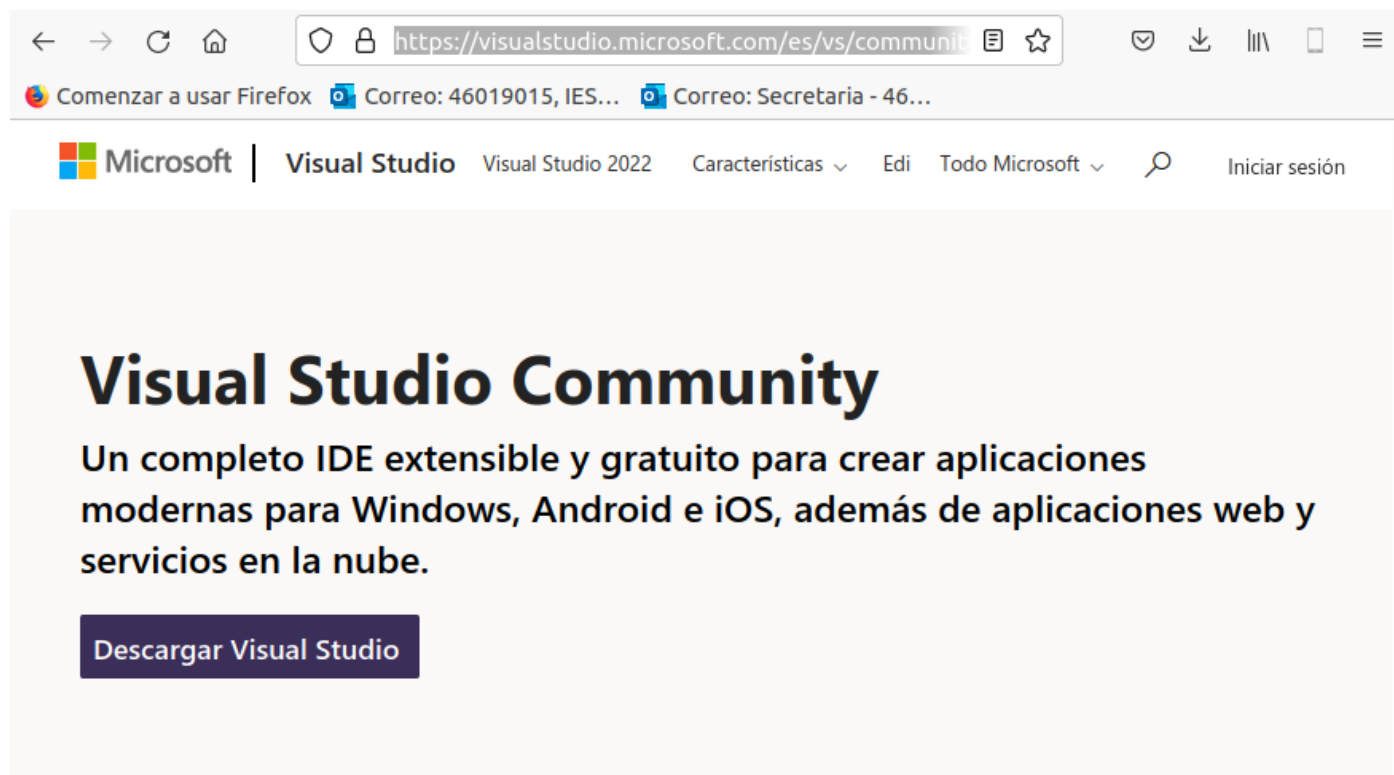
9. Instal·lació de Visual Studio Community

L'IDE que utilitzarem a l'assignatura serà Visual Studio Community (és la versió gratuïta de l'IDE) acompanyat del llenguatge C#. La seua instal·lació es molt senzilla.

