Aplicacions web. Característiques, funcionament i estructura

Implantació d'Aplicacions Web

Nacho Iborra
Traducció: Sergi Aracil

IES Sant Vicent

Aquesta obra està llicenciada sota la Llicència Creative Commons Atribució-NoComercial-CompartirIgual 4.0 Internacional. Per a veure una còpia d'aquesta llicència, visita
http://creativecommons.org/licenses/by-nc-sa/4.0/

Tipus d'aplicacions

Quan estem utilitzant un ordinador, una tauleta o un telèfon mòbil, quins tipus d'aplicacions o programes podem estar utilitzant? Bàsicament distingim dos grans grups:

- Aquelles aplicacions que no necessiten cap connexió a Internet o a una xarxa d'ordinadors per a funcionar. Aquest tipus d'aplicacions solen dir-se **aplicacions d'escriptori**, i podem trobar exemples molt variats: un processador de textos, un lector de llibres electrònics, un reproductor de música o vídeo, i fins i tot videojocs que tinguem instal·lats.
- Aquelles aplicacions que sí que necessiten connexió, bé siga a Internet o a un ordinador de la seua xarxa local. En aquest altre grup també tenim exemples variats d'aplicacions. Per exemple, si compartim un document de text en Google Drive, o si obrim el navegador per a accedir a una plataforma d'un curs en línia, o fins i tot si juguem a videojocs juntament amb altres persones d'altres llocs. Ací distingim diversos tipus d'aplicacions. Alguns dels més habituals són:
    - Les **aplicacions P2P** (*peer-to-peer*), on tots els elements connectats a la xarxa tenen el mateix "rang", per dir-ho així, i comparteixen informació entre ells. És el mecanisme en el qual es basen diversos programes de descàrrega, com els de descàrrega de torrents , o altres més antics com *Emule*.
    - Les **aplicacions client-servidor**, dites així perquè consisteixen en el fet que un conjunt d'ordinadors (anomenats *clients*) es connecten a un central (anomenat *servidor*) que és el que els proporciona la informació i els serveis que sol·liciten. En el cas de videojocs on ens connectem a un altre lloc per a jugar amb altres persones, estem utilitzant aplicacions client-servidor, on el nostre ordinador (un dels clients), té instal·lada una part de l'aplicació i el servidor al qual es connecta li proporciona la informació dels escenaris i de la resta de jugadors i personatges.
        - Dins de la mena d'aplicacions client-servidor, les **aplicacions web** són un subtipus, potser el més nombrós. Es diuen així perquè accedim a una pàgina web per a poder-les veure i usar. Google Drive, una plataforma d'un curs en línia, o una web de videojocs serien exemples d'aplicacions web.

Ens centrarem en aquest últim tipus d'aplicacions, les **aplicacions web**. Amb l'expansió d'Internet i de les xarxes d'ordinadors, són aplicacions en auge, i no deixen d'aparéixer noves eines i tecnologies per a desenvolupar aplicacions web cada vegada més vistoses, versàtils i adaptades a gran varietat de dispositius.

En aquesta unitat i vorem quins avantatges i inconvenients poden tindre les aplicacions web amb altres tipus d'aplicacions (en concret amb les d'escriptori), i també veurem quina arquitectura tenen, és a dir, quins mecanismes són els que les fan funcionar com funcionen. També donarem alguna pinzellada sobre quins patrons d'arquitectura programari podem utilitzar per a desenvolupar aplicacions web.

Arquitectura d'una aplicació web

Què és “el web”?

Podem veure el web com una espècie de plataforma mundial on tenim disponibles gran quantitat de recursos (documents, videojocs, xarxes socials, fòrums, etc.). Es va fer popular a principis dels anys 90 gràcies a aplicacions com el correu electrònic, els xats, etc. i amb l'aparició de el web 2.0 van vindre una altra sèrie d'aplicacions que la van potenciar encara més, com els blogs o les xarxes socials. A poc a poc s'han anat afegint funcionalitats, fins al punt que fa pocs anys era impensable poder veure vídeos o pel·lícules en Internet, i hui és una cosa molt habitual.

