UNIVERSITAT POLITÈCNICA DE CATALUNYA

Implementació d'una plataforma web per al suport del muntatge dels productes d'AUSA

1 de juny de 2016

Memòria del projecte que presenta Daniel Varo Garcia
sota la direcció Marta Isabel Tarrés Puertas
per assolir el grau d'Enginyer en Sistemes TIC.

"La programació és una carrera entre els desenvolupadors, intentant construir majors i millors programes a prova d'idiotes, i l'Univers, intentant produir majors i millors idiotes. Per ara va guanyant l'Univers." – Rich Cook

Índex

Índex de figures

Índex de taules

Abstract

This project consist in implement a website for AUSA Center S.L.U. Manresa enterprise. AUSA's workers can access and view a series of videos to learn how to assemble the various machines of the company.
To implement this project, the work will be divided into two parts:

- The first part, where studied the different tools can be used to develop the website, choosing which of them will be used for the implementation.
- A second part, where the tools have been chosen, explain the steps to do the implementation of the website.

Resum

Aquest projecte consisteix en implementar un lloc web per l'empresa AUSA Center S.L.U. de Manresa, perquè els treballadors hi puguin accedir i visualitzar una sèrie de vídeos, per aprendre com es munten les diferents màquines de l'empresa.
Per dur a terme aquest projecte, la feina es dividirà en dues parts:

- Una primera part, on s'estudiaran les diferents eines que es poden utilitzar per desenvolupar el lloc web, escollint quines d'elles es faran servir per la seva implementació.
- Una segona part, on amb les eines escollides, s'explicaran els passos que s'han dut a terme per poder fer la implementació del lloc web.

Introducció

Des de l'empresa AUSA Center S.L.U. de Manresa van detectar que el sistema que tenien per ensenyar el muntatge de les màquines als seus treballadors estava antiquat. Aquest sistema consistia en que, en cada lloc de treball, per cada màquina, hi havia una sèrie de documents impresos amb les instruccions que s'havien de fer el muntatge d'aquella màquina. Aquests sistema no era òptim, ja que les instruccions es quedaven antiquades de manera molt ràpida, i per actualitzar-les, s'havien de tornar a imprimir documents per tots els llocs de treball.
Detectant aquest problema, van decidir replantejar-se tot el sistema actual, i van contactar amb el departament EMIT de l'Escola Politècnica Superior d'Enginyeria de Manresa per trobar una solució òptima i actual aquest problema.
En aquest document es presentarà la solució que es va trobar aquest problema. Aquesta solució consisteix en implementar un lloc web, perquè els treballadors puguin accedir des de qualsevol lloc on es trobin, i visualitzar una sèrie de vídeos a mode de tutorial per poder aprendre com es munten les diferents màquines. D'aquesta manera, els treballadors al ser contractats o designats a un lloc de treball nou, poden aprendre com es munta una màquina de manera ràpida i interactiva. A més a més, el lloc web tindrà una part d'administració des de on es podran afegir, modificar o eliminar els tutorials de les màquines de manera ràpida i senzilla.
En primer lloc, en aquest document s'investigaran els conceptes teòrics de les diferents eines que es poden utilitzar per desenvolupar el projecte, escollint quines d'elles faré servir per implementar el projecte.
En segon lloc, un cop escollides les eines que utilitzaré, explicaré els passos que he dut a terme per fer la implementació del lloc web.
En tercer lloc, faré un manual d'usuari per el lloc web, per poder interactuar amb ell sense cap tipus de problema.
En quart lloc, presentaré els problemes que he tingut d'afrontar a l'hora de desenvolupar el projecte, i les decisions que he pres per solucionar-los.
En cinquè lloc, explicaré els requeriments que han de tenir els sistemes perquè els usuaris puguin interactuar amb el lloc web de manera totalment òptima.
En sisè lloc, exposaré les diferents millores que podrien aplicar-se al lloc web.
I per últim, hi haurà un apartat d'annexos per acabar de completar el document.

Part I.

Memòria

1. Conceptes Teòrics

Per poder desenvolupar aquest projecte, el primer que vaig haver de fer va ser identificar:

- Les diferents maneres que es poden crear tant el *backend* com el *frontend*.
- Les diferents bases de dades, per veure quina s'adapta millor per la implementació d'aquest projecte.
- Llenguatges amb els que es pot escriure una pàgina web.
- Els diferents servidors on podria emmagatzemar el projecte.

A continuació, explicaré les eines que he escollit, i les raons que m'han portat a fer aquesta elecció.

1.1. Backend

El *backend* és la part de software que processa l'entrada de dades que s'efectua des del *frontend* [Wik16g]. Per tant, es pot dir que són els processos per resoldre les peticions dels usuaris. Tenint en compte la funció que duu a terme, puc determinar que el *backend* es troba en la part del servidor.
Les funcions que té el *backend* que més s'utilitzen són:

1) Permetre pujada d'arxius al servidor
2) Proporcionar una interfície a l'administrador per gestionar fàcilment el lloc web
3) Proporcionar autentificació i gestió d'usuaris
4) Interactuar amb la base de dades i mostrar a l'usuari les dades que ha demanat
5) Juntament amb el *frontend*, interactuar amb un sistema web o software per resoldre les peticions dels usuaris

Per el desenvolupament de la part de *backend*, faré servir un *framework web* i una base de dades.

