TREBALL DE SÍNTESI FCT

Resolució de casos pràctics en l'entorn laboral

Cicle Formatiu de Grau Superior en Desenvolupament d'Aplicacions Web

Nom de l'alumne/a: Jalel Jordan Delgado
Curs: CFGS Desenvolupament d'Aplicacions Web
Centre educatiu: Institut Puig Castellar
Empresa de pràctiques: WebSolutions Digital
Tutor/a del centre: Ruben Arroyo
Tutor/a de l'empresa: Alex Casanova
Data de lliurament: 30 de juny de 2026

Índex

1. Introducció

Aquest treball de síntesi té com a objectiu demostrar l'aplicació pràctica dels coneixements adquirits durant el Cicle Formatiu de Grau Superior de Desenvolupament d'Aplicacions Web en un entorn laboral real o simulat. El treball se centra en la resolució de casos pràctics relacionats amb el desenvolupament, manteniment, desplegament i seguretat d'aplicacions web.

Durant les pràctiques en empresa, és habitual trobar incidències tècniques que requereixen capacitat d'anàlisi, ús d'eines de diagnosi, presa de decisions i aplicació de solucions adequades. Aquest document recull deu casos pràctics que representen situacions habituals dins d'una empresa dedicada al desenvolupament web.

Cada cas segueix una estructura comuna: descripció del problema, entorn afectat, anàlisi, eines utilitzades, proposta de solució, implementació, resultats obtinguts i mesures de prevenció. Aquesta metodologia permet entendre no només com s'ha resolt cada incidència, sinó també per què s'ha triat una solució concreta i quines accions es poden aplicar per evitar que el problema es repeteixi.

El treball està orientat especialment a l'àmbit del desenvolupament d'aplicacions web, però també inclou aspectes relacionats amb sistemes, servidors, bases de dades, control de versions, seguretat i bones pràctiques professionals.

2. Context de l'empresa

L'empresa escollida per contextualitzar aquest treball és WebSolutions Digital, SL, una empresa fictícia basada en situacions reals pròpies del sector del desenvolupament web.

WebSolutions Digital és una petita empresa dedicada al disseny, desenvolupament i manteniment de pàgines web, botigues en línia i aplicacions web a mida per a clients de diferents sectors. Entre els seus clients hi ha petits comerços, empreses de serveis, centres educatius i professionals autònoms.

L'empresa compta amb un equip reduït format per desenvolupadors frontend, desenvolupadors backend, un responsable de projectes i un tècnic encarregat del manteniment de servidors i desplegaments. Durant el període de pràctiques, l'alumne participa en tasques de suport al desenvolupament, revisió d'incidències, proves, manteniment de webs i documentació tècnica.

2.1. Tipus d'empresa

WebSolutions Digital es pot considerar una empresa petita del sector tecnològic. Tot i la seva mida reduïda, treballa amb diversos projectes simultàniament i necessita aplicar metodologies de treball organitzades per garantir la qualitat dels serveis.

Les tasques principals de l'empresa són:

- Desenvolupament de pàgines web corporatives.
- Creació de botigues en línia.
- Manteniment de webs en WordPress.
- Desenvolupament d'aplicacions web a mida.
- Configuració de dominis i allotjaments.
- Resolució d'incidències tècniques.
- Optimització de rendiment.
- Gestió de còpies de seguretat.
- Desplegament de projectes web.

2.2. Infraestructura tecnològica

L'empresa utilitza una infraestructura formada per ordinadors portàtils per als desenvolupadors, servidors d'allotjament compartit per a webs petites, servidors VPS per a aplicacions més complexes, entorns locals de desenvolupament i repositoris Git per controlar les versions del codi.

Els projectes solen tenir tres entorns diferenciats. L'entorn local s'utilitza per programar i provar canvis en l'ordinador del desenvolupador. L'entorn de proves serveix per revisar funcionalitats abans de publicar-les. L'entorn de producció és la versió final accessible pels clients i usuaris.

Aquesta separació és important perquè permet reduir el risc d'errors. Si una modificació es prova abans en local o en proves, és més fàcil detectar problemes abans que afectin els usuaris finals.

2.3. Tecnologies utilitzades

Durant el treball es fan servir o s'analitzen tecnologies habituals del cicle DAW:

- HTML5.
- CSS3.
- JavaScript.
- PHP.
- Laravel.
- WordPress.
- MySQL.
- Git.
- GitHub.
- Apache.
- Nginx.
- Linux.
- phpMyAdmin.
- Visual Studio Code.
- DevTools del navegador.
- Certificats SSL/TLS.

3. Objectius del treball

L'objectiu principal d'aquest treball és demostrar la capacitat de resoldre problemes tècnics dins d'un entorn laboral relacionat amb el desenvolupament d'aplicacions web.

