DESENVOLUPAMENT D'APLICACIONS WEB

Carles Farré[1]
Professor agregat del Departament d'Enginyeria de Serveis i Sistemes d'Informació. Universitat Politècnica de Catalunya (Barcelona)

Resum: Aquest article pretén aportar una visió panoràmica del desenvolupament de les aplicacions web i, en particular, donar a conèixer algunes de les tècniques, eines i metodologies que usen els professionals que s'hi dediquen. Per aquesta raó caldrà analitzar amb una mica de detall l'origen, les categories, les característiques i l'arquitectura interna de les aplicacions web.

Paraules clau: aplicacions web, enginyeria web.

WEB APPLICATION DEVELOPMENT

Abstract: The goal of this paper is to provide an overview of the development of web applications and, in particular, to explain some of the techniques, tools, and methodologies used by the professionals working in this field. To achieve this goal it will be necessary to analyze in some detail the origin, categories, characteristics and internal architecture of web applications.

Keywords: web applications, web engineering.

1. Introducció

No descobrirem res de nou si diem que el web ha esdevingut una tecnologia omnipresent i indispensable per a la indústria, el comerç, la comunicació, l'educació, el lleure, etc. En les seves poques dècades de vida, ha canviat, en alguns casos substancialment, la manera com comprem productes i serveis, ens comuniquem i interactuem amb altres persones, aprenem...

En aquest article ens centrarem en el desenvolupament de les aplicacions web i, en particular, en el coneixement d'algunes de les tècniques, les eines i les metodologies que usen els professionals que s'hi dediquen.

Abans de tot, cal precisar què entenem per *aplicació web*. Recollim la definició que ens en dóna Kappel i altres (2009): «Un programa informàtic basat en tecnologies i estàndards del World Wide Web Consortium (W3C) que proporciona recursos específics web, com ara continguts o serveis, a través d'una interfície d'usuari: el navegador web».

Desglossem una mica la definició anterior. El W3C és una organització internacional presidida per l'inventor del web, Tim Berners-Lee, que desenvolupa i promou estàndards web. Exemples d'aquests estàndards serien el llenguatge HTML, el format d'identificació/localització de recursos URI o el protocol HTTP, que configuren la columna vertebral de la tecnologia web. Un altre punt important que es desprèn de la definició és que les aplicacions web estan concebudes essencialment per ser usades per humans mitjançant navegadors web. Això exclou, per tant, d'aquesta definició els serveis web, car els seus usuaris no són persones sinó altres programes informàtics.

Per entendre millor què són les aplicacions web, proposem la taxonomia que presentem a continuació. A més, per il·lustrar-ho millor, en la figura 1 situem les categories identificades segons el seu grau de complexitat i el moment aproximat en què han aparegut en aquests vint-i-cinc anys i escaig d'història del web. Tant per a la taxonomia i com per a la figura, aquí ens hem basat també en Kappel i altres (2009).

— *Aplicacions web estàtiques*. Són les primeres que van aparèixer en el mateix moment en què es va inventar el web, cap al 1990. Consistien en una sèrie de documents HTML[2] proporcionats per un servidor web determinat. L'usuari obtenia aquests documents a partir de les peticions que feia a través del seu navegador. El navegador es comunicava amb el servidor web mitjançant el protocol HTTP. Els documents s'identificaven per un únic URI, que contenia la informació per localitzar el servidor on

1. Correspondència: Carles Farré, *carles.farre@upc.edu*.

2. Més endavant expliquem el significat d'aquestes sigles.

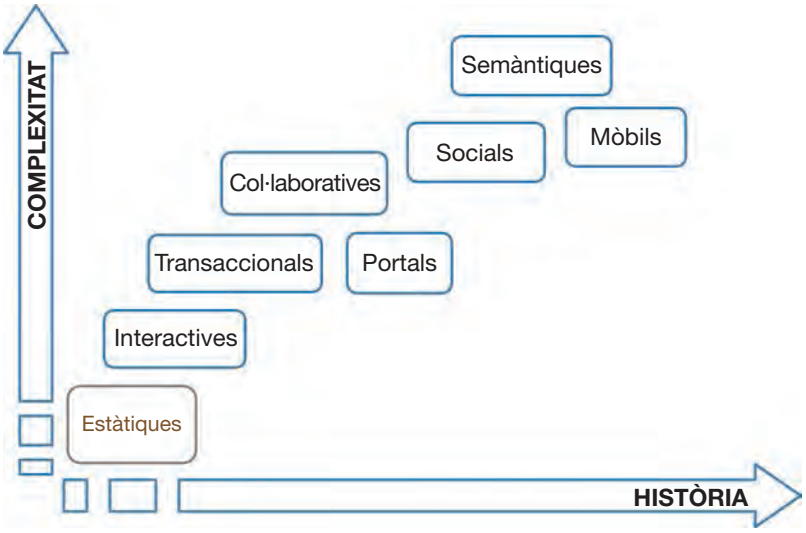

FIGURA 1. Categories d'aplicacions web.
FONT: Traducció de Gerti KAPPEL *et al.* (2009), *Web engineering*.

estaven hostatjats, així com la «ruta» per arribar a cada document, dins del seu servidor. Aquests documents HTML estaven físicament emmagatzemats com a fitxers dins del mateix servidor web. Per modificar-ne el contingut calia utilitzar editors de textos convencionals o específics per a pàgines HTML. Aquesta actualització del contingut quedava fora del que era la funcionalitat pròpia de l'aplicació web, i és per això que aquestes aplicacions reben el sobrenom de *estàtiques*. De fet, no s'acostumen a considerar aplicacions web com a tals i es prefereix anomenar-les *llocs web estàtics*.

— *Aplicacions web interactives*. Aquestes permetien parametritzar les peticions que feien els usuaris tot introduint filtres, preferències, etc., amb la qual cosa feien possible l'accés al contingut de manera més eficient. Els motors de cerca en serien un primer exemple. Tanmateix, el salt qualitatiu es va fer quan el contingut HTML retornat a l'usuari ja no fou el bolcat d'un fitxer físic, sinó el resultat de la consulta a una base de dades d'horaris de tren, per posar-ne un exemple. El contingut passa a ser, per tant, dinàmic. Per aconseguir-ho, va caldre ampliar les funcionalitats fins aleshores limitades dels servidors web per tal que fossin capaços d'executar programes informàtics específics encarregats d'accedir a la informació, emmagatzemada habitualment en una base de dades, per després generar-ne un document HTML degudament adaptat a la petició de l'usuari.