1.1.1. Framework Web

Per desenvolupar la pàgina web he decidit fer servir un *framework web*. Un *framework web* és un conjunt de components que t'ajuden a desenvolupar pàgines web d'una manera ràpida i senzilla.
Per entendre perquè he decidit fer servir un *framework web*, cal tenir en compte que m'ofereix una estructura sòlida, organitzada i eficient per el desenvolupament de l'aplicació web. Enlloc de tornar a reinventar la roda, és a dir, tornar a estructurar des de zero l'aplicació web, el

que ofereix el *framework web* és una estructura estàndard que s'adapta a les necessitats de l'aplicació web que vulgui desenvolupar. Així, no he de preocupar-me de l'estructura del projecte, ja que d'això ja se'n encarrega el *framework*.
A part d'aquest motiu, també he escollit fer servir un *framework web* perquè hem permet estalviar-me temps en desenvolupar mòduls que es repeteixen en la majoria d'aplicacions web, i així poder centrar-me en altres àrees del projecte més crítiques.

1.1.2. Elecció del Framework Web

Un cop presa la decisió d'utilitzar un *framework web*, vaig investigar diferents tipus, per veure quin d'ells s'adaptaria millor al meu projecte. Els *frameworks web* que vaig investigar van ser:

1) Django

2) AngularJS

3) Pylons

4) Laravel

1.1.2.1. Django

Django és un *framework* d'aplicacions web de codi lliure i obert, escrit en `Python`, que segueix l'arquitectura MVC (Model-Vista-Controlador) [Wik16e].
Django el que fa és proporcionar components "prefabricats"que puc utilitzar per construir el lloc web d'una manera senzilla. Les característiques principals de Django són les següents:

- Les aplicacions que crea es poden instal·lar en qualsevol pàgina gestionada per Django.
- Una `API` de base de dades robusta.
- Un sistema incorporat de vistes genèriques que estalvia tenir d'escriure la lògica en tasques comuns.
- Un sistema extensible de plantilles basat en etiquetes, amb herència de plantilles.
- Un sistema `middleware` per desenvolupar característiques addicionals com memòria cau, compressió a la sortida, normalització de `URLs`, protecció `CSRF` (protecció de falsificacions de petició en llocs creuats) i suport de sessions.
- Suport de internacionalització, i també suport a traduccions incorporades en la interfície d'administració.

L'estructura d'arxius que té un projecte en Django és la següent:

```
manage.py
recetario/
    __init__.py
    settings.py
    urls.py
    wsgi.py
    views.py
```

```
└── principal/
    ├── migrations/
    ├── models.py
    ├── urls.py
    ├── __init__.py
    ├── admin.py
    └── views.py
```

manage.py Fitxer principal que configura i retorna l'aplicació `WSGI`.

recetario És el projecte que he creat.

settings.py Fitxer on es fa tota la configuració per al projecte.

urls.py Fitxer on es fa la definició de les `URLs` del projecte.

principal És l'aplicació que es crea per el projecte.

models.py Fitxer on es defineix les taules de la base de dades.

admin.py Fitxer on es defineix el que es veurà de la base de dades en la pàgina d'administració de la web.

migrations Directori que fa la sincronització entre el projecte Django i la base de dades, perquè en ella hi hagi totes les taules necessàries, i en cadascuna d'elles, hi hagi els atributs corresponents.

views.py Fitxer on es defineix el que s'ha de visualitzar en els `templates`. Aquest fitxer és el que fa que els `templates` puguin ser dinàmics.

L'arquitectura que segueix Django, com he comentat anteriorment, és el patró MVC, que consisteix en:

- **Model** És on es descriuen les dades
- **Vista** És on es controla el què poden veure els usuaris
- **Controlador** És el que controla el què s'ha de carregar al navegador de l'usuari depenent de la `url` i els paràmetres que s'hagin enviat.

Gràficament, el patró MVC és el següent:

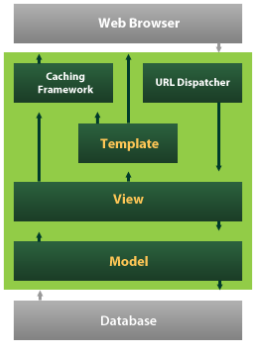

Figura 1.1.: Arquitectura de Django

El funcionament de l'arquitectura MVC és el següent:

1) L'`URL dispatcher` mapeja la petició d'una URL d'una vista i la crida. Si l'emmagatzematge en la memòria cau està habilitat, la funció `view` comprova si hi ha una versió en al memòria cau de la pàgina. Si és així, passa per alt totes les mesures addicionals i recupera la versió que es troba a la memòria cau.

2) La funció `view` duu a terme l'acció que s'ha sol·licitat, que normalment implica la lectura o escriptura de la base de dades, tot i que també pot incloure altres tasques.

3) El `model` defineix les dades de `Python` i interactua amb elles. Tot i que normalment es troba en una base de dades relacional, també és possible fer servir altres mecanismes.

4) Després de realitzar les tasques sol·licitades, la `view` retorna una resposta en forma d'objecte `HTTP` al navegador web.

5) Els `templates` solen retornar pàgines `HTML`. Django ofereix una sintaxi de llenguatge pels `templates` molt senzilla d'aprendre però que proporciona molta potència a la lògica de presentació.

Django utilitza objectes de petició i de resposta per passar l'estat a través del sistema. Quan se sol·licita una pàgina, Django crea un objecte `HttpRequest` que conté metadades sobre la sol·licitud. Tot seguit, Django carrega la `view` apropiada, passant el `HttpRequest` com a primer argument de la funció `view`. Cada view és responsable de retornar un objecte `HttpResponse`.