Els objectius específics són:

- Analitzar incidències tècniques en projectes web.
- Identificar símptomes i possibles causes dels problemes.
- Utilitzar eines adequades per diagnosticar errors.
- Aplicar solucions tècniques de manera justificada.
- Documentar correctament el procés de resolució.
- Valorar alternatives abans d'escollir una solució.
- Comprovar els resultats obtinguts després de la intervenció.
- Proposar mesures preventives.
- Reflexionar sobre els aprenentatges adquirits durant el procés.

Aquest treball també pretén demostrar competències transversals com la capacitat d'organització, la comunicació tècnica, la responsabilitat professional i la resolució autònoma de problemes.

4. Metodologia utilitzada

Per desenvolupar els casos pràctics s'ha seguit una metodologia ordenada basada en el procés habitual de resolució d'incidències tècniques.

En primer lloc, s'identifica el problema a partir dels símptomes observats. A continuació, es recull informació sobre l'entorn afectat, els canvis recents, els missatges d'error i l'impacte sobre els usuaris. Després, es plantegen possibles causes i s'utilitzen eines de diagnosi per confirmar o descartar hipòtesis.

Un cop identificada la causa principal, es defineixen diferents alternatives de solució. Es tria la més adequada tenint en compte criteris com la seguretat, el temps necessari, l'impacte sobre el servei, la facilitat de manteniment i la compatibilitat amb el projecte.

Finalment, s'implementa la solució, es comprova que el problema ha quedat resolt i es proposen mesures de prevenció.

Les fases seguides en cada cas són:

1. Detecció del problema.
2. Recollida d'informació.
3. Anàlisi dels símptomes.
4. Identificació de possibles causes.
5. Ús d'eines de diagnosi.
6. Proposta de solució.
7. Implementació.
8. Validació dels resultats.
9. Prevenció i millores.

Aquesta metodologia permet treballar de manera professional i evita aplicar solucions sense haver entès realment l'origen del problema.

5. Casos pràctics

Cas pràctic 1: Error en el formulari d'inici de sessió

1. Descripció del cas

En una aplicació web interna desenvolupada amb Laravel, diversos usuaris informen que no poden iniciar sessió tot i introduir correctament el seu correu electrònic i contrasenya. El formulari de login no mostra cap missatge clar, simplement recarrega la pàgina i torna a mostrar el formulari buit.

Aquesta incidència afecta els treballadors de l'empresa client, que utilitzen l'aplicació per consultar informació interna i gestionar documents. El problema impedeix l'accés normal a l'eina i genera una interrupció del servei.

2. Entorn afectat

- Aplicació afectada: intranet corporativa.
- Tecnologia principal: Laravel, PHP i MySQL.
- Entorn: producció.
- Usuaris afectats: treballadors interns del client.
- Grau d'impacte: alt.

3. Anàlisi del problema

Els primers símptomes detectats són:

- El formulari de login no permet accedir.
- No apareix cap missatge d'error visible.
- Els usuaris confirmen que les credencials són correctes.
- El problema afecta diversos comptes.
- No hi ha errors evidents a la interfície.

Les possibles causes plantejades són:

- Error en la validació del formulari.
- Problema amb les sessions de PHP.
- Error de connexió amb la base de dades.
- Canvi recent en la configuració de l'aplicació.
- Caducitat o modificació de la clau de l'aplicació.
- Problema amb permisos a la carpeta de sessions.

4. Eines utilitzades per diagnosticar

Per analitzar el problema s'utilitzen les eines següents:

- Consola del navegador.
- DevTools del navegador.
- Logs de Laravel.
- Fitxer `.env`.
- Terminal del servidor.
- phpMyAdmin.
- Comandes bàsiques de Laravel.

Es revisa la consola del navegador per comprovar si hi ha errors JavaScript, però no es detecta cap error relacionat amb el formulari. Després es revisen els logs de Laravel i es troba un missatge relacionat amb la impossibilitat d'escriure fitxers de sessió dins la carpeta `storage/framework/sessions`.

5. Proposta de solució

La solució proposada consisteix a revisar i corregir els permisos de la carpeta `storage` i `bootstrap/cache`, ja que Laravel necessita permisos d'escriptura en aquestes carpetes per gestionar sessions, memòria cau i fitxers temporals.

També es proposa netejar la memòria cau de configuració de Laravel per assegurar que l'aplicació utilitza els valors actualitzats del fitxer `.env`.

Es descarta modificar el codi del formulari perquè no s'han trobat errors en la lògica d'autenticació. També es descarta reiniciar la base de dades, ja que la connexió funciona correctament i els usuaris existeixen a la taula corresponent.

6. Implementació

Els passos aplicats són:

1. Accedir al servidor mitjançant SSH.
2. Revisar els permisos de les carpetes del projecte.
3. Comprovar que l'usuari del servidor web pot escriure dins `storage`.
4. Aplicar permisos correctes a les carpetes necessàries.
5. Executar les comandes de neteja de memòria cau de Laravel.
6. Provar novament l'inici de sessió.
7. Verificar que els usuaris poden accedir correctament.