— *Aplicacions web transaccionals*. El canvi qualitatiu següent es va produir quan, a més de consultar els horaris dels trens, els usuaris van poder comprar el bitllet de tren a través de l'aplicació web. Per tant, les aplicacions web ja no es limitaven a consultar bases de dades, sinó que també podien fer altes, baixes i modificacions. L'era del comerç electrònic havia començat.

— *Portals*. Aquest tipus d'aplicacions web van proliferar al final de la darrera dècada del segle XX, la qual cosa va donar lloc a una bombolla financera, la bombolla de les empreses puntcom, que va esclatar l'any 2000. Tots aquests portals competien per esdevenir el punt d'accés únic a un seguit de fonts d'informació i serveis heterogenis, i intentaven fidelitzar els seus usuaris amb la creació de *comunitats* on aquests podien interrelacionar-se mitjançant xats, fòrums, etc.

— *Col·laboratives*. El fòrum d'usuaris va ser una de les primeres eines a través de les quals els usuaris van poder aportar el seu propi contingut a les aplicacions web. El pas següent fou convertir els usuaris en els principals proveïdors de contingut. La *Viquipèdia* n'és l'exemple paradigmàtic.

— *Socials*. Les aplicacions web amb l'etiqueta *social* tenen en comú que proporcionen una plataforma de comunicació i compartició de tota mena de continguts (texts, imatges, vídeos, etc.) entre els usuaris, amb la qual cosa permeten mantenir i àdhuc establir vincles personals d'amistat, de col·laboració, de fidelització a determinades marques, etc. En aquest cas, la col·laboració desinteressada dels usuaris no només aporta tot el contingut de l'aplicació web, sinó que permet que els seus propietaris es facin milionaris. En són exemples Facebook, Instagram, LinkedIn, Twitter, etc.

— *Semàntiques*. La idea que hi ha darrere les aplicacions web semàntiques és la d'enriquir-ne el contingut afegint-hi una sèrie de marques i etiquetes estàndard que permetin que aquest sigui «comprès» per algorismes informàtics, els quals llavors seran capaços de realitzar cerques «intel·ligents», recomanacions o altres tipus d'inferències. Es tracta d'una altra idea original de Tim Berners-Lee, però que aquest cop encara no s'ha acabat de materialitzar.

— *Mòbils*. L'aparició i proliferació de tota mena de dispositius mòbils que porten incorporats navegadors web ha provocat la necessitat d'adaptar la interfície de les aplicacions web a un ventall quasi infinit de possibles mides de pantalla i a la interacció tàctil. Les aplicacions web mòbils lliuren una guerra, que de moment van perdent, contra les aplicacions mòbils.

2. Característiques de les aplicacions web

Una altra manera de mirar d'entendre les aplicacions web i per què aquestes es diferencien d'altres tipus de progra-

mes informàtics és estudiant-ne les característiques més rellevants. Això no implica necessàriament que totes aquestes característiques siguin exclusives de les aplicacions web, que no ho són en la majoria dels casos, ni que totes les aplicacions web les hagin de tenir en el mateix grau. Per a aquest apartat ens hem basat principalment en Pressman i Lowe (2008).

— *Ús intensiu de la xarxa*. Les aplicacions web, com és obvi, requereixen tota una infraestructura de xarxa de comunicacions, més enllà del maquinari que puguin requerir tant els proveïdors d'aquestes aplicacions com els seus usuaris finals. Avui dia tendim a donar per descomptat aquest accés gairebé universal a Internet, tot i ser conscients que en altres indrets del nostre planeta això no és així. Però no cal anar gaire lluny tampoc. Quan viatgem a països del nostre entorn proper evitem sistemàticament pagar tarifes en itinerància abusives, per la qual cosa passem llargues estones sense connexió mentre som fora de l'hotel. Encara més sagnant és la situació de moltes empreses que estan ubicades en algun dels molts polígons industrials del nostre país als quals no arriba encara la fibra òptica.

— *Tecnologies específiques*. Pel simple fet de funcionar en el web, les aplicacions web es veuen obligades a utilitzar els protocols i estàndards propis d'aquest mitjà. El que podria ser una limitació tecnològica s'ha convertit en un gran avantatge en tractar-se d'una tecnologia oberta i escalable que s'ha pogut estendre arreu en pocs anys.

— *Concurrència i temps de resposta*. El fet de tenir una aplicació al web implica que qualsevol persona del món amb un navegador web i una connexió a Internet n'és un usuari potencial. Per tant, un dels reptes de les aplicacions web és dimensionar adequadament els seus recursos (servidors, bases de dades, etc.) per tal d'oferir un temps de resposta acceptable a tots els usuaris que es prevegi que es puguin arribar a connectar simultàniament.

— *Hipermèdia*. L'ús d'HTML permet a les aplicacions web mostrar text, però també tota mena de contingut com ara imatges, àudios, vídeos, etc. Quan aquest contingut multimèdia és propi, el seu emmagatzemament i la gestió suposen uns reptes addicionals. Que una aplicació web consisteixi en un conjunt entrellaçat de pàgines web i que, a més, inclogui enllaços a pàgines d'altres aplicacions permet a l'usuari gaudir d'una llibertat en la navegació per l'aplicació web que no troba en altres tipus d'aplicacions, on la transició entre pantalles està molt més pautada. Però aquí es corre el risc que l'usuari es desorienti, no acabi tasques més complexes, com ara omplir dades en una sèrie de formularis que exigeixen fer-ho de manera seqüencial, o totes dues coses. Tot plegat fa que la *usabilitat*, és a dir, la facilitat en l'aprenentatge i ús d'una interfície, sigui un requisit de qualitat ineludible.

— *Estètica*. En moltes aplicacions web, les aparences importen. I si es tracta de vendre productes o serveis, encara més. En aquest darrer cas, s'ha de transmetre una imatge de professionalitat, seriositat i confiança. En altres tipus d'aplicacions, per contra, potser caldrà utilitzar una estètica innovadora per sorprendre i atreure nous usuaris. En qualsevol dels casos, el desenvolupament de l'aplicació web requerirà professionals experts en la definició i creació d'elements estètics.

— *Evolució contínua*. Els usuaris de les aplicacions web no s'han de descarregar i instal·lar les actualitzacions, ja que quan s'hi connecten sempre ho fan amb la darrera versió actualitzada. Això dóna molta més llibertat i flexibilitat als desenvolupadors a l'hora de planificar i implementar canvis, millores i correccions d'errors.