El cicle que segueix Django cada cop que fem una petició és el següent:

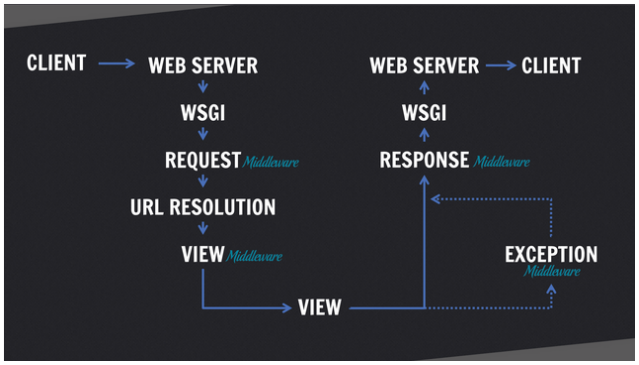

Figura 1.2.: Seqüència petició i resposta

Web server i WSGI Quan s'entra al navegador i carrega una pàgina, el que s'està fent és una petició a un servidor d'internet, que ha de saber què fer amb el que l'usuari li ha enviat. Aquesta feina la fa el `Web Sever Gateway Interface` (WSGI). Per tant, la feina del WSGI és agafar la sol·licitud i tornar la resposta corresponent.

Middleware Si ens fixem en l'imatge anterior, es pot veure que la paraula `Middleware` apareix diverses vegades entre les dues parts WSGI. El Middleware de Django és un *plug-in* que modifica lleugerament les peticions que venen i les respostes que surten.
El Middleware pot ser de cinc maneres diferents:

1) Request Middleware
2) View Middleware
3) Error Middleware
4) Template response Middleware
5) Response Middleware

La seva finalitat dependrà de es defineixi el Middleware en els fitxers de configuració. Suposant que defineixo el Middleware segons el següent ordre:

1) CommonMiddleware
2) SessionMiddleware
3) CsrfViewMiddleware
4) AuthenticationMiddleware
5) MessageMiddleware

L'ordre d'execució serà el que es mostra en la imatge següent:

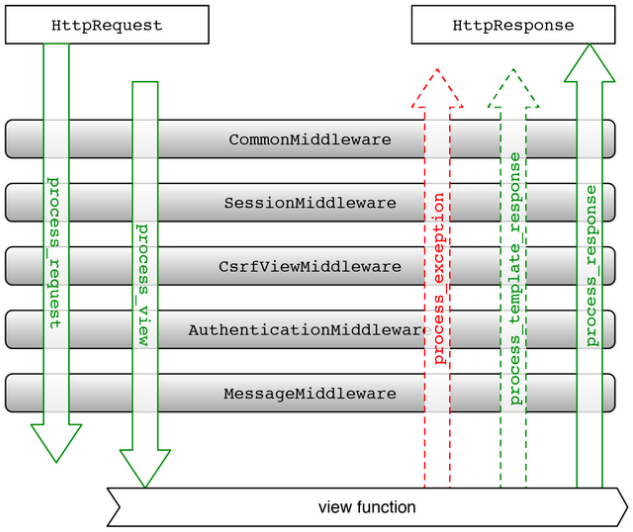

Figura 1.3.: Ordre d'execució del Middleware

Request Middleware El Request Middleware s'executa en cada petició que arriba abans que la sol·licitud sigui dirigida a la `view`. En aquesta part no sap que passarà amb aquesta sol·licitud, ja que al no haver-hi resolució de URL encara no sap a quina view ha d'anar. Tot el que sap el Middleware és que aquesta petició l'ha fet l'usuari i es transmet de la capa WSGI.

URL Resolution Aquesta és la part on Django mira la URL associada amb la sol·licitud i sap a quina view ha d'anar.

View Les views és on puc ajuntar dades de models i templates per crear les pàgines que els usuaris veuran. La view sempre accepta un objecte de petició i retorna un objecte de resposta, normalment un template.

Camí de tornada Si es produís un error o excepció en el codi, a continuació, s'executaria l'`Exception Middleware`. Si no s'ha aixecat cap excepció o error, la resposta generada des de la view es retorna a la capa WSGI, que va cap al servidor i torna cap a nosaltres.

1.1.2.2. AngularJS

AngularJS és un *framework* d'aplicacions web de codi obert, escrit en `Javascript`, que segueix l'arquitectura `MVC` i fa servir objectes de petició i resposta `HttpRequest` [Wik16a], les dues explicades com funcionen en l'apartat 1.1.2.1.
Les característiques principals d'AngularJS són les següents [HTM16]:

1) Està contínuament observant els canvis que es produeixen tant en la `views`, com en el `model` i sincronitza les dades entre els dos.

2) Qualsevol valor que tingui de visualitzar-se en la `views`, es col·loca dintre de dues claus dobles {{ *expressió o valor* }}

3) Les `directives` ens permeten crear els nostres propis elements

L'estructura d'arxius que té un projecte en AngularJS és la següent [Ang16]:

```
app.js
index.html
index.js
main.html
main.js
common/
    directives/
        directive1.js
        directive2.js
    filters/
        filter1.js
        filter2.js
    services/
        service1.js
        service2.js
lib/
    bootstrap/
        css/
            bootstrap.css
            bootstrap.min.css
        fonts/
            glyphicons-halflings-regular.eot
            glyphicons-halflings-regular.svg
            glyphicons-halflings-regular.ttf
            glyphicons-halflings-regular.woff
        js/
            bootstrap.js
            bootstrap.min.js
        angularjs/
            js/
                angular.js
                angular-route.js
                angular-locale_es-es.js
seguromedico/
    seguromedico-list.html
    seguromedico-list.js
    seguromedico-detail.html
    seguromedico-detail.js
    seguromedico.js
usuario/
    usuario-list.html
    usuario-list.js
    usuario-detail.html
    usuario-detail.js
```

```
└── usuario.js
```

app.js En aquest fitxer és on es defineix el mòdul de la funció, les constants, els blocs `config` i els blocs `run`.