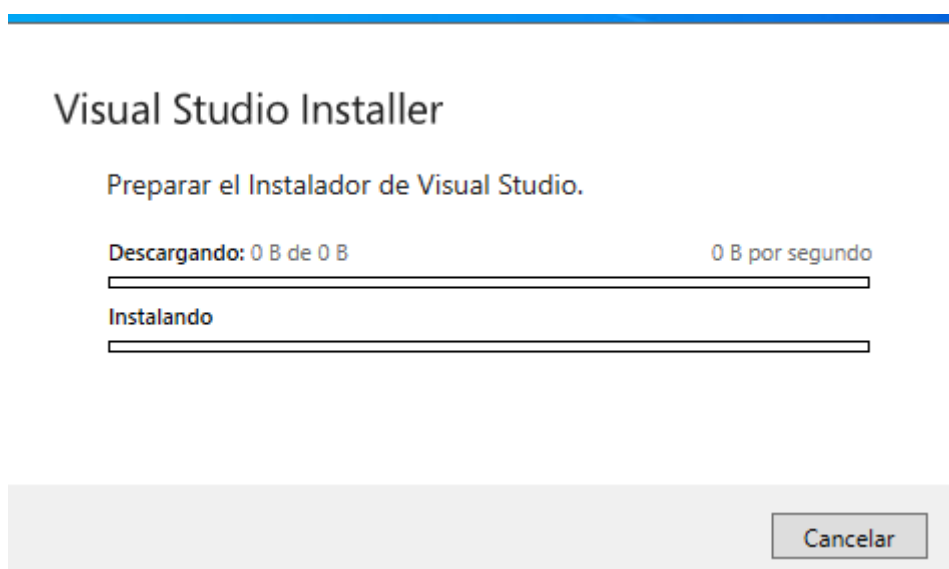
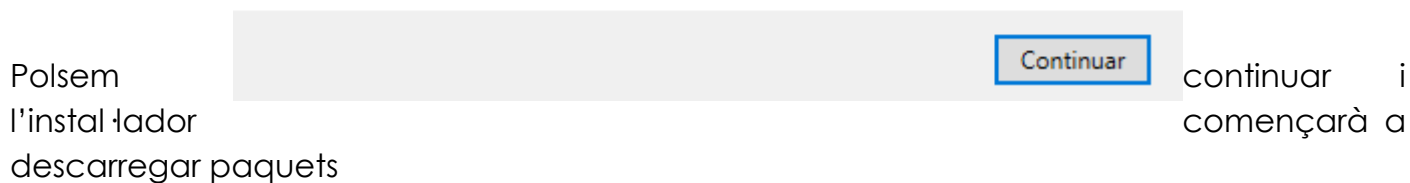
Per al mon Linux també existeix un projecte d'un IDE amb C# anomenat Monodevelop.

9.1. Instal·lació de Visual Studio Community

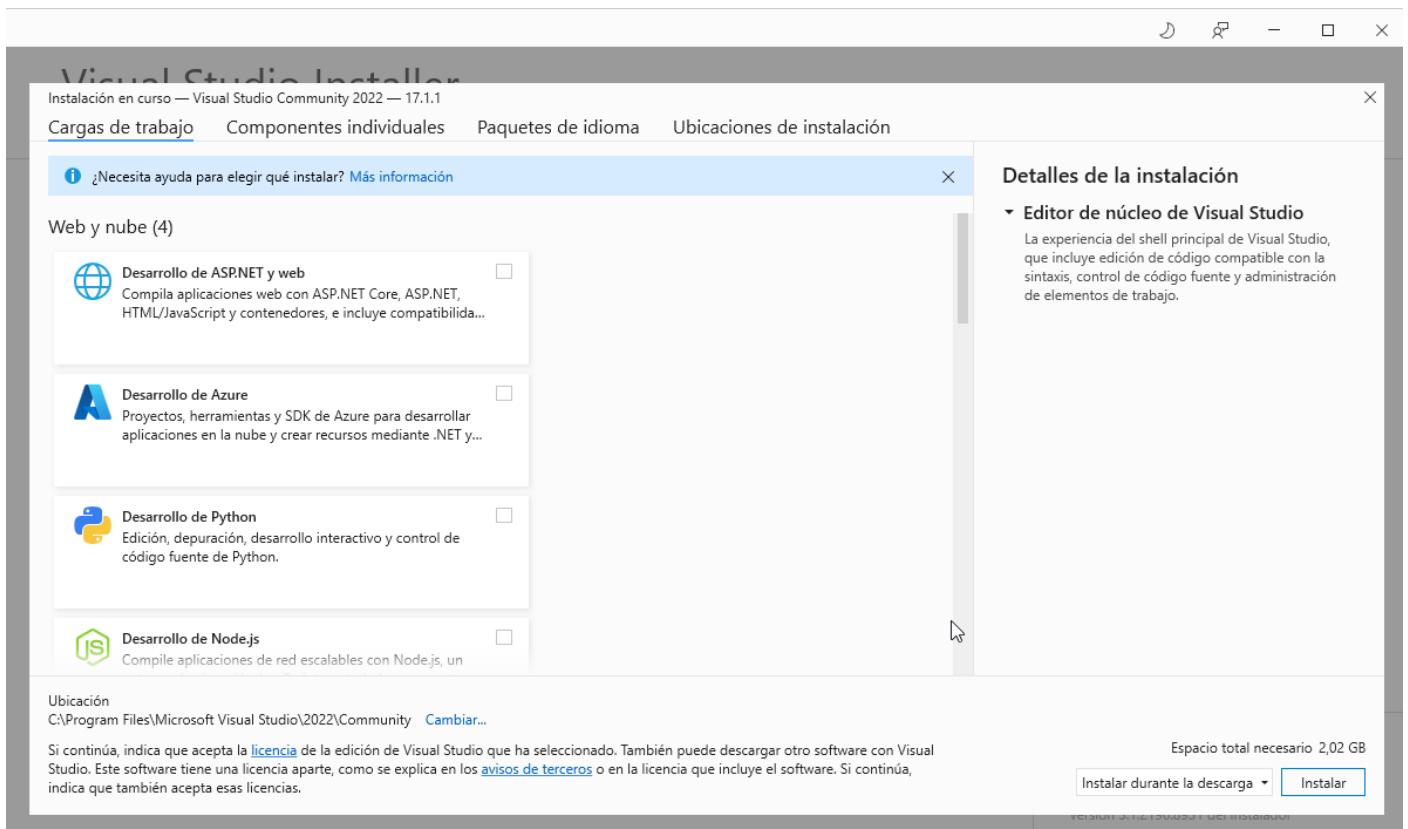
El primer seria anar a la web oficial de l'IDE <https://visualstudio.microsoft.com/es/vs/community/> i polsem el botó “**Descargar Visual Studio**”