— *Seguretat*. Aquest és un tema recurrent i força important. És segur comprar a X? Estaran segures les meves dades a Y? Com puc assegurar-me que ningú no intercepti les dades que envio a W? És fàcil que algú suplanti la meva identitat a Z? Totes aquestes preguntes i moltes altres relacionades amb la seguretat s'han de poder respondre de manera satisfactòria i, per tant, cal que els desenvolupadors d'aplicacions coneguin les principals amenaces i contramesures i que hi hagi experts que auditin de manera periòdica la seguretat de l'aplicació web.

3. Enginyeria web

Algunes de les característiques més «externes» de les aplicacions web, com ara la importància del contingut i de la seva presentació, la navegabilitat, la usabilitat, etc., van comportar l'emergència de nous perfils professionals: dissenyadors gràfics per al web, redactors de continguts, experts en experiència d'usuari, etc.

Tal com han explicat Pressman i Lowe (2008), això va donar lloc a un intens debat als primers anys d'aquest segle per decidir si els mètodes i les tècniques que s'havien aplicat fins aleshores en el desenvolupament de programes informàtics eren també vàlids per a les aplicacions web.

Arran d'aquest debat va aparèixer el concepte *enginyeria web*, que segons Kappel i altres (2009) es pot definir com «l'aplicació d'enfocaments sistemàtics i quantificables (conceptes, mètodes, tècniques, eines) a l'anàlisi de requisits, disseny, implementació, prova, operació i manteniment efectius d'aplicacions web d'alta qualitat». De fet, si a aquesta definició substituïm *aplicacions web* per *programes informàtics*, ens trobarem amb la definició gairebé oficial d'*enginyeria del programari*.

Vista ja amb certa perspectiva, aquesta aparent confrontació «enginyeria del programari *versus* enginyeria web» ha quedat clarament superada. Només cal veure l'auge que estem vivint aquests darrers anys en el camp del desenvolupament d'aplicacions per a mòbils, que comparteixen característiques comunes amb les aplicacions web, i a ningú no se li ha acudit parlar d'«enginyeria mòbil». De totes maneres, el concepte *enginyeria web* s'ha mantingut, potser com a testimoni d'aquells «anys bojos» en què aplicacions web com ara Amazon o Facebook es convertiren en «gegants d'Internet».

3.1. *Metodologies de desenvolupament per a aplicacions web*

Un dels aspectes clau tant en l'enginyeria web com en l'enginyeria del programari és decidir quina metodologia de desenvolupament cal seguir. Les metodologies de desenvolupament prescriuen quines fases i activitats s'han de realitzar durant el procés de desenvolupament d'una aplicació, des de l'inici, en què es formulen les necessitats i els objectius inicials de les parts interessades (*stakeholders*, en anglès), fins al final, en què l'aplicació s'instal·la i es lliura a aquestes parts.

S'han proposat moltes metodologies al llarg de la història de l'enginyeria del programari, encara que fins ara cap no ha assolit un grau d'acceptació i ús prou generalitzats per ser considerada la metodologia de referència per a tota la professió d'enginyers del programari. Pel que fa a l'enginyeria web, els desenvolupadors s'han limitat a adaptar les metodologies de desenvolupament del programari existents.

Pressman i Lowe (2008) no proposen la utilització d'una metodologia concreta, però sí un marc de referència al qual la metodologia escollida s'hauria d'acomodar, tenint en compte que en la majoria dels casos les aplicacions web es construeixen de manera incremental i s'han d'actualitzar sovint i en poc temps per adaptar-se a nous requisits o preferències. La figura 2 descriu a grans trets aquest marc de referència.

Aquest marc de referència proposa, per tant, seguir processos de desenvolupament iteratius, en què en la finalització de cada iteració s'obtingui una versió completament funcional de l'aplicació web, encara que no completa. Dins de cada iteració s'han de realitzar una sèrie d'activitats seqüencials:

— *Comunicació*. Implicant clients, usuaris i altres parts interessades, s'han de dur a terme les accions següents:

a) *Formulació*, en què s'analitza el context i les necessitats de negoci, especificant clarament totes les parts interessades.

b) *Determinació dels requisits* que s'hauran de satisfer, tant pel que fa a les funcionalitats com a la qualitat d'execució.

c) *Negociació*, per conciliar els interessos i les prioritats de les diverses parts interessades.

— *Planificació*. Es defineixen totes les tasques a realitzar durant la iteració, amb la programació temporal i l'assignació de recursos (programadors, equipaments...) necessaris.

— *Modelització*. S'adapten les eines i les tècniques convencionals de l'enginyeria del programari per determinar i documentar, abans de programar el codi, què cal fer (anàlisi) i com s'ha de fer (disseny).

— *Construcció*. La programació pròpiament dita dels components de la nova versió de l'aplicació, que es van provant a mesura que s'implementen o modifiquen (proves unitàries) i es van acoblant amb altres components (proves d'integració).

— *Lliurament i/o instal·lació* de la nova versió de l'aplicació web construïda, que s'avalua amb les parts interessades (proves d'acceptació) perquè manifestin la seva opinió i grau de satisfacció (retroacció).

De les metodologies de desenvolupament existents, les anomenades *àgils* (*Scrum*, *eXtreme Programming*, etc.) són les que s'adapten millor a aquest marc proposat i, de fet, són les que més s'estan utilitzant actualment. En aquest tipus de metodologies, les iteracions són relativament curtes (entre dues i vuit setmanes) i la modelització és poca, és realitzada informalment o totes dues coses, ja que hi ha una certa «aversió» a dedicar massa temps a qualsevol activitat que no es consideri estrictament necessària per a la implementació del codi.

4. Arquitectura de les aplicacions web

En informàtica sovint utilitzem el concepte *arquitectura* quan volem descriure un sistema mostrant-ne els components i la manera en què s'interrelacionen. Descriure completament i amb precisió una arquitectura és sempre una tasca d'una gran dificultat i complexitat, pel fet que sempre hi ha nombrosos nivells de detall i diferents punts de vista.