index.html Pàgina principal que contindrà la resta de pàgines.

index.js És el `javascript` específic de `index.html`.

main.html És el `HTML` de la ruta per defecte que s'ha definit en `app.js`.

main.js És el `javascript` específic de `main.html`.

common Directori que conté les directives, els filtres, serveis, etc que són comuns a tota l'aplicació.

lib Directori on es guarda una de les llibreries de tercers que es farà servir.

seguromedico Directori que conté la funcionalitat per fer el que sigui necessari sobre l'entitat `SeguroMedico`.

usuario Directori que conté la funcionalitat per fer el que sigui necessari sobre l'entitat `Usuario`.

1.1.2.3. Pylons

Pylons és un *framework* d'aplicacions web de codi obert, escrit en `Python`, que segueix l'arquitectura `MVC` i fa servir objectes de petició i resposta `HttpRequest` [Wik16t], les dues explicades com funcionen en l'apartat 1.1.2.1.
Les característiques principals de Pylons són les següents [Sli16]:

- Té com objectiu fer un desenvolupament web ràpid, flexible i senzill.
- L'estil de l'aplicació web es la piràmide.
- Permet simplificar certs processos de desenvolupament.
- Ofereix un esquelet base per començar a construir qualsevol aplicació web.

L'estructura d'arxius que té un projecte en Pylons és la següent:

```
pyramid_blogr/
├── __init__.py
├── models.py
├── scripts/
├── static/
├── templates/
├── tests.py
└── views.py
```

pyramid_blogr Directori on es guarda tot el projecte Pylons.

__init__.py Fitxer principal que configura i retorna l'aplicació `WSGI`.

models.py Fitxer on es defineixen les taules de la base de dades.

scripts Directori on es guarden els scripts de Python.

static Directori on es guarden els fitxers estàtics, com els `css, js o imatges`.

templates Directori on es guarden els `templates`.

tests.py Fitxer per fer proves a les `views`.

views.py Fitxer on es fa la feina lògica.

1.1.2.4. Laravel

Pylons és un *framework* d'aplicacions web de codi obert i `PHP`, escrit en `PHP`, que segueix l'arquitectura `MVC` i fa servir objectes de petició i resposta `HttpRequest` [Wik16m], les dues explicades com funcionen en l'apartat 1.1.2.1.
Les característiques principals de Laravel són les següents [Pla16]:

- És modular i extensible, el que significa que li puc afegir tots els paquets que necessiti per l'aplicació que vull desenvolupar.
- Permet fer un desenvolupament ràpid i senzill de micro-serveis i té una `API` de gran rendiment per als projectes.
- Té un sistema d'enrutament ràpid i eficient.
- Compta amb un `Middleware` que analitza i filtra les peticions `HTTP` en el servidor.
- Té un sistema d'autentificació d'usuaris natiu.
- Pot encriptar les seves pròpies dades per tenir més seguretat.
- Pot executar processos llargs i complexos en segon pla.

L'estructura d'arxius que té un projecte en Laravel és la següent [Git16]:

```
app/
bootstrap/
config/
database/
public/
resources/
storage/
tests/
vendor/
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
gulpfile.js
```

```
├── package.json
├── phpspec.yml
├── phpunit.xml
├── readme.md
└── server.php
```

app Directori per guardar el codi personal del projecte.

bootstrap Directori on hi ha el sistema d'arrencada.

config Directori per configurar tant el *framework* com per l'aplicació que estigui dintre del directori.

database Directori on es troben tots els fitxers relacionats amb la manipulació de la base de dades.

public Directori on es col·loquen tots els recursos estàtics de l'aplicació.

resources Directori on hi ha els fitxers per desenvolupar el `frontend`.

storage Directori que es fa servir quan es vol escriure alguna dada en el disc.

tests Directori on hi ha els fitxers de prova.

vendor Directori on es guarden les llibreries externes.

.env Fitxer de les variables d'entorn.

.env.example Fitxer de les variables d'entorn.

.gitattributes Fitxer per tenir en el projecte atributs del `Git`.

.gitignore Fitxer per obviar configuracions del `Git`.

artisan Fitxer executable que es fa servir per executar la interfície de comandes `Artisan` per Laravel. Conté comandes per obtenir dreceres o funcionalitats addicionals al *framework*.

composer.json Conté informació per el `Composer`.

composer.lock Conté informació per el `Composer`.

gulpfile.js Fitxer per crear una compilació versionada dels arxius, per evitar que es carreguin sessions anteriors del projecte que hi hagi guardades en la `memòria cau` del navegador de l'usuari.

package.json Fitxer que conté les comandes del `Composer`.

phpspec.yml Fitxer per interactuar amb el `Composer`.

phpunit.xml Fitxer que conté per defecte les proves unitàries de `PHPUnit`.

readme.md Fitxer on hi ha una breu explicació de com començar a treballar amb el Laravel.

server.php Fitxer que crea un servidor intern per desenvolupar el projecte.