Les comandes utilitzades són similars a:

php artisan config:clear

php artisan cache:clear

php artisan view:clear

php artisan route:clear

També es revisen permisos mitjançant:

ls -la storage

7. Resultats obtinguts

Després de corregir els permisos, el formulari d'inici de sessió torna a funcionar correctament. Els usuaris poden accedir a la intranet sense errors i les sessions es creen de manera normal.

La incidència queda resolta sense necessitat de modificar el codi de l'aplicació. Això confirma que l'origen del problema estava relacionat amb la configuració del servidor i no amb la programació del formulari.

8. Prevenció i millores

Per evitar que aquesta incidència es repeteixi, es proposen les mesures següents:

- Documentar els permisos necessaris en projectes Laravel.
- Revisar permisos després de cada desplegament.
- Utilitzar scripts de desplegament automàtics.
- Monitorar els logs de l'aplicació.
- Mostrar missatges d'error més clars en entorns interns.
- Separar correctament entorns de proves i producció.

Cas pràctic 2: Problema de connexió amb la base de dades

1. Descripció del cas

En una aplicació web desenvolupada amb PHP i MySQL, apareix un error en carregar qualsevol pàgina que necessita dades de la base de dades. El missatge indica que no es pot establir connexió amb el servidor MySQL.

La incidència apareix després de migrar la web d'un allotjament antic a un nou servidor. El client informa que la pàgina principal carrega parcialment, però les seccions dinàmiques mostren errors.

2. Entorn afectat

- Aplicació afectada: web corporativa amb gestor de continguts propi.
- Tecnologia principal: PHP i MySQL.
- Entorn: producció.
- Grau d'impacte: alt.
- Usuaris afectats: visitants de la web i administradors.

3. Anàlisi del problema

Els símptomes detectats són:

- Error de connexió amb MySQL.
- Les pàgines estàtiques carreguen parcialment.
- Les seccions dinàmiques no mostren contingut.
- El panell d'administració no funciona.
- L'error apareix després de la migració.

Les possibles causes són:

- Credencials incorrectes de base de dades.
- Nom de base de dades diferent al nou servidor.
- Usuari MySQL sense permisos suficients.

- Host incorrecte.
- Servei MySQL aturat.
- Fitxer de configuració no actualitzat.

4. Eines utilitzades per diagnosticar

S'utilitzen aquestes eines:

- Fitxer de configuració PHP.
- phpMyAdmin.
- Terminal del servidor.
- Panell d'allotjament.
- Logs d'errors PHP.
- Comanda `mysql` des de terminal.

Es revisa el fitxer de configuració i es comprova que encara conté les dades de connexió de l'antic servidor. També es comprova que al nou servidor el nom de la base de dades té un prefix diferent.

5. Proposta de solució

La solució consisteix a actualitzar les dades de connexió del fitxer de configuració de l'aplicació:

- Nom de la base de dades.
- Usuari.
- Contrasenya.
- Host.
- Charset.

També cal verificar que l'usuari MySQL tingui permisos suficients sobre la base de dades.

Es descarta modificar consultes SQL perquè el problema es produeix abans d'executar consultes concretes. També es descarta restaurar novament la base de dades, ja que les taules existeixen i són accessibles des de phpMyAdmin.

6. Implementació

Els passos realitzats són:

1. Accedir al panell d'allotjament.
2. Obrir phpMyAdmin i comprovar que la base de dades existeix.
3. Revisar l'usuari MySQL assignat.
4. Obrir el fitxer de configuració de l'aplicació.
5. Actualitzar les credencials.
6. Guardar els canvis.
7. Provar la connexió.
8. Navegar per les pàgines afectades.
9. Accedir al panell d'administració.

Exemple de configuració revisada:

$db_host = 'localhost';

$db_name = 'client_web';

$db_user = 'client_user';

$db_pass = '******';**

7. Resultats obtinguts

Un cop actualitzades les credencials, l'aplicació torna a connectar correctament amb la base de dades. Les pàgines dinàmiques carreguen el contingut i el panell d'administració és accessible.

La incidència queda resolta i es comprova que la migració s'ha completat correctament.

8. Prevenció i millores

Es proposen les següents mesures:

- Crear una checklist de migracions.
- Documentar credencials en un gestor segur.
- Comprovar la connexió a base de dades abans de posar la web en producció.
- Fer proves en un entorn temporal abans del canvi definitiu.
- Fer còpies de seguretat abans i després de la migració.

Cas pràctic 3: Lentitud en una pàgina web corporativa

1. Descripció del cas

Un client informa que la seva pàgina web corporativa triga massa a carregar, especialment des de dispositius mòbils. La web està desenvolupada amb WordPress i utilitza diverses imatges d'alta resolució a la pàgina d'inici.