Aquí farem nostre el recurs proposat per Fox i Patterson (2013) d'utilitzar l'altitud com a metàfora dels diferents

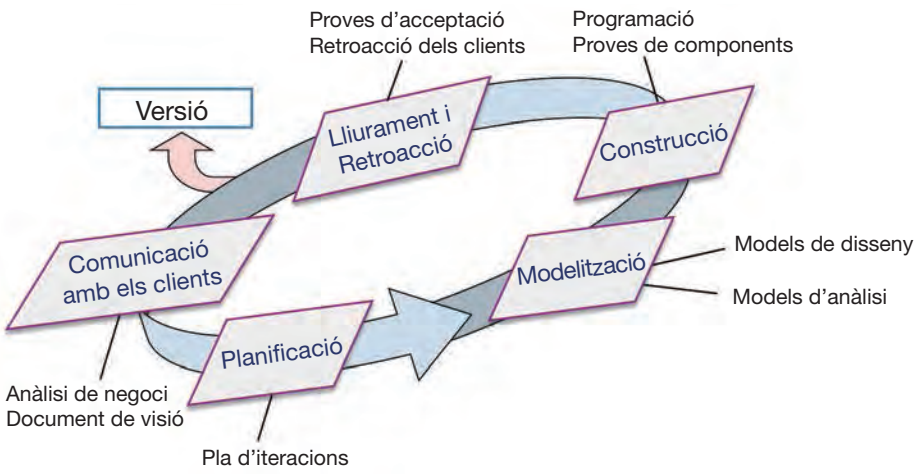

FIGURA 2. Marc de referència per a metodologies de desenvolupament d'aplicacions web.
FONT: Traducció de Roger PRESSMAN i David LOWE (2008), *Web engineering: A practitioner's approach*.

nivells de detall amb què descrivim l'arquitectura d'una aplicació web, tal com es pot veure en la figura 3. El resultat serà una descripció informal i incompleta, però creiem que serà prou didàctica i representativa. Partirem de dalt de tot, a 50.000 metres, per anar descobrint els diferents elements a mesura que perdem altura.

4.1. *Altitud 50.000 metres: arquitectura client-servidor*

Suspesos per un moment a aquesta altura, observem les aplicacions web des del punt de vista d'un usuari qualsevol. Independentment del dispositiu (ordinador, telèfon, tauleta...) que vulguem utilitzar per accedir al web i a les seves aplicacions, necessitem un programa informàtic específic per poder-ho fer: un navegador web. Per accedir als continguts i les aplicacions web, el navegador es connecta amb els servidors que proporcionen aquests recursos. El navegador web és un client universal, en el sentit que ens permet accedir a qualsevol aplicació web i interactuar-hi. Per la seva banda, els servidors d'aplicacions i continguts web estan preparats per interactuar amb molts clients alhora. En l'arquitectura client-servidor, el client és sempre qui inicia la connexió, enviant la petició corresponent al servidor, que li retornarà una resposta tan bon punt la tingui.

4.2. *Altitud 25.000 metres: HTTP i URI*

Evidentment, donem per fet que tant els navegadors com els servidors web estan connectats i es comuniquen via Internet. Ara no entrarem a explicar amb detall com funciona Internet. N'hi ha prou de dir que és una immensa infraestructura descentralitzada, al capdamunt de la qual es troben els protocols de comunicació TCP/IP. Aquests protocols defineixen l'anomenat *nivell de transport*, que proporciona una connectivitat punt a punt entre dos elements qualssevol. Per sobre d'aquest nivell de transport tenim el nivell d'aplicació. Exemples d'aplicacions d'Internet són, entre altres, el correu electrònic, la telefonia per IP, la compartició d'arxius, la reproducció de vídeo en *streaming* i, per descomptat, el web. El protocol que utilitzen els navegadors web per comunicar-se amb els servidors web és el protocol de transferència d'hipertext o HTTP, per la sigla en anglès. L'altre element clau són els identificadors uniformes de recursos o URI, per la sigla en anglès. Tot allò a què es pot accedir a través d'un web, ja sigui un fitxer HTML, una imatge, una aplicació, etc., té el seu URI.

Vegem ara el cas més senzill de funcionament del web, que hem il·lustrat mitjançant la figura 4.

1. L'usuari escriu a la barra del navegador l'URI de la pàgina web a què vol accedir, en aquest cas, *http://www.essi.upc.edu/index.html*. El més habitual, però, és que l'usuari no