Elements d'una aplicació web

En una aplicació web podem distingir en primer lloc dos grans costats: el **client**, on està l'usuari, que utilitza un navegador web (Google Chrome, Firefox, Internet Explorer, etc.) per a accedir a l'aplicació, i el **servidor**, on està situada l'aplicació (el fòrum, la xarxa social, el blog, el curs en línia, etc.), i que s'encarrega d'atendre les peticions dels clients i proporcionals la informació que sol·liciten.

Funcionament d'una aplicació web

Com hem comentat, les aplicacions web són un tipus d'aplicacions client-servidor. Aquest tipus d'arquitectures distribueixen les tasques entre els qui presten els recursos i serveis (els servidors) i els qui els sol·liciten (els clients).

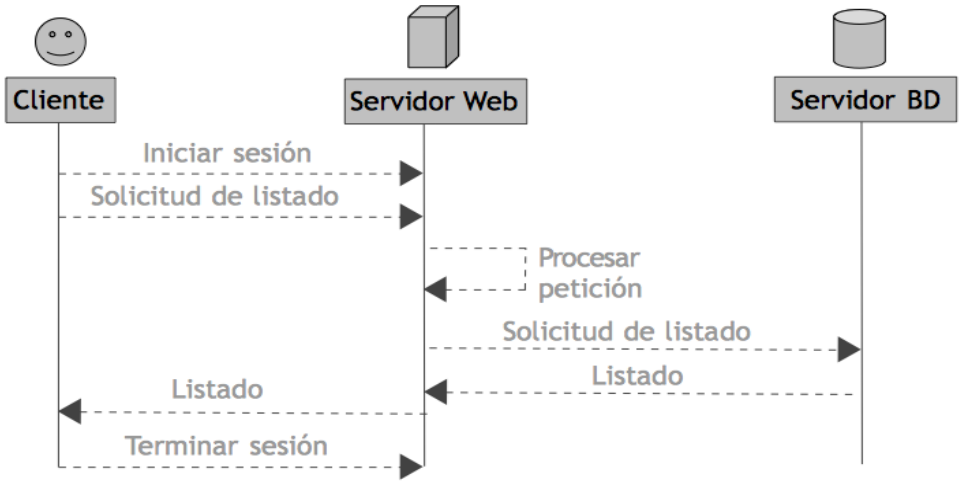
Els passos que se segueixen en la comunicació client-servidor, són, en general:

1. El client inicia sessió en el servidor
2. El client sol·licita al servidor el recurs o servei que vol utilitzar (una pàgina web, un document, pujar informació, etc.)
3. El servidor rep la resposta del client, la processa i decideix quin programa ha de donar-li servei, enviant la petició a aquest programa.
4. El programa responsable processa la petició, prepara la resposta i el lliurament al servidor.
5. El servidor envia la resposta al client
6. El client pot tornar al pas 2 i realitzar una nova petició, o bé
7. El client acaba la sessió en el servidor.

En general, el servidor no té per què executar-se només, sinó que podem tindre diferents aplicacions en diferents equips (o en el mateix), la qual cosa es coneix com a arquitectura **multicapa o multinivell**. Per exemple, un servidor de bases de dades en una màquina, un servidor web en una altra (o en la mateixa que el de bases de dades), un servidor de correu electrònic... i així distribuir els processos i el treball a realitzar, i fins i tot configurar opcions de seguretat i rendiment separades per a cada servidor.

Exemple:arquitectura de dos o tres nivells

Per exemple, si el client connectara amb el servidor per a demanar un llistat de notícies emmagatzemades en una base de dades, expressat com un diagrama de seqüència, el funcionament bàsic d'aquesta petició (i de l'arquitectura client-servidor en general) pot veure's com una cosa així:



En aquest exemple, el servidor web i el servidor de bases de dades podrien estar instal·lats en la mateixa màquina o en màquines separades, cadascuna amb el seu maquinari específic i control d'accés d'usuaris específic. En qualsevol cas, estem parlant d'una arquitectura de tres **nivells** (client, servidor web i servidor de base de dades), que és una cosa bastant habitual en les aplicacions web, perquè quasi totes compten amb una base de dades amb informació que consultar i modificar.

Sense el servidor de base de dades, estaríem davant una arquitectura de dos **nivells**, on el servidor és polivalent, i pot respondre directament a les peticions dels clients sense consultar amb altres servidors o aplicacions. Aquesta opció és menys flexible, menys segura i pot oferir pitjor rendiment en sistemes congestionats, al no poder dividir el treball entre diferents tipus de servidors.