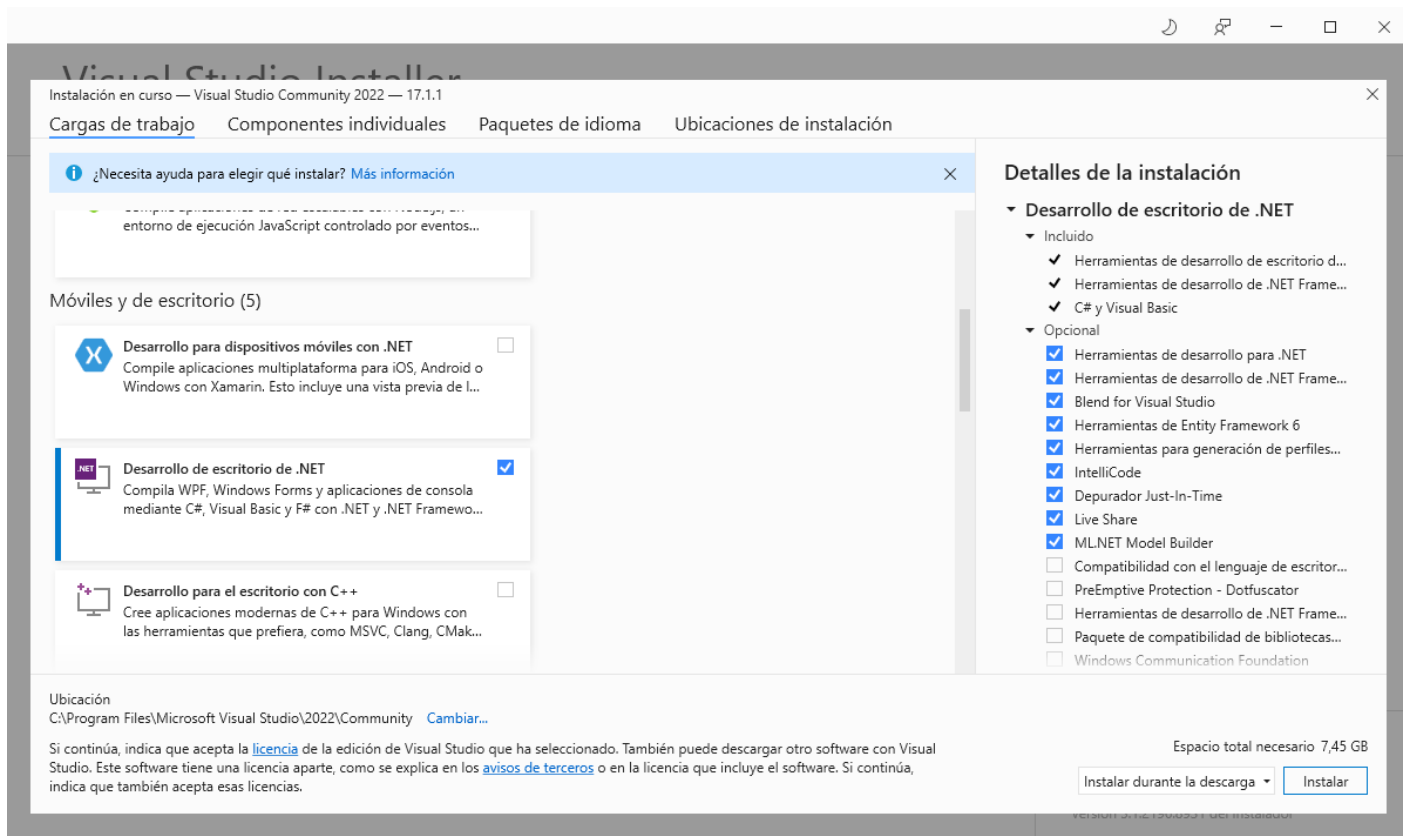
Açó descarregarà una fitxer anomenat **VisualStudioSetup.exe**. A continuació l'executem:



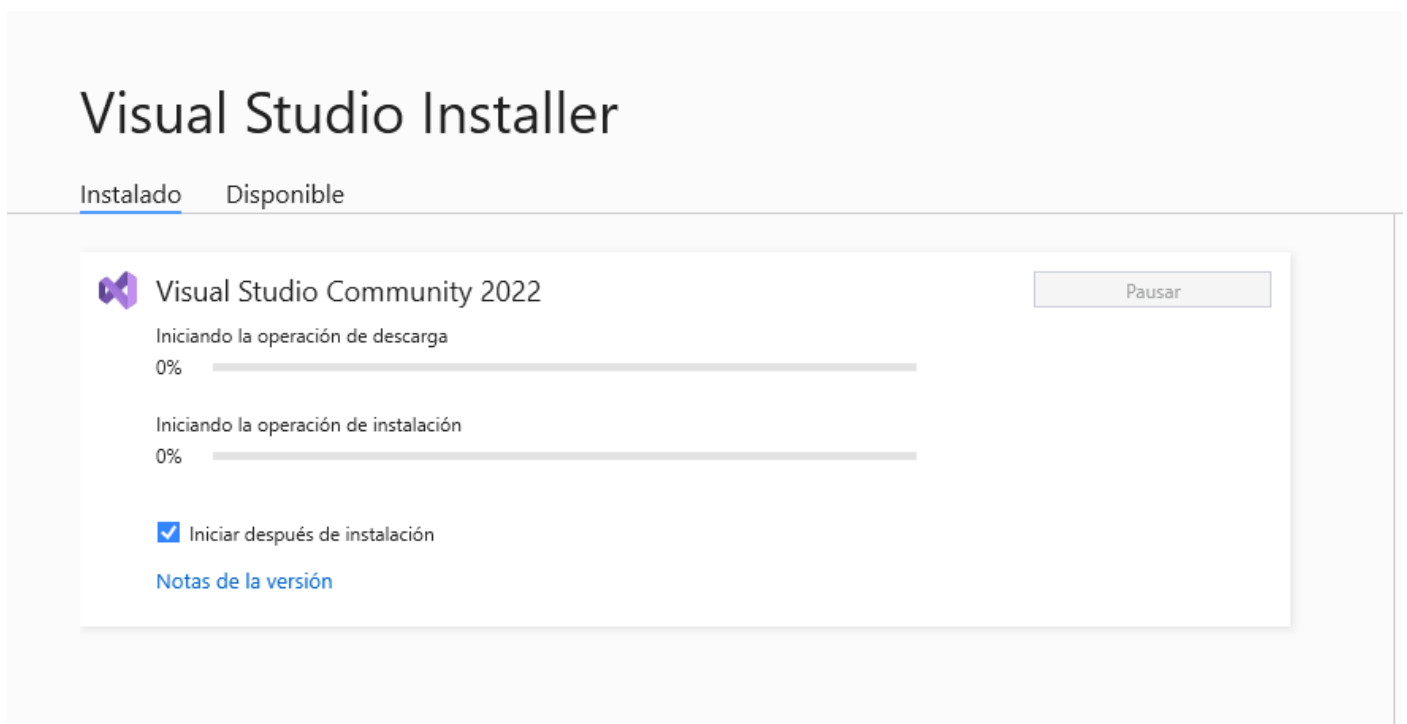
En acabar ens mostrarà les opcions d'instal·lació:



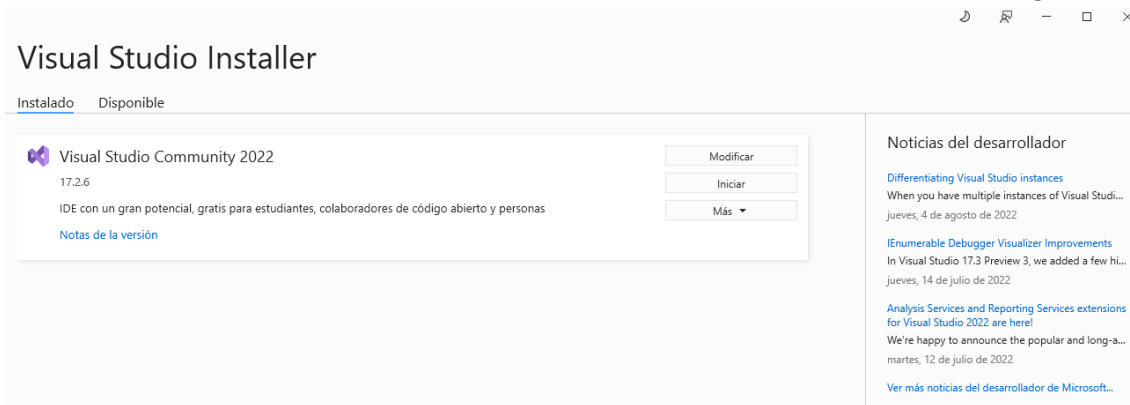
Per al nostre cas únicament haurem de triar baixar un poc i triar el desenvolupament d'aplicacions d'escriptori .NET:



Polsem instal·lar per a comence la instal·lació (observa l'espai requerit per a completar la instal·lació):