1.1.2.5. Comparació i Tria del Framework

Un cop vaig investigar aquests quatre *frameworks*, el que vaig fer va ser comparar les seves prestacions, per decidir amb quin d'ells treballaria per desenvolupar aquest projecte. Aquesta comparació la vaig fer amb els paràmetres que es poden observar en la taula següent [vsC16c] [vsC16d]:

Nom	Django	AngularJS	Pylons	Laravel
Llenguatge	Python	JavaScript	Python	PHP
Llicència	Lliure	Lliure	Lliure	Lliure
Patró de Disseny	MVC	MVC	MVC	MVC
Multi-llenguatge	Si	Si	Si	Si
Sistemes Operatius	Multi-plataforma	Multi-plataforma	Multi-plataforma	Multi-plataforma
Models de Bases de Dades	Relacionals, NoSQL, Document-oriented	JSON Database, NoSQL	Relacionals, NoSQL, Schema-less, Key-value, Dynamic Schema, Object-oriented, Object-relational	Object-oriented
Bases de Dades	IBM DB2, Cassandra, Couchbase, SQLite, PostgreSQL, MySQL, Oracle, MongoDB, Redis, Microsoft BI	MongoDB, MySQL, SQLite	MySQL, MongoDB, Microsoft BI, PostgreSQL, SQLite, MariaDB, Redis, CouchDB, Cassandra Oracle	SQLite, MySQL, PostgreSQL, Redis, Microsoft BI, MongoDB
Sistema Multi-usuari	Si	Si	Si	Si
Plug-in	Si	Si	Si	Si
Intèrpret	Si	No	Si	Si
Transaccions	Si	No	Si	Si
Unicode	Si	Si	Si	Si
Projectes Múltiples	Si	Si	Si	Si
Compilació Estàndard	Si	No	Si	Si
Escalable Horitzontalment	Si	No	Si	Si

Nom	Django	AngularJS	Pylons	Laravel
Nivell de Dificultat	Mitjà, avançat	Avançat	Mitjà, avançat	Principiants, mitjà, avançat i mestre
Nivell de Documentació	Alt	Mitjà	Alt	Alt
Generació de Codi	Si	Si, però amb condicions	Si, però amb condicions	Si
Menús Jeràrquics	Si,	Si, però amb condicions	Si, però amb condicions	Si
Llenguatges de suport per Scripts	Python, CoffeeScript, JavaScript	Dart, CoffeeScript, TypeScript, JavaScript	Python	PHP
Backend	Python	No en té	Python	PHP
Generació d'Administrador	Si	No	No	No
Navegadors que el suporten	Tots	Internet Explorer, Firefox, Chrome, Opera, Safari	Chrome, Firefox, Internet Explorer, Opera, Safari, Lynx, Maxthon	Tots
Seguretat de les dades	Si	No	Si	Si
Previsió contra Malware	Si	Si	No	Si

Taula 1.1.: Comparació de frameworks

Després d'observar aquesta taula, amb les prestacions que m'oferia cada *framework*, vaig decidir escollir el *framework* `Django`. El que hem va fer decantar per Django van ser les raons següents:

- El llenguatge amb el que es programa és `Python`, que ja el conec i estic familiaritzat amb ell. A més, és un llenguatge d'alt nivell.
- Tenia una major quantitat de bases de dades per escollir que els altres *frameworks*.
- El projecte és escalable horitzontalment, per tant el projecte el podrà fer més gran més endavant, si es vol que faci alguna altre funció en el futur.
- El nivell de dificultat és mitjà.
- El nivell de documentació es alt.
- Et proporciona un sistema d'administració. Cap dels altres `frameworks` té aquesta opció.

- El suporten tots els navegadors.
- A més a més, llocs com Instagram, New York Times, la NASA i el Washington Post entre d'altres fan servir Django per el seu lloc web

Per tant, tenint en compte aquestes raons, `Django` és la millor opció per desenvolupar aquest projecte.

1.1.3. Base de Dades

Les bases de dades són bancs de informació que contenen dades relatives a diverses temàtiques i categories de diferents maneres, però que entre si comparteixen algun tipus de vincle o una relació que busca ordenar-los i classificar-los en conjunt. Les dades estan emmagatzemades sistemàticament per el seu ús posterior [Wik16b].
Les bases de dades les podem classificar per el model d'administració de dades que fan servir. Un model de dades és una descripció d'alguna cosa coneguda com contenidor de dades, així com dels mètodes per emmagatzemar i recuperar informació d'aquests contenidors.
Els models utilitzats en les bases de dades són:

- Bases de dades jeràrquiques
- Bases de dades de xarxa
- Bases de dades transaccionals
- Bases de dades relacionals
- Bases de dades no relacionals
- Bases de dades multi-dimensionals
- Bases de dades orientades a objectes
- Bases de dades documentals
- Bases de dades deductives

Després d'estudiar els diferents models existents, m'he centrat en dos en concret, ja que són els que millor s'adapten al projecte, i els que més s'utilitzen. Aquests dos models són el **model relacional** i el **model no relacional**.

1.1.3.1. Model Relacional

El model relacional permet establir interconnexions o relacions entre les dades que estan guardades en taules, i fent servir les connexions, establir relacions de les dades de les diverses taules [Wik16c]. Les característiques del model relacional són:

- La base de dades es composa de vàries taules o relacions.
- No poden existir dues taules amb el mateix nom ni registre.
- Cada taula és a la vegada un conjunt de files i columnes.