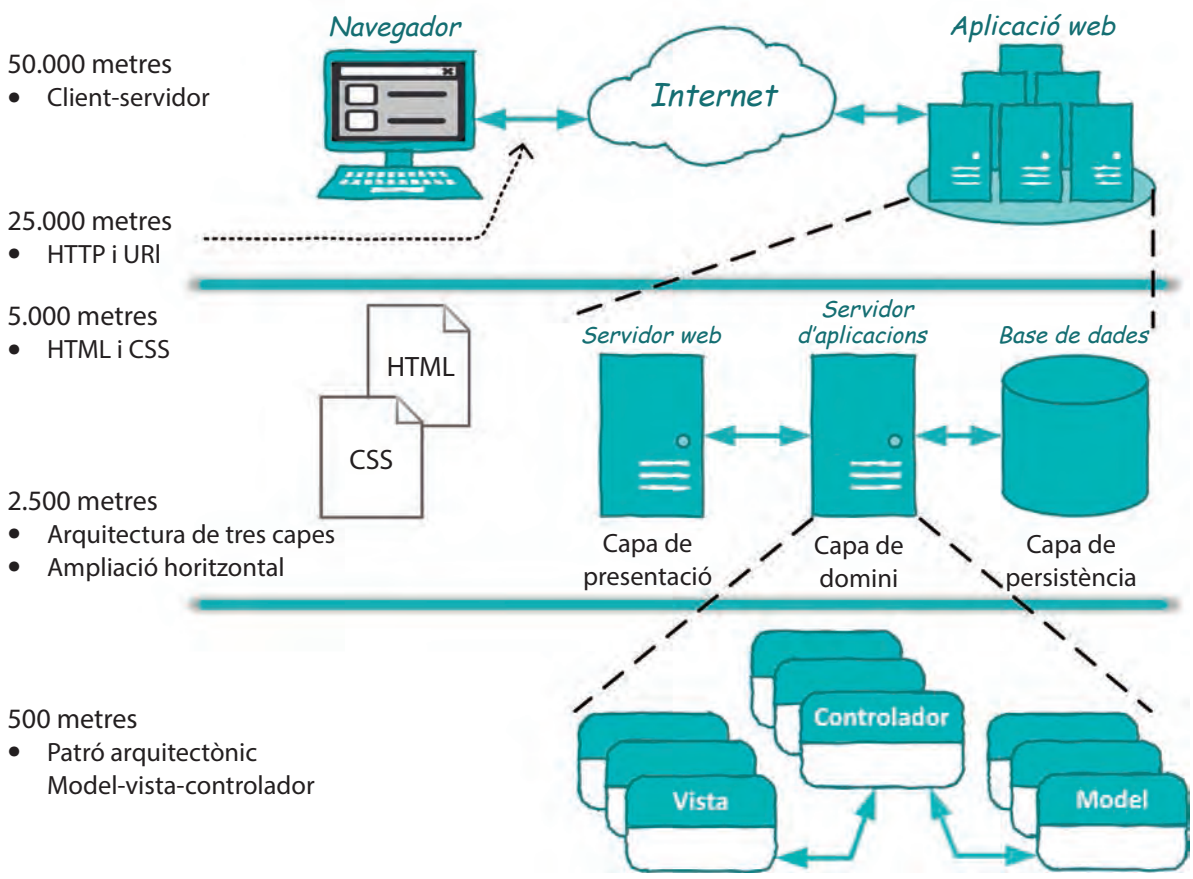

FIGURA 3. Arquitectura d'una aplicació, usant l'altitud com a metàfora dels diferents nivells de detall.
FONT: Adaptació d'Armando Fox i David PATTERSON (2013), *Engineering software as a service*: *An agile approach using cloud computing*.

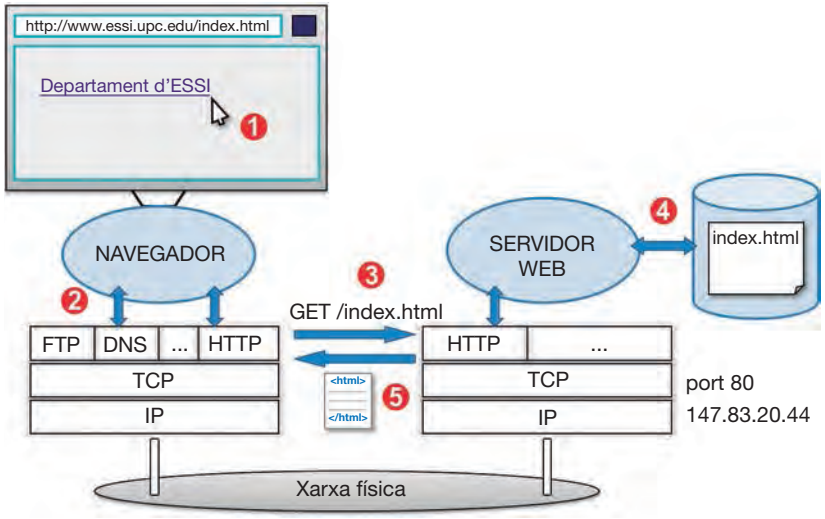

FIGURA 4. Esquema d'una interacció HTTP típica entre un navegador i un servidor web.
FONT: Elaboració pròpia.

escrigui directament aquest URI, sinó que l'activi en clicar l'enllaç corresponent dins de la pàgina web que està visualitzant en aquell moment.

2. El navegador analitza l'URI introduït o clicat i interpreta que ha d'usar el protocol HTTP, perquè això ho indica l'URI (*http://*). Per tant, necessita establir primer una connexió punt a punt TCP/IP amb el servidor indicat (*www.essi.upc.edu*). Abans, però, cal utilitzar un altre servei d'Internet, el DNS (*sistema de noms de domini*, en català), per tal d'obtenir l'adreça IP corresponent (147.83.20.44). El navegador es connectarà amb el port 80 del servidor, que és el que usen per defecte els servidors web. Si aquest port fos un altre, per exemple, el 8080, caldria incloure'l explícitament a l'URI: *http://www.essi.upc.edu*:8080/*index.html*.

3. Establerta la connexió TCP/IP amb el port 80 de l'adreça 147.83.20.44, el navegador envia un missatge de text amb un format determinat: una *petició HTTP*. En aquest cas, la petició indica quina acció es vol fer (GET, que vol dir 'obtenir') i sobre quin recurs (*/index.html*).

4. El servidor web rep la petició HTTP. Per simplificar, suposem que el recurs demanat correspon a un fitxer que té emmagatzemat al seu disc dur.

5. El servidor retorna al navegador una resposta HTTP, que inclou el contingut del fitxer esmentat, i tanca la connexió TCP/IP amb el client.

6. Com a pas final, no mostrat a la figura, el navegador mostrarà per pantalla el recurs demanat. En tractar-se d'una pàgina web, és molt probable que aquesta tingui associada altres recursos: imatges, fulls d'estil, etc. El contingut HTML que ha obtingut com a resposta del servidor només conté les referències (URI) a aquests recursos. Per tant, per visualitzar degudament la pàgina web, el navegador haurà d'obtenir també aquests recursos i, en conseqüència, haurà de repetir els passos 1-5 anteriors per a cadascun d'ells.

A banda de GET, hi ha altres accions que es poden usar en les peticions HTTP. L'acció POST (enviar) és una altra de les que s'utilitzen sovint en les aplicacions web. Habitualment, el navegador l'utilitza quan ha d'enviar al servidor les dades que l'usuari ha introduït omplint un formulari.

4.3. *Altitud 5.000 metres: HTML i CSS*

Les respostes HTTP són «agnòstiques» respecte al tipus (i el format) del contingut que el servidor retorna al navegador: pot tractar-se d'una imatge (PNG, JPG, GIF...), d'un àudio (MPEG, WAV...), d'un vídeo, PDF, etc. Tanmateix, el contingut HTML és, amb molta diferència, el més important, fins al punt de poder-lo considerar el tercer pilar del web, al costat d'HTTP i els URI, inclosos, tots tres, en la proposta original de Tim Berners-Lee (Berners-Lee, 2000).

L'HTML (llenguatge d'etiquetatge d'hipertext) és, per tant, el llenguatge principal de les pàgines web i, per extensió, de les aplicacions web. Una de les seves característiques ja l'hem esmentat abans: permetre referenciar, mitjançant els URI corresponents, tots els recursos (imatges, fulls d'estil, etc.) que el navegador haurà de tenir en compte per mostrar el contingut HTML a l'usuari, més enllà del text que ja hi està inclòs. Ara bé, sens dubte, la característica més important de totes és la possibilitat de definir i mostrar (hiper)enllaços a altres pàgines web, a través dels URI corresponents. Ho hem vist també en l'exemple de la figura 4: el «Departament d'ESSI» que apareix a la pantalla del navegador és un enllaç, que en HTML s'especificaria de la manera següent: Departament d'ESSI.