En acabar obtindrem la següent finestra:

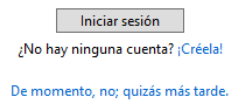


Si polsem “Iniciar” s’obrirà l’eina

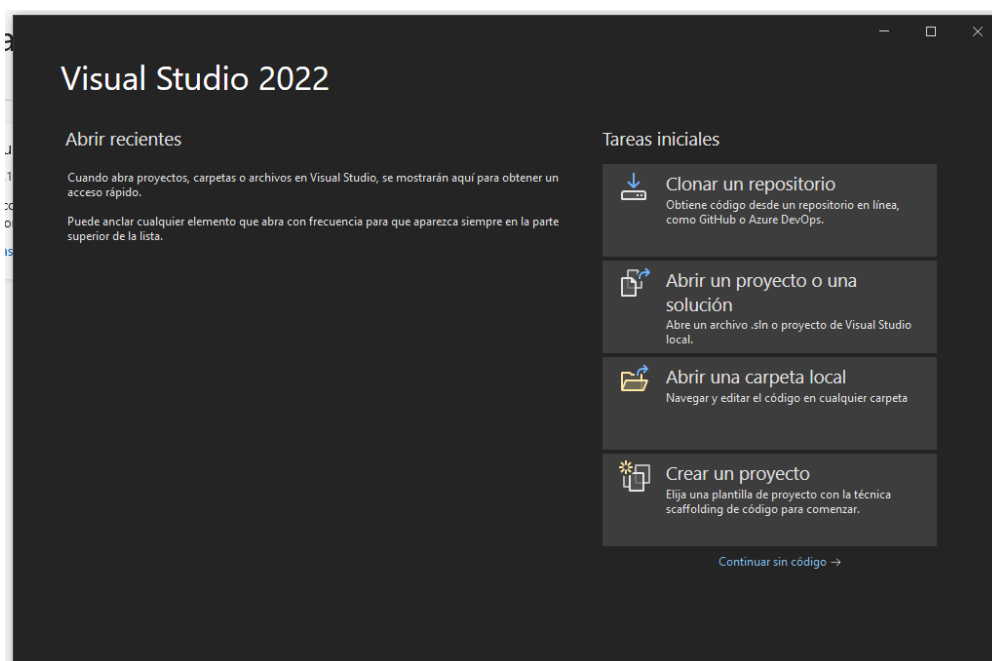
Visual Studio

Iniciar sesión en Visual Studio

- Sincronizar configuración entre dispositivos
- Colaborar en tiempo real con LiveShare
- Integrar sin problemas con los servicios de Azure



Polsem en “De moment no, pot ser més avant” per iniciar l’eina sense iniciar sessió amb un compte Microsoft



Podem tancar-la, més avant vorem com crear nostre primer projecte

9.2. Creació del primer projecte

Per a crear el primer projecte farem ús de les interfícies WPF de Microsoft.

Que es WPF?

WPF és l'abreviació de **Windows Presentation Foundation**. En termes de programació, és una sèrie d'assemblats i eines del framework .NET. Està destinat a proporcionar una API (Interfície de programació d'aplicacions de l'anglès Application Programming Interface) per a crear interfícies d'usuari enriquides i sofisticades per a Windows. Està suportat des de Windows XP fins a l'última versió de Windows, la versió 11.

Combina diferents plataformes de desenvolupament. Agafa coses del desenvolupament web, de les aplicacions d'Internet enriquides o RIA (de l'anglès Rich Internet Applications) i per descomptat del desenvolupament d'aplicacions per a Windows.