- La relació entre taules és duu a terme per mitjà de claus primàries i claus foranes.
- La clau primària ha de complir la integritat de dades.
- La clau forana es col·loca en una taula secundària, i aquesta conté el valor de la clau primària de la taula primària

Estructura L'estructura del model relacional està organitzada en dues seccions diferenciades: l'esquema i les dades.
L'`esquema` és la definició de l'estructura de la base de dades, en la que es guarden les següents dades principalment:

- El nom de cada taula.
- El nom de cada columna.
- El tipus de dades de cada columna.
- La taula a la que pertany cada columna.

Les bases de dades relacionals passen per un procés de normalització de la base de dades, el resultat del qual és un esquema que permet que la base de dades es pugui fer servir de manera òptima.
Les `dades` és el contingut de la base de dades en un moment concret, és el contingut de tots els registres.

Avantatges i Desavantatges Les avantatges que presenta una base de dades relacional són les següents:

1) Proveeix d'eines que garanteixen evitar la duplicitat de registres.
2) Garanteix la integritat referencial, i així, al eliminar un registre, elimina tots els registres relacionats dependents.
3) Afavoreix la normalització per ser més comprensible i aplicable.

D'altra banda, les desavantatges que presenta una base de dades relacional són les següents:

1) Presenten deficiències amb dades gràfiques, multimèdia, disseny assistit de computadores (CAD) i sistemes d'informació geogràfica.
2) No es manipulen de forma manejable els blocs de text com tipus de dades.

1.1.3.2. Sistema de Gestió de Bases de Dades (DBMS)

Per poder treballar amb les bases de dades relacionals, existeix un software anomenat Sistema de Gestió de Bases de Dades (DBMS). Els `DBMS` més importants són l'Oracle, MySQL, PostGreSQL, i SQLite.

Oracle És considerat com un dels `DBMS` més complert, destacant per el seu suport de transaccions, la seva estabilitat, la seva escalabilitat i el seu suport Multi-plataforma [Wik16r].

MySQL Destaca per aprofitar la potència dels sistemes multiprocessador, gràcies a la seva implementació multi fil. Suporta una gran quantitat de llenguatges, disposa de `API's` amb una gran quantitat de llenguatges. Gran portabilitat entre sistemes, pot treballar en diferents `sistemes operatius`. Suporta fins a 32 índexs per taules. Gestió d'usuaris i contrasenyes, mantenint un nivell molt bo de seguretat en les dades [Wik16o].

PostGreSQL Destaca per les seves `vistes`, que són el resultat d'una consulta `SQL` d'una o vàries taules, també es pot considerar una taula virtual. Per la seva integritat transaccional. Per l'herència de taules. Pel tipus de dades i operacions geomètriques. Suport per transaccions distribuïdes, permet a PostGreSQL integrar-se en un sistema distribuït format per varis recursos [Wik16s].

SQLite Destaca perque implementa la major part del estàndard `SQL-92`. Inclou transaccions de base de dades atòmiques. Consistència de la base de dades, aïllament i durabilitat (`ACID`). Triggers. Inclou la major part de les consultes complexes [Wik16x].

1.1.3.3. Comparació dels DBMS

Un cop vaig investigar els `DBMS` més destacats, vaig fer la següent taula per comparar les seves prestacions [vsC16b] [dbE16]:

Nom	Oracle	MySQL	PostGreSQL	SQLite
Interfície	GUI, SQL	GUI, SQL	GUI, SQL	GUI, SQL
Llenguatges Suportats	C, C#, C++, Clojure, Cobol, Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp, Objective C, OCaml, Perl, PHP, Python, R, Ruby, Scala, Tcl i Visual Basic	Ada, C, C#, C++, D, Eiffel, Erlang, Haskell, Java, Objective C, OCaml, Perl, PHP, Python, Ruby, Scheme, Tcl	.NET, C, C++, Java, Perl, Python i Tcl	Actionscript, Ada, Basic, C, C#, C++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Objective-C, OCaml, Perl, PHP, PL/SQL, Python, R, Ruby, Scala, Scheme, Smalltalk, Tcl
Llenguatge d'Implementació	C i C++	C i C++	C	C

Nom	Oracle	MySQL	PostGreSQL	SQLite
Sistemes Operatius	Windows, Linux, Solaris, HP-UX, OS X, z/OS i AIX	Windows, Linux, OS X, FreeBSD i Solaris	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix i Windows	Windows, Android, Linux, OS X, FreeBSD i Solaris
Llicència	Privada	Lliure	Lliure	Lliure
Esquema de Dades	Si	Si	Si	Si
Tipus de Dades	Si	Si	Si	Si
Índex Secundaris	Si	Si	Si	Si
Triggers	Si	Si	Si	Si
Mètodes de Partició	Horitzontal	Horitzontal	No	No
Claus Foranes	Si	Si	Si	Si
Concepte de Transacció	ACID	ACID	ACID	ACID
Concurrència	Si	Si	Si	Si

Taula 1.2.: Comparació de DBMS

Observant la taula anterior, vaig determinar que el `DBMS` que millor s'adaptarà al projecte és el `SQLite`, ja que és un punt mig entre l'Oracle i el PostGreSQL, és programari lliure i té quasi totes les característiques que té l'Oracle que és privatiu. I comparat amb el MySQL, és gairebé igual, l'única característica diferent és la de la partició, que el MySQL la té i SQLite no. Però com que aquesta característica no la faré servir, no afecta a l'hora de prendre la decisió.
També m'he decidit a treballar amb SQLite perque ja la vaig estudiar en una assignatura de la universitat, el que fa que també estigui familiaritzat a treballar amb aquesta base de dades. A més a més, dóna solucions a moltes limitacions de les que presenta `MySQL`.