La lentitud afecta negativament l'experiència d'usuari i pot perjudicar el posicionament SEO del lloc web.

2. Entorn afectat

- Aplicació afectada: web corporativa en WordPress.
- Tecnologia principal: WordPress, PHP, MySQL, JavaScript i CSS.
- Entorn: producció.
- Grau d'impacte: mitjà-alt.
- Usuaris afectats: visitants de la web.

3. Anàlisi del problema

Els símptomes detectats són:

- Temps de càrrega elevat.
- Imatges molt pesades.
- Moltes peticions HTTP.
- Renderització lenta en mòbil.
- Puntuació baixa en eines d'anàlisi de rendiment.

Possibles causes:

- Imatges sense optimitzar.
- Massa plugins actius.
- Fitxers CSS i JavaScript no minificats.
- Absència de memòria cau.
- Allotjament amb recursos limitats.
- Consultes lentes a la base de dades.

4. Eines utilitzades per diagnosticar

Per diagnosticar el problema s'utilitzen:

- Google Lighthouse.
- DevTools del navegador.
- Eina Network del navegador.
- Panell de WordPress.
- Plugin d'anàlisi de rendiment.
- phpMyAdmin.

S'observa que diverses imatges superen 2 MB i que es carreguen encara que no siguin visibles inicialment. També es detecta que no hi ha cap sistema de memòria cau actiu.

5. Proposta de solució

La solució proposada consisteix en:

- Comprimir i redimensionar les imatges.
- Convertir imatges a format WebP.
- Activar càrrega diferida o lazy loading.
- Instal·lar i configurar un sistema de memòria cau.
- Desactivar plugins innecessaris.
- Minificar fitxers CSS i JavaScript.
- Revisar la pàgina d'inici per reduir elements pesats.

Es descarta canviar immediatament de servidor perquè primer cal optimitzar els recursos de la web. Si després de l'optimització el rendiment continua sent baix, es podria valorar una millora de l'allotjament.

6. Implementació

Els passos seguits són:

1. Fer una còpia de seguretat de la web.
2. Analitzar la pàgina amb Lighthouse.
3. Exportar les imatges més pesades.
4. Redimensionar-les segons la mida real necessària.
5. Comprimir-les i convertir-les a WebP.
6. Substituir les imatges antigues.
7. Activar lazy loading.
8. Configurar memòria cau.
9. Desactivar plugins no utilitzats.
10. Repetir l'anàlisi de rendiment.

7. Resultats obtinguts

Després de l'optimització, la pàgina redueix notablement el temps de càrrega. Les imatges passen de pesar diversos megabytes a uns pocs centenars de kilobytes. També disminueix el nombre de peticions i millora la puntuació de rendiment en dispositius mòbils.

Els usuaris poden navegar de manera més fluida i el client percep una millora clara en la rapidesa de la web.

8. Prevenció i millores

Per evitar problemes similars es proposa:

- Optimitzar imatges abans de pujar-les.
- Utilitzar formats moderns com WebP.
- Revisar periòdicament els plugins instal·lats.
- Mantenir activa la memòria cau.
- Fer auditories de rendiment cada cert temps.
- Formar el client perquè no pugi imatges massa grans.

Cas pràctic 4: Error en el desplegament d'una aplicació web

1. Descripció del cas

Una aplicació desenvolupada en Laravel funciona correctament en l'entorn local, però en desplegar-la al servidor de producció mostra un error 500. Aquest tipus d'error és genèric i no mostra informació detallada a l'usuari final.

La incidència es produeix just després d'un desplegament de nova versió. L'aplicació queda temporalment inaccessible.

2. Entorn afectat

- Aplicació afectada: portal de gestió de reserves.
- Tecnologia principal: Laravel, PHP, MySQL.
- Entorn: producció.
- Grau d'impacte: alt.
- Usuaris afectats: clients i administradors.

3. Anàlisi del problema

Els símptomes són:

- Error 500 en accedir a la web.
- L'aplicació funcionava abans del desplegament.
- No hi ha canvis visibles a la interfície.
- El problema no es reprodueix en local.

Possibles causes:

- Dependències no instal·lades.
- Fitxer `.env` incorrecte.
- Permisos incorrectes.
- Memòria cau antiga.
- Versió de PHP incompatible.
- Migracions de base de dades pendents.

4. Eines utilitzades per diagnosticar

S'utilitzen:

- Logs del servidor.
- Logs de Laravel.
- Terminal SSH.
- Git.
- Composer.
- Fitxer `.env`.
- Comandes `php artisan`.

Als logs es detecta que falta una dependència instal·lada en local però no en producció. També es comprova que després del desplegament no s'havia executat `composer install`.