URLs

Hem vist que, en l'esquema de funcionament d'una aplicació web, el client sol·licita recursos al servidor. La forma en què els sol·licita és mitjançant **URLs**. Una URL (*Uniform Resource Locator*) és una manera d'identificar i localitzar cada recurs d'una web. Per exemple, quan escrivim en un navegador una adreça com *http://www.miweb.com/paginas/pagina.html*, estem introduint una URL per a localitzar un recurs (en aquest cas, una pàgina HTML). Una URL es compon de:

- El **protocol**, que indica les regles que se seguiran per a comunicar-se client i servidor. Veurem més endavant alguns exemples de protocols, però per al que ens importa, en una URL el protocol va al principi, fins als dos punts i el delimitador //. En el nostre exemple, el protocol seria *http://*
- El **nom de domini**, que identifica al servidor i l'empresa/web a la qual connectarem. Va just darrere del protocol, fins a la següent barra. Normalment acaba en *.com, .és, .net*, etc. En el nostre exemple seria *www.miweb.com*

- La **ruta cap al recurs**, que comprén totes les carpetes i subcarpetes (si n'hi ha) i el nom d'arxiu que volem obtindre. En el nostre exemple, la ruta seria */pàgines/pàgina.html*

El navegador obté la URL que ha escrit l'usuari, transforma el nom de domini en una adreça IP (segons el seu servidor DNS assignat) i envia al servidor indicat la petició.

Llenguatges

En parlar d'aplicacions web en els següents temes, parlarem d'alguns conceptes que ens han de començar a sonar ja, com els ja comentats de client, o servidor. Un altre d'aquests conceptes és el de llenguatge . Per a les persones, un llenguatge és un conjunt de paraules i signes que ens permeten comunicar-nos. Per a un ordinador ve a ser una cosa semblant: un conjunt de paraules i signes que permeten que l'usuari li done una sèrie d'ordres, i que l'ordinador sàpia interpretar. En l'àmbit de les aplicacions web, distingim dos tipus de llenguatges:

- Llenguatges en l'entorn client o **llenguatges client**: són els que permeten que el client interactue amb l'aplicació web. Per a això, el client ha de poder veure l'aplicació web en el navegador, i interactuar amb ella (punxar en enllaços, emplenar formularis, etc.). En aquest costat, normalment es tenen dos llenguatges perquè tot això es puga dur a terme: el llenguatge HTML per a poder crear i visualitzar pàgines web, i el llenguatge Javascript per a poder facilitar la interacció entre l'usuari i el navegador. Els veurem en temes posteriors.
- Llenguatges en l'entorn servidor o **llenguatges servidor**: són els que permeten que el servidor faça unes certes tasques quan li arriben les peticions dels clients, com per exemple consultar una base de dades, guardar informació en un fitxer, o carregar una foto que l'usuari està pujant al servidor. En aquest altre costat, existeixen diverses famílies de llenguatges que podem triar, depenent del servidor web que vulguem utilitzar. Per exemple, podem utilitzar llenguatge ASP .NET (per a servidors de Microsoft i entorns Windows), o el llenguatge JSP (llenguatge Java per a aplicacions web), o el llenguatge PHP, entre altres.

Aplicacions web vs escriptori

Una vegada hem vist quina arquitectura tenen les aplicacions web, quins avantatges i inconvenients podem veure que tenen enfront de l'altre gran grup d'aplicacions, les d'escriptori?