1.1.3.4. Model No Relacional

El `model no relacional` (NoSQL) és un sistema de gestió de bases de dades que és diferencia del model relacional en aspectes importants [Wik16q]:

- No fan servir el `SQL` com a principal llenguatge de consultes.
- Les dades emmagatzemades no requereixen estructures fixes com taules.
- No suporten operacions `JOIN`, que és fer una consulta en dues taules a simultàniament.
- No garanteixen completament l'`ACID` (atomicitat, consistència, aïllament i durabilitat).
- Escalen bé horitzontalment.

Les bases de dades NoSQL es classifiquen segons la seva forma d'emmagatzemar les dades, i comprenen categories com *clau-valor*, les implementacions de BigTable, bases de dades documentals i bases de dades orientades a grafs.
Les NoSQL estan altament optimitzades per les operacions de recuperar i afegir, i normalment no ofereixen molt més que la funcionalitat d'emmagatzemar els registres. La pèrdua de flexibilitat en el temps d'execució, comparat amb els sistemes clàssics de SQL, és veu compensat pels guanys significatius en la escalabilitat i el rendiment quan es tracta de certs models de dades. La seva característica principal és que estan pensades per manipular quantitats enormes de informació de manera molt ràpida.

Arquitectura Les arquitectures NoSQL moltes vegades aporten escasses garanties de consistència. No obstant això, alguns sistemes aporten totes les garanties dels sistemes ACID si li afegim una capa intermitja. També hi ha dos sistemes que aportes aïllament per l'emmagatzematge de les columnes. Aquests sistemes fan servir conceptes similars per aconseguir transaccions ACID distribuïdes de múltiples files amb garanties d'aïllament per el sistema subjacent d'emmagatzemament en aquesta columna, sense sobrecarregar més la gestió de les dades.
Molts sistemes NoSQL fan servir una arquitectura distribuïda, mantenint les dades de forma redundant en varis servidors, fent servir molt sovint una taula hash distribuïda. D'aquesta manera, el sistema pot realment escalar afegint més servidors, i la fallada en un servidor es pot tolerar.

Avantatges i Desavantatges Les avantatges que presenta una base de dades NoSQL són les següents:

- Responen a les necessitats d'escalabilitat horitzontal que tenen cada cop més empreses.
- Poden manipular gran quantitat de dades.
- No generen *colls d'ampolla*.
- Escalament senzill.
- Diferents bases de dades NoSQL per diferents projectes.
- S'executen en un grup de màquines barates.

D'altra banda, les desavantatges que presenta una base de dades NoSQL són les següents:

- No estan suficientment desenvolupades per algunes empreses.
- Possibles problemes d'inestabilitat.
- Limitació en la intel·ligència de les consultes, ja que una simple consulta implica tenir coneixements de programació bastant bons.
- Al ser relativament noves, no hi ha una gran quantitat de desenvolupadors i administradors que coneguin aquesta tecnologia.
- Problemes de compatibilitat, ja que no comparteixen gaires normes. Cadascuna té la seva pròpia API, interfícies de consulta i les seves peculiaritats.

Sistemes Per poder treballar amb les bases de dades `NoSQL`, existeixen diferents sistemes de bases de dades, els quals són:

- Bases de dades documentals
- Bases de dades en graf
- Bases de dades clau/valor
- Bases de dades multivaluar
- Bases de dades orientades a objectes
- Bases de dades tabulars
- Bases de dades de arrays

Després d'estudiar els diferents sistemes existents de bases de dades, m'he centrat en estudiar més a fons el sistema de `base de dades clau/valor` i el de `base de dades documentals`, ja que són les dues que millor s'adapten aquest projecte.

Base de Dades Clau/Valor Les `bases de dades clau/valor` és un paradigma d'emmagatzemament de dades dissenyat per emmagatzemar, recuperar i administrar matrius associatives, una estructura de dades més coneguda com a diccionari o hash [Wik16l]. Els diccionaris contenen una col·lecció d'objectes, o registres, que a la vegada tenen molts camps diferents dintre d'ells, cada un de les dades que conté. Aquests registres s'emmagatzemen i es recuperen mitjançant una clau que identifica el registre, i s'utilitza per trobar ràpidament les dades dintre de la base de dades. Aquesta clau com a màxim apareix un cop en tota la col·lecció de dades. Les bases de dades més importants d'aquest tipus són `MongoDB` i `Redis`.

MongoDB És la base de dades `NoSQL` més utilitzada. Les seves característiques són [Wik16n]:

1) De propòsit general, i amb casi totes les funcionalitats de les bases de dades relacionals.
2) Alta disponibilitat.
3) Flexible.
4) Escalabilitat, des de un servidor aïllat a arquitectures distribuïdes de grans `clusters` (grup).
5) Aggregation Framework. Processament batch (execució d'una sèrie de programes sense interacció humana en un computador) de dades per càlculs agrupats fent servir operacions natives de MongoDB.
6) Auto balancejat de la càrrega fent servir diferents `shards` (fragments).
7) Replicació nativa. Sincronització de dades entre servidors.
8) Seguretat. Autentificació, autorització, etc.
9) Gestió avançada dels usuaris.
10) Automàtic `failover`. Elecció automàtica d'un nou primari si aquest cau.
11) S'actualitza sense deixar de donar servei.