5. Proposta de solució

La solució consisteix a executar el procés complet de desplegament:

- Instal·lar dependències amb Composer.
- Revisar variables d'entorn.
- Executar migracions si cal.
- Netejar memòria cau.
- Comprovar permisos.
- Reiniciar serveis si és necessari.

Es descarta modificar el codi perquè el mateix codi funciona correctament en local. El problema està relacionat amb el procés de desplegament incomplet.

6. Implementació

Passos realitzats:

1. Accedir al servidor per SSH.
2. Entrar a la carpeta del projecte.
3. Revisar l'últim commit desplegat.
4. Executar:

composer install --no-dev --optimize-autoloader

5. Revisar el fitxer `.env`.
6. Executar:

php artisan config:clear

php artisan cache:clear

php artisan route:clear

php artisan migrate

7. Revisar permisos de `storage`.
8. Provar l'aplicació.

7. Resultats obtinguts

Després d'instal·lar les dependències i netejar la memòria cau, l'aplicació torna a funcionar. L'error 500 desapareix i els usuaris poden accedir correctament al portal.

La incidència mostra la importància de seguir un procés de desplegament ordenat i documentat.

8. Prevenció i millores

Mesures proposades:

- Crear un script de desplegament.
- Utilitzar pipelines d'integració contínua.

- Documentar tots els passos necessaris.
- Fer desplegaments primer en entorn de proves.
- Guardar una versió anterior per poder fer rollback.
- Comprovar compatibilitat de versions de PHP i dependències.

Cas pràctic 5: Conflicte de versions amb Git

1. Descripció del cas

Durant el desenvolupament d'una nova funcionalitat, dos desenvolupadors modifiquen el mateix fitxer en branques diferents. Quan s'intenta fusionar una branca amb la branca principal, Git detecta un conflicte.

El conflicte impedeix completar la fusió fins que es resolgui manualment.

2. Entorn afectat

- Projecte afectat: aplicació web de gestió de clients.
- Tecnologia principal: Git, GitHub, JavaScript.
- Entorn: desenvolupament.
- Grau d'impacte: mitjà.
- Usuaris afectats: equip de desenvolupament.

3. Anàlisi del problema

Símptomes:

- Git mostra un missatge de conflicte.
- El fitxer afectat conté marques especials.
- No es pot completar el merge automàticament.
- Les dues branques han modificat les mateixes línies.

Possibles causes:

- Manca de comunicació entre desenvolupadors.
- Branques massa antigues.
- Modificació simultània del mateix component.
- Absència de revisió prèvia abans de fusionar.

4. Eines utilitzades per diagnosticar

S'utilitzen:

- Terminal.

- Git.
- Visual Studio Code.
- GitHub.
- Historial de commits.

Git indica quin fitxer està en conflicte. En obrir-lo amb Visual Studio Code, es veuen les dues versions del codi i les marques `<<<<<<<`, `=======` i `>>>>>>>`.

5. Proposta de solució

La solució consisteix a revisar manualment les dues versions del codi i decidir quins canvis s'han de conservar. En alguns casos, cal combinar parts de les dues versions.

Es descarta acceptar automàticament una versió sense revisar-la, perquè això podria eliminar funcionalitats desenvolupades per un altre membre de l'equip.

6. Implementació

Passos realitzats:

1. Executar `git status`.
2. Identificar el fitxer en conflicte.
3. Obrir el fitxer amb Visual Studio Code.
4. Comparar les dues versions.
5. Combinar manualment els canvis necessaris.
6. Eliminar les marques de conflicte.
7. Executar proves locals.
8. Afegir el fitxer corregit amb `git add`.
9. Completar el merge amb un commit.

Comandes utilitzades:

git status

git add .

git commit

7. Resultats obtinguts

El conflicte es resol correctament i la branca es pot fusionar amb la branca principal. Les funcionalitats dels dos desenvolupadors es conserven i el projecte continua funcionant.

8. Prevenció i millores

Mesures preventives:

- Fer pull freqüentment de la branca principal.
- Crear branques petites i de curta durada.

- Comunicar canvis importants a l'equip.
- Revisar pull requests abans de fusionar.
- Dividir components grans en fitxers més petits.
- Utilitzar una metodologia clara de treball amb Git.

Cas pràctic 6: Problema de permisos en el servidor

1. Descripció del cas

Una aplicació web permet pujar fitxers PDF des d'un formulari d'administració. Després d'una migració de servidor, els administradors informen que no poden pujar documents. L'aplicació mostra un error indicant que no es pot guardar el fitxer.

2. Entorn afectat

- Aplicació afectada: gestor documental web.
- Tecnologia principal: PHP, Linux, Apache.
- Entorn: producció.
- Grau d'impacte: mitjà-alt.
- Usuaris afectats: administradors.