Avantatges de les aplicacions web

- Una dels principals avantatges que tenen és que el client no ha d'instal·lar res en el seu equip, o quasi res. N'hi ha prou amb tindre instal·lat un navegador web, i com a molt en alguns casos, algun plugin especial (Flaix, Java, etc.). El servidor és el que necessita tindre instal·lat el programari per a fer funcionar l'aplicació (servidor web, base de dades, arxius necessaris, etc.)
- Un altre avantatge és que el client a penes té **càrrega de treball**. El servidor porta gran part del pes del processament (emmagatzemar dades, enviar arxius, etc.). Això a vegades pot suposar un desavantatge, si el servidor se satura. Veurem que hi ha mecanismes perquè el client faça part del treball i allibere així el servidor, amb llenguatges com Javascript, i tecnologies com AJAX.
- A més, per a utilitzar una mateixa aplicació diverses persones, no necessitem instal·lar-la en tots els equips on es vaja a utilitzar, ja que tots es connectaran amb el navegador al servidor web. D'altra banda, si fem un canvi en l'aplicació (per exemple, canviem els colors dels menús, o afegim alguna funcionalitat més), es farà en el servidor i automàticament el veuran actualitzat tots els usuaris. Això fa que les aplicacions web siguen més fàcils de mantindre **i actualitzar**
- La **compatibilitat** és un altre factor important. Moltes aplicacions d'escriptori només funcionen per a uns certs sistemes operatius, o funcionen diferent depenent del sistema. Una aplicació web, en general, funciona de la mateixa forma independentment de la plataforma. Això sí, per a dispositius mòbils solen tindre una versió adaptada, sense tant de text i amb funcionalitats més reduïdes o localitzades, per a poder-se manejar millor en una pantalla xicoteta.
- El **control d'usuaris i la seguretat** també estan centralitzats en el servidor, de manera que des d'ell podem veure i donar permisos a qui intenta accedir. En una aplicació d'escriptori és més difícil controlar que no s'entre de manera il·legal en algun dels equips on estiga instal·lada.
- La **mobilitat**, ja que en estar l'aplicació instal·lada en un servidor remot, si disposem de connexió a Internet podem utilitzar l'aplicació des de qualsevol lloc.
- L'escalabilitat del sistema, ja que la major part del treball recau en el servidor, es poden sempre millorar les seues prestacions i ampliar la capacitat de nombre de clients millorant el maquinari, afegint més servidors, etc.

Inconvenients de les aplicacions web

- Un dels inconvenients que podem trobar en una aplicació web és la seua **riquesa gràfica**, si la comparem amb aplicacions d'escriptori. Existeixen alguns tipus d'efectes (animacions, 3D, etc.) que encara són molt difícils o impossibles d'aconseguir en una aplicació web. A poc a poc han anat apareixent eines per a intentar pal·liar això, com a Flaix, llibreries Javascript, etc., però encara estan lluny de la potència i comoditat de les aplicacions d'escriptori.
- Un altre inconvenient evident és la **necessitat de connexió** a la xarxa per a poder utilitzar l'aplicació, encara que algunes aplicacions com Dropbox permeten treballar amb un document sense connexió i actualitzar els canvis en el servidor quan existisca connexió.
- El **trànsit generat** en la xarxa per a accedir al servidor també pot ser un factor important a considerar. Si hi ha diversos clients accedint a l'aplicació i sol·licitant dades, la quantitat d'informació que s'envia per Internet o que se sol·licita al servidor pot arribar a col·lapsar-lo, i que deixe de donar servei a tots els clients fins que es recupere.
    - Açò no és un problema en una altra mena d'arquitectures, com per exemple les xarxes P2P, on el nombre de clients millora el rendiment del sistema, a l'haver més llocs des d'on compartir i carregar/descarregar els recursos.
- Un altre inconvenient és el **temps de resposta**. En ser aplicacions client/servidor, pot passar un temps considerable des que, per exemple, enviem un formulari fins que el servidor ens avise que les dades s'han guardat correctament. Aquests temps són molt menors en aplicacions d'escriptori, al no necessitar comunicació entre equips.
- També podem citar entre els inconvenients l'aparença . A pesar que les aplicacions web són més compatibles, la seua aparença final depén del navegador que utilitze el client per a visualitzar-les. Així, la mateixa aplicació pot veure's de manera diferent segons si estem utilitzant Chrome, Internet Explorer, Firefox, etc.

Protocols més utilitzats

A l'hora de comunicar clients i servidors, és necessari establir un **protocol** de comunicació, és a dir, una sèrie de regles que indiquen quin tipus de missatges s'intercanviaran, en quina ordre i quin contingut tindrà cada tipus de missatge, de manera que els dos extrems de la comunicació (client i servidor) puguen entendre's.

En treballar amb aplicacions web, els protocols de comunicació més emprats són:

- **HTTP** (*HyperText Transfer Protocol*), un protocol existent des de 1990 i que permet la transferència d'arxius en general, encara que principalment d'arxius HTML (és a dir, documents web). Se segueix un esquema de peticions i respostes entre client i servidor com el vist anteriorment.
- **HTTPS**, versió segura del protocol anterior, on les dades de les peticions i les respostes s'envien encriptats, perquè ningú que intercepte la comunicació puga desxifrar el contingut d'aquesta. Aquest tipus de protocols se sol utilitzar en sistemes bancaris, plataformes de pagament (Paypal, per exemple), i altres aplicacions que manegen informació delicada (DNIs, nombres de targetes de crèdit, etc.).