Evidentment, es pot enllaçar qualsevol cosa que tingui un URI, independentment de si pertany o no al mateix lloc o aplicació web. D'aquesta manera es teixeix tota una teranyina (*web*, en anglès) de referències creuades que permet als usuaris navegar d'un lloc o una aplicació a l'altre sense solució de continuïtat.

Els fulls d'estil en cascada (*cascading style sheets*, CSS) foren introduïts amb posterioritat amb la finalitat d'«externalitzar» de l'HTML la definició dels aspectes més visuals de la seva representació: els colors, les mides de lletra, la

posició dels diferents elements, etc. Això és força important perquè facilita, alhora, la divisió del treball i la col·laboració entre els qui s'han d'encarregar de proporcionar el contingut, per una banda, i els qui dissenyen com aquest ha d'aparèixer per pantalla, per l'altra.

4.4. *Altitud 2.500 metres: arquitectura en capes*

En aquest descens que fem des de les altures, arribem ara al punt on ja es fan paleses les diferències entre un lloc web «clàssic», que serveix contingut HTML estàtic, i una aplicació web pròpiament dita. L'exemple de la figura 4 ens n'il·lustra el primer tipus. Les aplicacions web es caracteritzen, doncs, per generar contingut HTML dinàmic sota demanda. Per fer-ho, cal que executin programes informàtics específics capaços d'interpretar les peticions HTTP dels navegadors, extreure'n els paràmetres necessaris, realitzar les operacions necessàries a la base de dades i retornar la resposta adequada.

Tornant a la figura 3, podem veure que les aplicacions web habitualment s'estructuren en tres capes lògiques. A la capa de presentació tenim el servidor web pròpiament dit, que rep les peticions dels navegadors i serveix el contingut estàtic (imatges, fulls d'estils, etc.). Si la petició requereix la generació de contingut dinàmic, aquesta és transferida a la capa de domini, on tenim el servidor d'aplicacions que executa els programes específics que esmentàvem abans i que, òbviament, són diferents en funció de les funcionalitats que admet cada aplicació web. Finalment, tenim la capa de persistència, on tindrem un servidor de dades en què s'emmagatzemarà tota la informació necessària per al correcte funcionament de l'aplicació.

En els casos en què tant la complexitat com la càrrega (nombre de peticions per unitat de temps) es preveuen petites, les tres capes lògiques poden coexistir en una sola màquina «física». Així, per exemple, tenim el cas d'Apache, el programari amb funcionalitats de servidor web més utilitzat arreu, que té una arquitectura extensible que permet afegir-hi els mòduls necessaris per executar programes informàtics escrits en uns llenguatges determinats, i també fa, per tant, la funció de servidor d'aplicacions. Si a la màquina on tenim aquest servidor Apache afegim un servidor de dades lleuger com ara MySQL, llavors ja tenim tot el quadre complet.

Quan els requisits de càrrega de l'aplicació són més exigents, és més convenient tenir les capes lògiques també separades físicament. D'aquesta manera, a més, ens podem anar adaptant als possibles increments de demanda de càrrega futurs a còpia d'escalar horitzontalment la capa que més ho necessiti en cada moment, és a dir, afegir servidors replicats a la capa en qüestió. En la figura 5, per exemple, s'observa com s'han escalat horitzontalment les capes de presentació i de domini afegint-hi un i dos servidors, respectivament. Això ha requerit afegir dos nous tipus de components que apareixen també a la figura. Els equilibradors de càrrega distribueixen les peticions entrants entre els diferents servidors disponibles segons el grau d'ocupació que té cadascun d'ells en cada moment. El tallafocs és un mecanisme de seguretat que filtra les peticions entrants a partir d'unes regles determinades.

4.5. *Altitud 500 metres: model-vista-controlador*

El codi de la capa de domini pot resultar força complex. De nou, la solució habitualment adoptada en informàtica és la

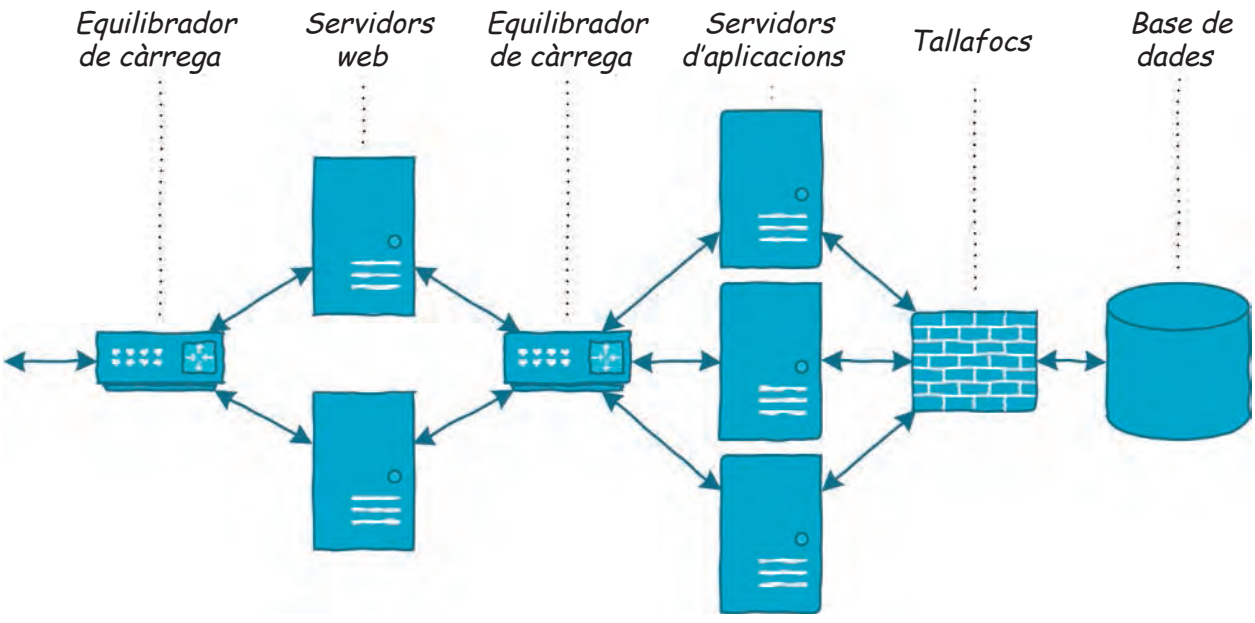

FIGURA 5. Arquitectura en capes i escalabilitat horitzontal.
FONT: Traducció i adaptació d'Armando Fox i David PATTERSON (2013), *Engineering software as a service: An agile approach using cloud computing.*

modularitat: descompondre un sistema gran en diversos components interrelacionats amb unes responsabilitats repartides i unes regles del joc ben clares. En aplicacions que requereixen un gran nivell d'interactivitat amb usuaris finals a través d'interfícies gràfiques, com és el cas que ens ocupa, l'aplicació del patró model-vista-controlador (MVC), amb variants diferents, s'ha acabat generalitzant. Segons aquest patró, els principals components d'una aplicació pertanyen a un d'aquests tipus: models, vistes o controladors.