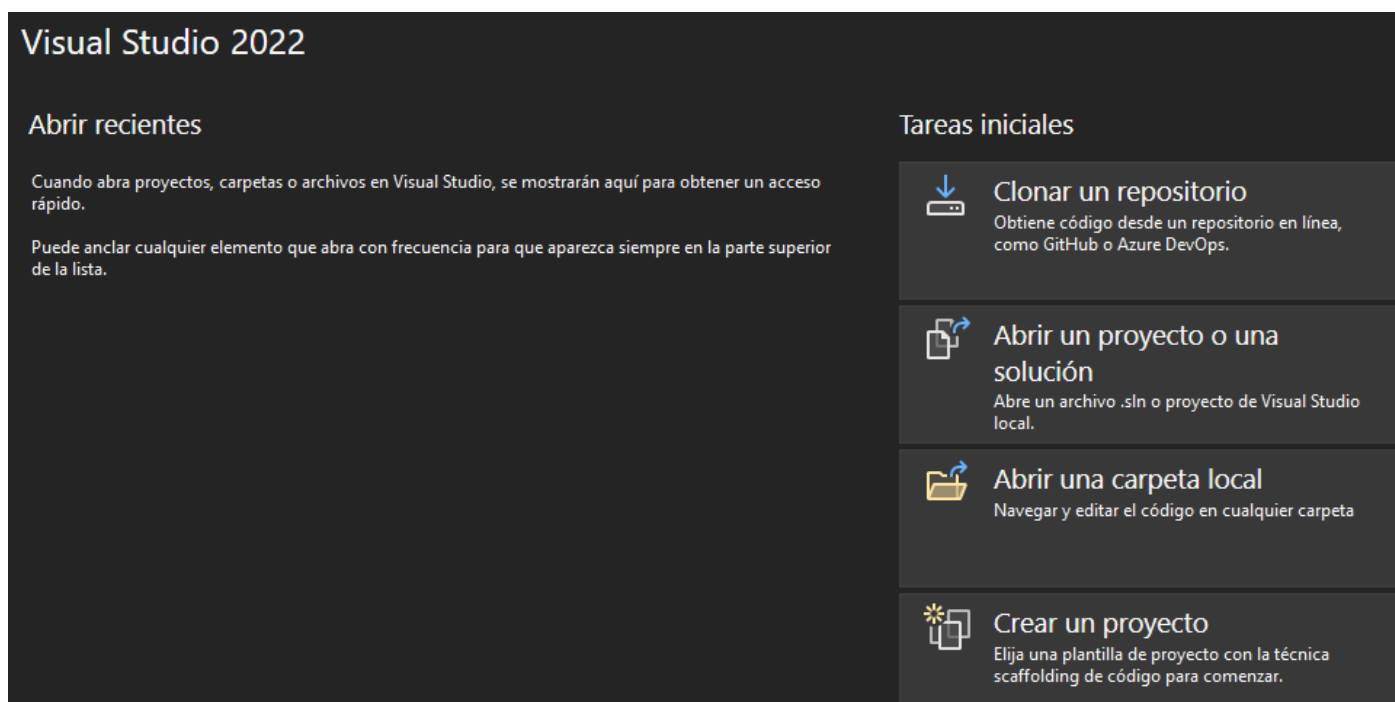
Del desenvolupament web hereta la utilització d'un llenguatge de meta etiquetes per al desenvolupament de la interfície gràfica UI i els estils (XAML). De les aplicacions RIA hereta els gràfics vectorials, animacions i el suport per a multimèdia.

A més de les aportacions d'altres plataformes, WPF incorpora noves funcionals que fan que el seu predecessor, Windows Forms, es quedi antiquat. Facetes com a suport per a 3D, tipografia avançada i documents similars al PDF.

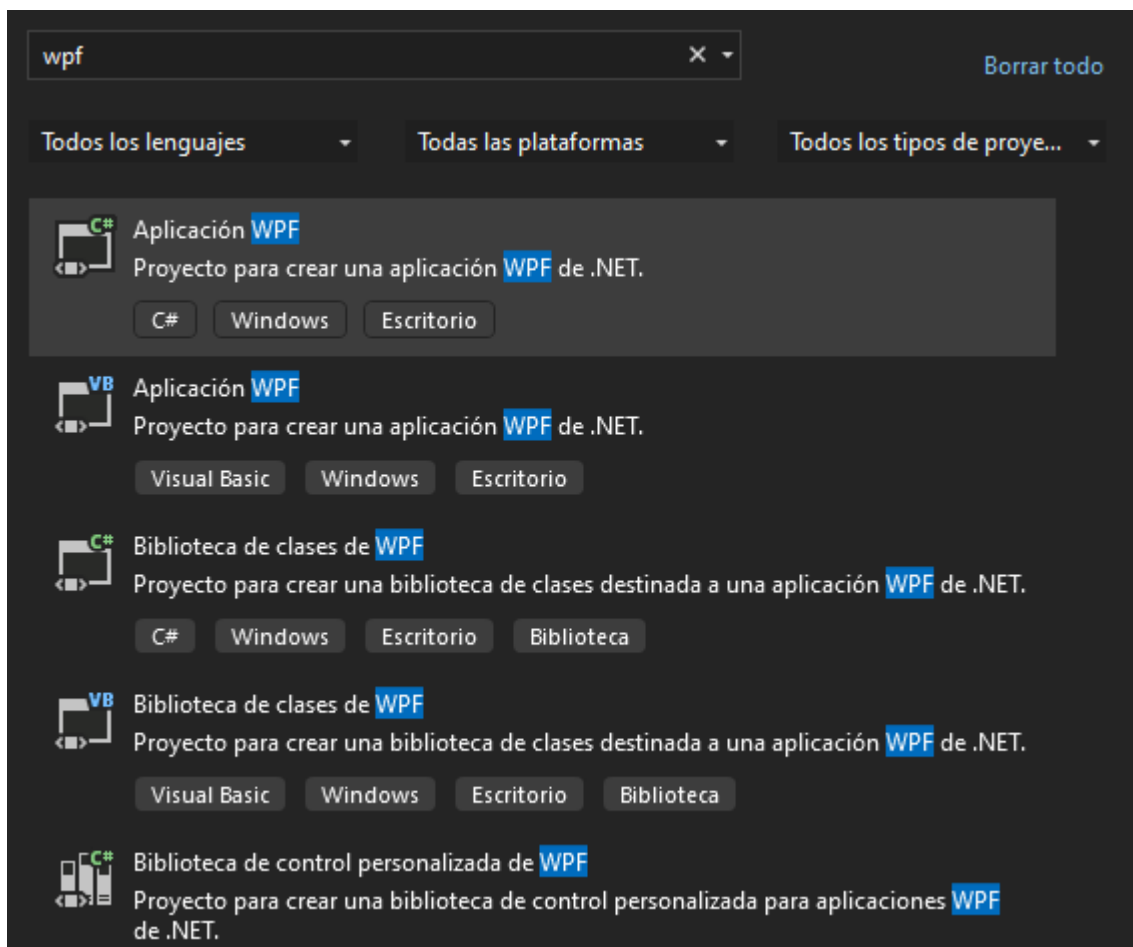
Al propera unitat parlarem més profundament de WPF i l'univers .NET, per acabar l'unitat vorem com crear un primer projecte WPF.