3. Anàlisi del problema

Símptomes:

- Error en pujar fitxers.
- La resta de l'aplicació funciona.
- El problema apareix després d'una migració.
- No es creen nous fitxers a la carpeta de pujades.

Possibles causes:

- Permisos incorrectes a la carpeta.
- Propietari incorrecte.
- Límit de mida de fitxer massa baix.
- Configuració incorrecta de PHP.
- Ruta de carpeta incorrecta.

4. Eines utilitzades per diagnosticar

S'utilitzen:

- Terminal SSH.
- Logs d'Apache.
- Logs PHP.
- Comanda `ls -la`.

- Fitxer `php.ini`.
- Formulari de pujada.

Es comprova que la carpeta `uploads` existeix, però pertany a un usuari diferent i Apache no té permisos d'escriptura.

5. Proposta de solució

La solució consisteix a assignar el propietari i permisos correctes a la carpeta de pujades. També es revisa que la configuració de PHP permeti la mida màxima dels fitxers necessaris.

Es descarta modificar el formulari perquè el problema no està en la validació ni en el frontend, sinó en l'escriptura al sistema de fitxers.

6. Implementació

Passos realitzats:

1. Accedir al servidor.
2. Localitzar la carpeta de pujades.
3. Revisar permisos.
4. Canviar propietari o grup si cal.
5. Aplicar permisos d'escriptura adequats.
6. Fer una prova de pujada.
7. Revisar que el fitxer es guarda correctament.

Exemple de revisió:

ls -la uploads

Exemple d'ajust de permisos:

chmod 755 uploads

En funció del servidor, també pot ser necessari ajustar el propietari amb `chown`.

7. Resultats obtinguts

Després de corregir els permisos, els administradors poden tornar a pujar fitxers PDF. Els documents es guarden correctament i es poden consultar des de l'aplicació.

8. Prevenció i millores

Mesures proposades:

- Revisar permisos després de migracions.
- Documentar propietaris i permisos necessaris.
- Evitar permisos massa oberts com 777.
- Afegir validació clara d'errors de pujada.

- Configurar alertes en cas d'errors repetits.
- Incloure aquesta comprovació en la checklist de desplegament.

Cas pràctic 7: Restauració d'una còpia de seguretat

1. Descripció del cas

Un client elimina accidentalment diverses entrades del seu blog corporatiu des del panell d'administració. Demana recuperar el contingut eliminat, ja que algunes entrades tenien valor SEO i rebien visites.

La web està feta amb WordPress i disposa de còpies de seguretat automàtiques diàries.

2. Entorn afectat

- Aplicació afectada: blog corporatiu en WordPress.
- Tecnologia principal: WordPress, MySQL.
- Entorn: producció.
- Grau d'impacte: mitjà.
- Usuaris afectats: client i visitants.

3. Anàlisi del problema

Símptomes:

- Diverses entrades han desaparegut.
- El client confirma que les va eliminar per error.
- Algunes no es poden recuperar des de la paperera.
- La web continua funcionant.

Possibles causes:

- Eliminació manual accidental.
- Paperera buidada.
- Error d'un plugin.
- Restauració parcial necessària.

4. Eines utilitzades per diagnosticar

S'utilitzen:

- Panell de WordPress.
- phpMyAdmin.
- Sistema de còpies de seguretat.
- Panell d'allotjament.
- Base de dades MySQL.

Es comprova que les entrades no es troben a la paperera de WordPress. Per tant, cal recuperar-les des d'una còpia de seguretat anterior.

5. Proposta de solució

La solució consisteix a restaurar una còpia de seguretat en un entorn temporal, extreure les entrades eliminades i importar-les de nou a la web en producció.

Es descarta restaurar tota la web directament perquè això podria eliminar canvis recents realitzats després de la còpia. La restauració parcial és més segura.

6. Implementació

Passos realitzats:

1. Localitzar una còpia de seguretat anterior a l'eliminació.
2. Crear un entorn temporal de restauració.
3. Restaurar la base de dades en aquest entorn.
4. Accedir al WordPress restaurat.
5. Exportar les entrades necessàries.
6. Importar-les a la web de producció.
7. Revisar categories, imatges i enllaços.
8. Comprovar que les URL funcionen.

7. Resultats obtinguts

Les entrades eliminades es recuperen correctament sense afectar els canvis recents de la web. El client pot tornar a consultar i editar els articles.

La solució evita una restauració completa que hauria pogut provocar pèrdua d'informació nova.

8. Prevenció i millores

Mesures preventives:

- Mantenir còpies de seguretat automàtiques.
- Fer còpies abans de canvis importants.
- Limitar permisos d'usuaris no tècnics.
- Formar el client sobre l'ús de la paperera.
- Revisar plugins que puguin afectar continguts.
- Documentar el procés de restauració.