Els models són els components que s'encarreguen de gestionar les dades de l'aplicació. Estan, per tant, connectats directament a la capa de persistència per fer les consultes, les actualitzacions, les insercions i les eliminacions necessàries. Normalment, hi ha tants models com entitats, o taules, hi ha a la base de dades.

Les vistes presenten la informació dels models als usuaris. En molts casos, això implica que són els components encarregats de generar directament el contingut HTML dinàmic requerit.

Els controladors són els components encarregats de processar en primera instància totes les peticions que es fan a l'aplicació, cridant les operacions dels models que siguin necessàries, amb els paràmetres que hagin vingut amb la petició, i, en funció del resultat d'aquestes operacions, determinant quines vistes s'han d'executar a continuació, es passen a les vistes les dades dels models que aquelles necessiten mostrar a la resposta.

Atenent la interpretació «clàssica» de l'arquitectura lògica en capes que hem esmentat en l'apartat anterior, les vistes i els controladors formarien part de la capa de presentació, en el sentit que els usuaris interactuen directament amb aquests components per accedir a les funcionalitats del domini. I, de fet, és així en la majoria de les aplicacions d'escriptori que apliquen el patró MVC. Ara bé, quan traslladem al web aquests dos patrons arquitectònics, arquitectura en capes i MVC, la cosa ja no és tan clara. Aquí els usuaris interactuen directament amb el navegador, que, al seu torn, es comunica amb un servidor web que llavors necessita un servidor d'aplicacions. Es fa difícil, per tant, establir la frontera entre capa de presentació i capa de domini. Podem suposar que està entre el navegador i el servidor web, per exemple, o, per contra, dins del mateix servidor d'aplicacions, separant aquells components que processen més directament les peticions i les respostes HTTP (controladors i vistes, respectivament) d'aquells altres que encapsulen la lògica del domini (models). En la figura 3 s'ha optat —de manera salomònica, podríem dir— per situar la frontera entre la capa de presentació i la capa de domini en la separació que hi ha entre el servidor web i el servidor d'aplicacions.

4.6. *Altitud 100 metres:* web application frameworks

El nivell 0, el sòl, d'aquest descens que hem anat fent és el codi dels components de l'aplicació. Aquest codi estarà escrit principalment en un llenguatge de programació determinat. Hi ha molts llenguatges de programació per triar: Java, JavaScript, Ruby, Python, PHP, etc. Tots tenen avantatges i inconvenients, partidaris incondicionals i detractors acèrrims.

Ara bé, avui dia és pràcticament inconcebible plantejar-se la programació d'una aplicació web sense tenir en consideració la utilització d'un *web application framework* (WAF), que es podria traduir per 'entorn de treball per a aplicacions web'. Els WAF faciliten la implementació de les aplicacions web en un llenguatge de programació determinat prescrivint una estructura interna consistent per als components de l'aplicació, així com la manera i l'ordre en què aquests components s'invoquen en el temps d'execució, a banda que donen suport a les tasques més habituals (autenticació d'usuaris, gestió de sessions, accés a bases de dades...), o bé les tenen parcialment automatitzades.

Podem trobar WAF de propòsit general, que serveixen per implementar qualsevol tipus d'aplicació. La majoria segueixen l'arquitectura en capes i el patró MVC ja esmentats. Alguns exemples, segons el llenguatge de programació, són:

- Java: Spring, Grails, Struts 2
- JavaScript: AngularJS, Backbone.js, Ember.js
- Ruby: Ruby on Rails
- PHP: Laravel, Silex, Symfony
- Python: Django

Llavors tenim els WAF especialitzats, que s'adrecen a un tipus específic d'aplicacions web amb unes funcionalitats habituals. Un exemple paradigmàtic són els sistemes de gestió de continguts (SGC, o per la sigla en anglès, CMS). Les funcions bàsiques dels SGC són la publicació, l'edició, l'organització i el manteniment de diferents tipus de contingut, alhora que donen suport a la participació dels usuaris a través de diversos mitjans (comunitats, votacions, comentaris, fòrums, etc.). Exemples coneguts són Drupal, Joomla i WordPress, tots tres basats en el llenguatge PHP, o Liferay, basat en Java. Un altre segment especialitzat són els WAF per a comerç electrònic, que proporcionen les funcionalitats típiques de les botigues en línia. Exemples d'aquest segon tipus són Magento, PrestaShop o osCommerce, escrits en PHP. A diferència dels WAF de propòsit general, en els WAF específics no s'espera que els desenvolupadors que els utilitzin hagin d'escriure el seu propi codi si no és en casos molt excepcionals. Per contra, la seva feina se centrarà a configurar i adaptar el WAF a les necessitats concretes de la seva aplicació.

5. Modelització d'aplicacions web

En moltes disciplines de l'enginyeria, la construcció de models, ja siguin plànols, diagrames, maquetes, prototipus, etc., és un element clau, imprescindible i sovint obligatori per a la realització de qualsevol projecte. El fet és que, en les enginyeries del programari i web, la modelització no és una activitat gaire estesa dins de la pràctica professional, fins al punt que algunes de les metodologies de desenvolupament més utilitzades actualment, les anomenades *àgils*, podem dir que «presumeixen» de mantenir la

modelització en la seva mínima expressió per no interferir en les tasques que es consideren principals: escriure codi i provar-lo.

La utilització de models en les enginyeries del programari i web pot respondre a objectius diferents en funció del grau d'importància que se'ls vulgui donar:

— Els models com a esbossos. Aquí només es pretén millorar la comunicació i la discussió d'alternatives entre els desenvolupadors, i entre aquests i les parts interessades.

— Els models com a guia i prescripció detallada d'allò que s'implementarà, i on es consignen totes les decisions de disseny. A part del temps que això pot consumir, el repte principal aquí és com aquests models evolucionen de manera sincronitzada amb el programari, que necessita adaptar-se contínuament tan bon punt la primera versió s'ha construït d'acord amb aquests models.