Mi primer projecte WPF

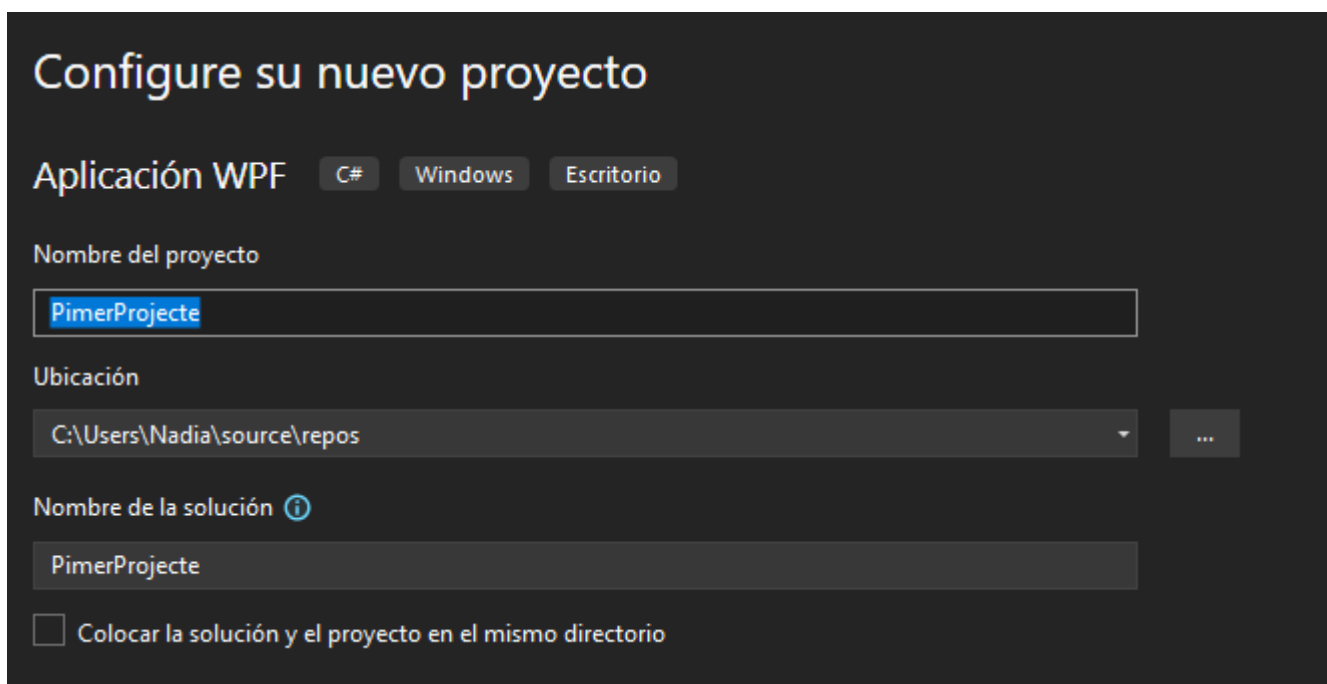
Per a crear un projecte WPF únicament hem d'iniciar el Visual Studio Community i pulsar baix a la dreta en l'opció "Crear un nou projecte":