  Normalment, els navegadors web canvien automàticament del protocol “normal” HTTP a HTTPS en connectar amb pàgines que necessiten ser més segures (login, dades de pagament, etc.). Es pot comprovar el canvi mirant la barra de direcció del navegador: en accedir al protocol segur es mostrarà el protocol *https* en la barra, o bé la icona d'un cadenat. No obstant això, sí que haurem de configurar el nostre servidor web per a acceptar comunicacions HTTPS, si fora el cas.
- Altres protocols són menys utilitzats a l'hora de treballar amb aplicacions web, però sí que s'utilitzen igualment en altres aplicacions que requerisquen d'Internet. Per exemple, per a l'enviament i recepció de correu electrònic s'empren els protocols **SMTP** o **POP3/IMAP**, respectivament. Per a enviar arxius a un servidor remot es pot emprar (a més del propi protocol HTTP) el protocol **FTP**.

Patrons de disseny programari

Un patró de disseny o arquitectura programari comprén un conjunt de pautes a seguir, elements a desenvolupar, jerarquies i ordre que doten a una aplicació d'una estructura preestablida, que la fa més propícia per a funcionar com deu. Els seus principals objectius són, d'una banda, estandarditzar la forma en què es desenvolupen les aplicacions, i per un altre, elaborar elements o components reutilitzables entre diverses aplicacions, en ajustar-se tots a un mateix patró.

En l'àmbit de les aplicacions web, existeixen patrons de disseny específics que ens guien a l'hora d'estructurar, dissenyar i programar aquestes aplicacions. Un dels més utilitzats (o potser el més utilitzat) és el patró MVC, que comentarem a continuació, però també han sorgit uns altres (molts a partir d'aquest), que han volgut fer un volt de rosca més, o adaptar-se a les necessitats d'aplicacions web més específiques o concretes. Veurem també alguns d'aquests patrons en aquest apartat.

El patró MVC

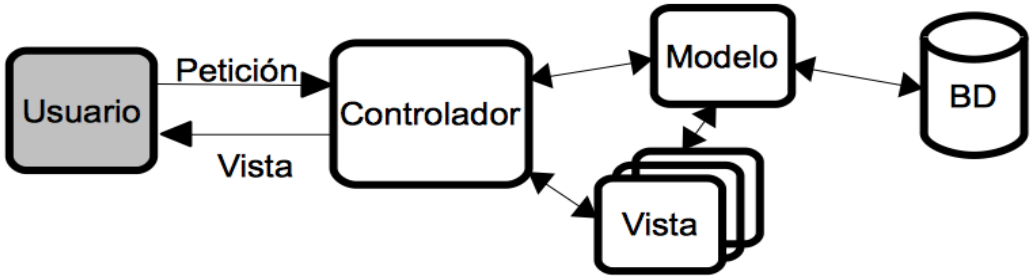
MVC són les sigles de Model-Vista-Controlador (o en anglés, *Model-View-Controller*), i és, com déiem abans, el patró per excel·lència ara mateix en el món de les aplicacions web, i fins i tot moltes aplicacions d'escriptori.

Com el seu nom indica, aquest patró es basa a dividir el disseny d'una aplicació web en tres components fonamentals:

- El **model**, que podríem resumir com el conjunt de totes les dades o informació que maneja l'aplicació. Típicament seran variables o objectes extrets d'una base de dades o qualsevol altre sistema d'emmagatzematge, per la qual cosa el codi del model normalment estarà format per instruccions per a connectar amb la base de dades, recuperar informació d'ella i emmagatzemar-la en algunes variables determinades. Per tant, no tindrà coneixement de la resta de components del sistema.
- La **vista**, que és l'intermediari entre l'aplicació i l'usuari, és a dir, la qual cosa l'usuari veu en pantalla de l'aplicació. Per tant, la vista la compondran les diferents pàgines, formularis, etc, que l'aplicació mostrarà a l'usuari per a interactuar amb ell.
- El **controlador** (o controladors), que són els fragments de codi encarregats de coordinar el funcionament general de l'aplicació. Davant peticions dels usuaris, les recullen, les identifiquen, i accedeixen al model per a actualitzar o recuperar dades, i al seu torn, decideixen quina vista mostrar-li a l'usuari a continuació de l'acció que acaba de realitzar.