— Els models com a codi. Les aplicacions es generen de manera semiautomàtica a partir dels models. Les ampliacions i els canvis futurs no s'apliquen mai sobre el codi generat sinó sobre els models de partida.

Si bé, com hem dit abans, les metodologies de desenvolupament àgils tendeixen a usar els models, a tot estirar, com a esbossos, trobaríem una excepció en aquells casos en què s'aplica la tècnica anomenada *desenvolupament guiat pel comportament* (*behaviour driven development*, BDD). Segons aquesta tècnica, el funcionament de l'aplicació s'especifica prèviament mitjançant uns models anomenats *històries d'usuari*, que descriuen una funcionalitat concreta i ben acotada des del punt de vista de l'usuari final, que interactua amb la interfície de l'aplicació; per exemple, afegir un producte a la cistella de la compra. Aquestes històries d'usuari s'escriuen en llenguatge natural, però de manera força pautada. Continuant amb el mateix exemple:

Funció: Cistella de la compra
Escenari 1:
Atès que jo sóc un usuari autoritzat
Quan jo vaig a la pàgina d'un producte
I jo clico a "Afegeix producte a la cistella"
Llavors la quantitat de productes de la meva cistella s'hauria d'incrementar
I el preu de la meva cistella hauria d'augmentar.

Aquestes històries no s'utilitzaran per generar el codi de l'aplicació, però sí per comprovar que la implementació que se'n faci compleixi fil per randa tot el que s'hi descriu. Per a aquest fi, s'utilitzen eines de prova com ara Cucumber o JBehave, que són capaces de simular la interacció descrita en una història d'usuari sobre l'aplicació que es vol provar.

6. Implementació d'aplicacions web: eines i tècniques

A l'hora d'escriure el codi de l'aplicació, potser una de les decisions més importants és triar el WAF a partir del qual es desenvoluparà l'aplicació. Es poden tenir en compte diferents factors per prendre aquesta decisió, com ara la familiaritat amb el llenguatge de programació admès, l'experiència prèvia en utilització d'entorns similars, la maduresa del WAF (existència d'un historial de versions estables), la disponibilitat d'una documentació de suport actualitzada (llibres, tutorials, casos pràctics...), l'acceptació i el suport que pugui tenir el WAF per part d'una comunitat nombrosa i activa de desenvolupadors, etc.

Un altre factor que es pot tenir en compte a l'hora de triar el WAF és el suport que pugui tenir en entorns de desplegament d'aplicacions en núvol, un tipus de servei d'informàtica en núvol anomenat en anglès *plataform as a service* (PaaS). Amb els serveis d'informàtica en núvol (*cloud computing*), els desenvolupadors i emprenedors no han d'adquirir ni fer el manteniment dels servidors que necessiten per instal·lar la seva aplicació web, sinó que els poden llogar a les empreses que proporcionen aquests serveis, pagant per l'ús que se'n faci. En el cas concret del PaaS, els servidors que s'ofereixen ja vénen preconfigurats al màxim perquè el desenvolupador pràcticament només hagi de copiar-hi el codi específic de la seva aplicació. Exemples de serveis de PaaS comercials són Heroku i BlueMix, d'IBM; Google App Engine i Elastic Beanstalk, d'Amazon.

Un altre element imprescindible avui dia en el desenvolupament de programari és la utilització d'un sistema de control de versions de programari. Aquests sistemes permeten la compartició de programari entre diferents programadors i, al mateix temps, treballar separadament en diferents versions d'aquest programari. El més habitual és tenir una «branca» principal on hi ha la versió estable del programari. A partir d'aquí, els programadors realitzen els canvis nous en altres branques. Si aquests canvis s'acaben acceptant, les branques noves es fusionen amb la branca principal. Hi ha molts sistemes de control de versions disponibles, però avui dia el més utilitzat és potser Git. Aplicacions web com ara GitHub o Bitbucket permeten sincronitzar i compartir repositoris Git a través del web.

Pel que fa a les eines de proves de programari, els WAF acostumen a integrar-les plenament, automatitzant al màxim l'execució dels jocs de prova. En molts casos, els WAF permeten definir tres «entorns» diferents: desenvolupament, prova i desplegament. Això permet tenir configuracions diferents per als tres entorns, que poden afectar les llibreries que s'utilitzen en cada entorn, les bases de dades, etc. En l'entorn de prova, per exemple, pot interessar tenir una base de dades que sigui independent i no interfereixi en el funcionament normal de la base de dades «real».

L'automatització de l'execució dels jocs de proves ha facilitat l'adopció de la tècnica anomenada *desenvolupament guiat per proves* (*test-driven development*, TDD). Segons aquesta tècnica, els programadors han d'escriure els jocs de proves abans que els programes que els hauran de passar. Llavors els programadors aniran escrivint el codi del programa de manera incremental per anar passant d'un en un cadascun dels jocs de proves, fins que al final ja els hagin superat tots satisfactòriament.

Una eina de proves específica per a aplicacions web és Selenium. Amb aquesta eina, que s'integra en el navegador, es poden «gravar» totes les accions que un usuari realitza sobre una aplicació web: seleccionar enllaços, omplir dades de formularis, etc. Aquestes gravacions es poden integrar en l'entorn de proves de l'aplicació web en forma de joc de proves que «reprodueix» les accions gravades quan són executades. Això és útil per realitzar les anomenades *proves de regressió*, que s'executen per verificar que funcionalitats que ja funcionaven correctament continuen fent-ho després d'haver realitzat canvis en l'aplicació que aparentment no les havien d'afectar.

Agraïments

Vull donar les gràcies a Antoni Olivé per haver-me animat a escriure aquest article, i també pels valuosos comentaris i correccions que ha fet del primer esborrany. ■

Bibliografia

Berners-Lee, Tim (2000). *Weaving the Web: The past, present and future of the World Wide Web by its inventor*. Londres: Texere. ISBN 978-1-58799-018-2.

Fox, Armando; Patterson, David (2013). *Engineering software as a service: An agile approach using cloud computing*. San Francisco: Strawberry Canyon LLC. ISBN 978-0-9848812-4-6.

Kappel, Gerti; Proll, Brigit; Reich, Siegfried; Rischitzegger, Werner (2009). *Web engineering*. Nova Delhi: Wiley India. ISBN 978-81-265-2162-3.

Pressman, Roger; Lowe, David (2008). *Web engineering: A practitioner's approach*. Nova York: McGraw-Hill Education. ISBN 978-0-07-352329-3.