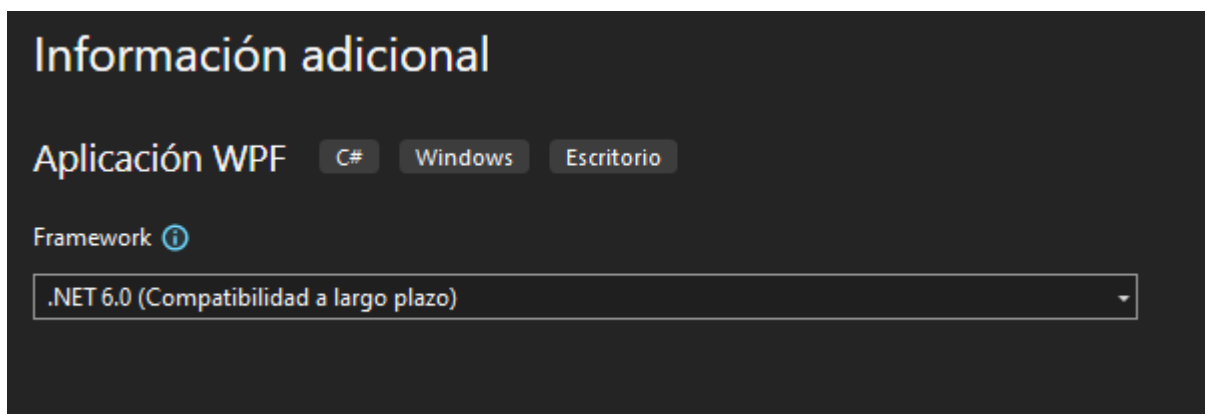
El tipus de projecte a triar serà “**Aplicació WPF**” i que incloga C# com a llenguatge suportat. Podem trobar el tipus de projecte filtrant amb la paraula “**wpf**” al requadre de recerca:



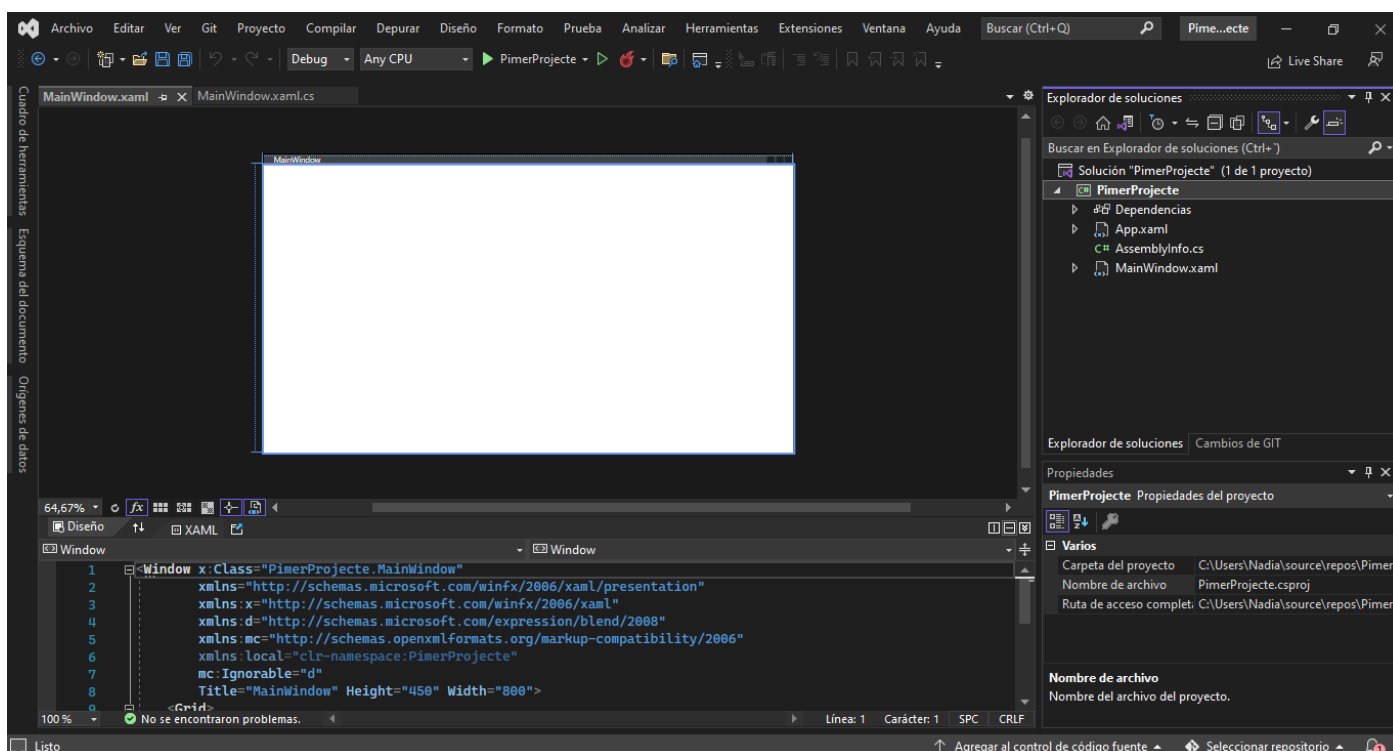
Una vegada triat el tipus de projecte polsem **Següent**



Posem nom al projecte (per exemple PrimerProyecto) i polsem **Següent**



Triem la compatibilitat amb les versions del FrameWork .NET (en principi ho deixem tal i com està) i polsem **Crear**



En la propera unitat començarem a aprendre com utilitzar l'eina Visual Studio