És un patró de disseny molt concís i ben estructurat, la qual cosa li ha valgut la fama que té hui dia. Entre els seus molts avantatges, permet aïllar el codi dels tres elements involucrats (vista, model i controlador), de manera que el treball és molt més modular i divisible, podent encarregar-se de les vistes, per exemple, un dissenyador web que no tinga molta idea de programació en el servidor, i del controlador un programador PHP que no tinga moltes nocions d'HTML.

En forma d'esquema, podríem representar-ho així:



Les peticions de l'usuari arriben al controlador, que les identifica, i es comunica amb el model per a obtindre les dades necessàries, i amb les vistes per a decidir quina vista mostrar a continuació i omplir-la amb les dades del model, per a després servir-li-la a l'usuari com a resposta.

Altres alternatives al MVC

Com a alternatives al patró MVC, i arran d'aquest mateix patró, van sorgir uns altres una mica més específics. Quasi tots ells tenen com a base la part del *model* (és a dir, l'accés i gestió de les dades o informació de l'aplicació) i la part de la *vista* (és a dir, la presentació a l'usuari). Així, podríem dir que l'únic punt "discordant" seria el controlador, que en altres patrons s'ha substituït per altres elements. De fet, al conjunt de patrons que segueixen aquesta filosofia (és a dir, centrar-se en la vista i el model, i afegir alguna cosa més), se'ls sol cridar en general MVW (en anglés, *Movel-View-Whatever*, en espanyol, *Model-Vista-Qualsevol cosa*).

La finalitat d'això és, en alguns casos, descompondre el treball realitzat pels controladors en diversos submòduls, i en altres casos, prescindir directament del controlador. Vegem alguns dels patrons més populars.

El patró MVVM

El patró MVVM, com recullen les seues sigles, se centra exclusivament en els components del model i de la vista (*Model-Vista-Vista-Model*), i prescindeix del controlador. D'aquesta manera, l'usuari interactua directament amb la vista, i les accions o canvis que introduïsca en ella afecten directament el model, i viceversa (els canvis en el model es reflecteixen de manera automàtica en la vista).

Aquest patró està cobrant especial rellevància en les anomenades SPA (*Single Page Applications*), aplicacions web amb una sola pàgina que recarrega parcialment els seus continguts davant les accions de l'usuari. En aquests casos, no és necessari un controlador que diga quina vista carregar, perquè només hi ha una vista principal (que pot estar composta per subvistes), i si l'estructura és prou senzilla, vista i model poden estar intercomunicats sense intermediaris. Per a donar major suport encara a aquesta metodologia, han sorgit algunes llibreries de comunicació asíncrona amb el servidor (veurem aquest concepte en temes posteriors), com ara *Angular.js*.

El patró MOVE

El patró MOVE substituïx el controlador del patró MVC per dos elements. Un que denomina **operacions** (que seria l'O de les seues sigles), i que englobaria tot el conjunt d'accions que l'aplicació és capaç de realitzar, i un altre que serien els **esdeveniments**

(que seria l'E de les seues sigles), i que representarien tots aquells successos que desencadenen que s'execute una acció determinada. Així, per exemple, les accions dels usuaris són esdeveniments sobre l'aplicació que provoquen que s'executen determinades operacions. Aquestes operacions, al seu torn, poden accedir al model per a obtindre o actualitzar informació, i poden generar o cridar a una vista que mostrar a l'usuari com a resposta. Es divideix així la tasca dels controladors entre els esdeveniments i les operacions.

El patró MVP

El patró MVP substitueix el controlador (o controladors) del MVC pel que es denominen **presentadors**. Aquests presentadors són una espècie d'intermediaris entre el model i la vista, de manera que cada vista té el seu propi, i actua després de la vista per a comunicar-se amb el model, obtindre les dades, i carregar-los en ella per a mostrar-los a l'usuari. Es té així encapsulat amb cada vista el seu presentador, i l'aplicació pot considerar-se un conjunt de parells *vista-presentador,* que s'encarrega de comunicar-se amb el model, que queda per darrere.