Cas pràctic 8: Configuració d'HTTPS en una web de client

1. Descripció del cas

Un client informa que el navegador mostra un avís indicant que la seva web no és segura. La web carrega amb HTTP i no disposa de certificat SSL actiu. Això pot generar desconfiança en els usuaris i afectar el posicionament als cercadors.

2. Entorn afectat

- Aplicació afectada: web corporativa.
- Tecnologia principal: Apache, domini, certificat SSL.
- Entorn: producció.
- Grau d'impacte: mitjà-alt.
- Usuaris afectats: visitants de la web.

3. Anàlisi del problema

Símptomes:

- El navegador mostra "No segur".
- La web carrega amb `http://`.
- No hi ha redirecció automàtica a HTTPS.
- Alguns formularis envien dades sense connexió segura.

Possibles causes:

- Certificat SSL no instal·lat.
- Certificat caducat.
- Redirecció HTTPS no configurada.
- Contingut mixt.
- Configuració incorrecta del servidor.

4. Eines utilitzades per diagnosticar

S'utilitzen:

- Navegador web.
- Panell d'allotjament.
- Configuració del domini.
- Fitxer `.htaccess`.
- Eines de comprovació SSL.
- DevTools del navegador.

Es comprova que el certificat no està actiu i que la web no força la redirecció cap a HTTPS.

5. Proposta de solució

La solució consisteix a instal·lar un certificat SSL, activar HTTPS i configurar una redirecció permanent des d'HTTP cap a HTTPS.

També cal revisar si hi ha contingut mixt, és a dir, recursos carregats encara amb HTTP, com imatges, scripts o fulls d'estil.

Es descarta deixar la web amb HTTP perquè no és recomanable en termes de seguretat, confiança i posicionament.

6. Implementació

Passos realitzats:

1. Accedir al panell d'allotjament.
2. Activar un certificat SSL per al domini.
3. Esperar la validació del certificat.
4. Comprovar que la web carrega amb HTTPS.
5. Configurar la redirecció des d'HTTP.
6. Revisar contingut mixt.
7. Actualitzar URLs internes si cal.
8. Provar formularis i navegació.

Exemple de redirecció en `.htaccess`:

RewriteEngine On

RewriteCond %{HTTPS} off

RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

7. Resultats obtinguts

Després de la configuració, la web carrega correctament amb HTTPS. El navegador ja no mostra l'avís de "No segur" i les dades enviades pels formularis viatgen xifrades.

També es comprova que les pàgines internes redirigeixen correctament a la versió segura.

8. Prevenció i millores

Mesures proposades:

- Activar renovació automàtica del certificat.
- Revisar periòdicament la validesa SSL.
- Utilitzar sempre HTTPS en nous projectes.
- Configurar redireccions des del primer desplegament.
- Evitar contingut mixt.
- Incloure la comprovació SSL en la checklist de publicació.

6. Prevenció i millores generals

Després d'analitzar els deu casos pràctics, es poden extreure diverses mesures generals de prevenció aplicables a qualsevol empresa de desenvolupament web.

En primer lloc, és fonamental disposar d'una bona documentació tècnica. Moltes incidències es poden resoldre més ràpidament si existeixen documents que expliquen com desplegar un projecte, quins permisos necessita, quines variables d'entorn utilitza o com es fan les còpies de seguretat.

També és molt important separar correctament els entorns de treball. No és recomanable aplicar canvis directament a producció sense haver-los provat abans en local o en un entorn de proves. Aquesta separació redueix el risc d'errors greus i permet validar les funcionalitats abans que arribin als usuaris finals.

Una altra mesura bàsica és utilitzar control de versions amb Git. Git permet registrar l'historial de canvis, recuperar versions anteriors, treballar amb branques i col·laborar amb altres desenvolupadors. Tot i això, cal seguir bones pràctiques per evitar conflictes i desorganització.

La seguretat també ha de ser una prioritat. Formularis, APIs, panells d'administració i bases de dades han d'estar protegits. Cal validar dades, escapar sortides, gestionar correctament permisos, utilitzar HTTPS i mantenir actualitzades les dependències.

Pel que fa al rendiment, és recomanable revisar periòdicament la velocitat de càrrega de les webs. Imatges massa grans, plugins innecessaris o consultes lentes poden afectar negativament l'experiència d'usuari. Eines com Lighthouse o DevTools ajuden a detectar aquests problemes.

Finalment, les còpies de seguretat són imprescindibles. Qualsevol projecte en producció hauria de tenir còpies automàtiques i comprovades. No n'hi ha prou amb generar còpies: també cal saber restaurar-les correctament quan sigui necessari.

Les principals millores generals proposades són:

- Crear checklists de desplegament.
- Documentar configuracions importants.
- Automatitzar processos repetitius.
- Fer còpies de seguretat periòdiques.
- Revisar permisos del servidor.
- Mantenir dependències actualitzades.
- Fer proves abans de publicar canvis.
- Utilitzar HTTPS en tots els projectes.
- Monitorar logs i errors.
- Formar usuaris i clients.
- Aplicar bones pràctiques de seguretat.
- Revisar rendiment de manera periòdica.

7. Conclusions

Aquest treball de síntesi m'ha permès aplicar coneixements del cicle formatiu de Desenvolupament d'Aplicacions Web a situacions pràctiques pròpies d'un entorn laboral. Els casos analitzats mostren que el desenvolupament web no consisteix únicament a escriure codi, sinó també a mantenir aplicacions, diagnosticar errors, gestionar servidors, protegir dades i garantir una bona experiència d'usuari.

Una de les idees més importants apreses és que abans d'aplicar una solució cal entendre bé el problema. Moltes incidències poden semblar errors de programació, però en realitat poden estar relacionades amb permisos, configuració del servidor, credencials incorrectes, dependències no instal·lades o processos de desplegament incomplets.

També he comprovat la importància dels logs i de les eines de diagnosi. Eines com DevTools, Postman, phpMyAdmin, Git, la terminal o els registres del servidor permeten obtenir informació objectiva sobre l'origen d'una incidència.

Pel que fa a la seguretat, els casos relacionats amb XSS, autenticació d'APIs i HTTPS mostren que qualsevol aplicació web ha de protegir correctament les dades dels usuaris. La validació, el sanejament, l'escapament de dades i l'ús de connexions segures són pràctiques indispensables.

Un altre aprenentatge important és la necessitat de documentar els processos. Quan una empresa disposa de procediments clars per desplegar, restaurar còpies, configurar permisos o resoldre errors habituals, es redueix el temps de resposta i es minimitzen riscos.

Les principals dificultats trobades durant l'elaboració del treball han estat identificar causes reals entre diverses possibilitats, justificar tècnicament cada decisió i ordenar la informació de manera clara. Tot i això, aquestes dificultats formen part del procés professional i ajuden a desenvolupar una mentalitat més analítica.

Com a valoració final, aquest treball m'ha ajudat a consolidar coneixements tècnics i a entendre millor com es treballa en una empresa de desenvolupament web. La resolució de casos pràctics és una forma molt útil de demostrar competències, ja que obliga a combinar teoria, pràctica, criteri tècnic i capacitat de comunicació.

8. Bibliografia

Per elaborar aquest treball s'han consultat coneixements adquirits durant el cicle formatiu i documentació tècnica relacionada amb les tecnologies utilitzades.

Fonts recomanades:

- Mozilla Developer Network. Documentació sobre HTML, CSS i JavaScript.
- Documentació oficial de PHP.

- Documentació oficial de Laravel.
- Documentació oficial de MySQL.
- Documentació oficial de Git.
- Documentació oficial de WordPress.
- Documentació d'Apache HTTP Server.
- Materials i apunts del CFGS de Desenvolupament d'Aplicacions Web.
- Documentació interna de processos de desplegament i manteniment web.

9. Annexos

Annex 1. Exemple de checklist de desplegament

Abans de desplegar una aplicació web a producció, es recomana revisar:

- El codi està pujat al repositori.
- La branca correcta està actualitzada.
- Les dependències estan instal·lades.
- El fitxer d'entorn està configurat.
- La connexió amb la base de dades funciona.
- Les migracions s'han executat.
- Els permisos de carpetes són correctes.
- La memòria cau s'ha netejat.
- L'aplicació s'ha provat en entorn de proves.
- Existeix una còpia de seguretat recent.

Annex 2. Eines utilitzades

Durant els casos pràctics s'han utilitzat o mencionat les eines següents:

- Visual Studio Code.
- DevTools del navegador.
- Google Lighthouse.
- Postman.
- phpMyAdmin.
- Terminal Linux.
- Git.
- GitHub.
- Panell d'allotjament.
- Logs de servidor.
- Logs de Laravel.
- WordPress.

Annex 3. Comandes habituals

Algunes comandes útils en els casos pràctics són:

Git

git status

git pull

git add .

git commit -m "Missatge del commit"

git push

PHP

php artisan cache:clear

php artisan config:clear

php artisan route:clear

php artisan view:clear

Ternimal (Sistema de carpetes)

ls -la

chmod 755 carpeta

Annex 4. Bones pràctiques generals

- No treballar directament sobre producció si es pot evitar.
- Fer còpies de seguretat abans de canvis importants.
- Utilitzar noms clars en commits i branques.
- Documentar incidències i solucions.
- Mantenir actualitzades les dependències.
- Revisar logs quan apareix un error.
- Utilitzar HTTPS en totes les webs.
- Comprovar permisos després de migracions.
- Optimitzar imatges abans de pujar-les.
- Comunicar canvis importants a l'equip.
- Provar restauracions de còpies de seguretat.
- Fer proves després de cada desplegament.