12) No té *colls d'ampolla* al processar grans quantitats de informació.
13) Utilitza objectes `JSON` per guardar i transmetre la informació.
14) Permet realitzar consultes i càlculs espaials, tant en 2D com en 3D.
15) Permet utilitzar `Map-Reduce` per el processat de la informació fent servir funcions `JavaScript` que s'executen en els servidors.
16) Pot emmagatzemar i executar funcions `JavaScript` en el servidor.
17) Té una eina web potent que permet monitoritzar tot el que passa en les nostres bases de dades i les nostres màquines.
18) `Backup` continu, i múltiples còpies en diferents data centers.
19) Automatització de tasques.

Redis És una de les bases de dades `NoSQL` més destacades per tenir les següents característiques [Wik16u]:

1) Totes les dades són en la memòria.
2) Rapidesa.
3) Permet que les dades persisteixin.
4) Mono-fil.
5) Operacions atòmiques.
6) Estructura de dades avançada en hashs, llistes i sets.
7) Suporta tipus de dades i transaccions.
8) Escalabilitat.
9) Rendiment previsible, només hi ha operacions eficients, optimitzades per defecte.
10) Particionat automàtic, les dades són en diferents nodes.
11) Modelat de les dades.

Base de Dades Documentals El concepte central de les `bases de dades documentals` és la noció d'un document [Wik16f]. Mentre que cada implementació de base documental difereix en els detalls d'aquesta definició, en general, tothom assumeix que els documents encapsulats i les dades codificades (o informació) en alguns formats o codificacions estàndards. La codificació en ús incloent `XML, YAML i JSON`, així com les formes binàries com `BSON`. Els documents s'aborden en la base de dades fent servir una clau única que representa aquest document. Una altre de les característiques que defineixen una base de dades documental és que, a més de la busca de claus realitzada per un magatzem de `claus/valor`, la base de dades ofereix una `API` que recupera els documents en funció del seu contingut.
Diferents implementacions que ofereixen diferents formes d'organitzar o agrupar documents són:

- Col·leccions
- Etiquetes
- Metadades no visibles

- Jerarquia de Directoris

En comparació amb les bases de dades relacionals, les col·leccions podrien considerar-se anàlogues a les taules i els documents anàlegs als registres. Però són coses diferents, cada registre d'una taula té la mateixa seqüència de camps, mentre que els documents en una col·lecció poden tenir camps completament diferents. La base de dades més important d'aquest tipus és la `CouchDB`.

CouchDB És una de les bases de dades `NoSQL` més destacades del mercat [Wik16d]. Es caracteritza per:

1) Emmagatzemament de documents. Això és, un o més parells de camp/valor expressats en `JSON`. Els valors dels camps poden ser dades simples (caràcters, números, dates), però també poden ser llistes ordenades i vectors associatius.
2) Semàntica `ACID`.
3) Vistes i índex `Map-Reduce`. Les dades emmagatzemades s'estructuren per mitjà de vistes.
4) Arquitectura distribuïda amb replicació. Cada replica pot tenir una còpia de les mateixes dades. La replica és horitzontalment.
5) Interfície `REST`. Tots els ítems tenen una `URL` única que queda exposada via `HTTP`.
6) Consistència eventual. Ofereix tant disponibilitat com tolerància a les particions.
7) Feta per operar off-line.
8) Altament concurrent.

1.1.3.5. Comparació de Bases de Dades NoSQL

Un cop vaig investigar les bases de dades `NoSQL` més destacades, vaig fer la següent taula per comparar les seves prestacions [vsC16a]:

Nom	MongoDB	Redis	CouchDB
Interfície	GUI, HTTP	GUI, HTTP	GUI, HTTP REST
Llenguatges Suportats	Actionscript, C, C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, Javascript, Lisp, Lua, MatLab,Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk	C, C#, C++, Clojure, Crystal, D, Dart, Elixir, Erlang, Fancy, Go, Haskell, Haxe, Java, JavaScript, Lisp, Lua, MatLab, Objective C, OCaml, Perl, PHP, Prolog, Pure Data, Python, R, Rebol, Ruby, Rust, Scala, Scheme, Smalltalk, Tcl	C, C#, ColdFusion, Erlang, Haskell, Java, Javascript, Lisp, Lua, Objective C, OCaml, Perl, PHP, PL/SQL, Python, Ruby, Smalltalk
Llenguatge d'Implementació	C++	C	Erlang
Sistemes Operatius	Linux, OS X, Solaris i Windows	BSD, Linux, OS X i Windows	Android, BSD, Linux, OS X, Solaris i Windows
Llicència	Lliure	Lliure	Lliure
Esquema de Dades	Esquema lliure	Esquema lliure	Esquema lliure
Tipus de Dades	Si	Parcial	No
Índex Secundaris	Si	No	Si
Triggers	No	No	Si
Mètodes de Partició	Sharding	Sharding	Sharding
Claus Foranes	No	No	No

Nom	MongoDB	Redis	CouchDB
Concepte de Transacció	No	Bloqueig optimista, execucions atòmiques de blocs de comandes i scripts	No
Concurrència	Si	Si	Si

Taula 1.3.: Comparació de bases de dades NoSQL

Observant la taula anterior, vaig determinar que el sistema `NoSQL` que millor s'adaptarà al projecte és el `MongoDB`, ja que és de llicència lliure, té tipus de dades i índexs secundaris, cosa que els altres dos no tenen a la vegada. Per tant, és el sistema `NoSQL` més complert, i també el que millor s'adaptaria al projecte.

1.1.3.6. Elecció Base de Dades Relacional o Base de Dades NoSQL

Després d'haver estudiat el model relacional i el model NoSQL, m'he decantat per fer servir el `model relacional`, per les raons que exposo a continuació:

1) Les he estudiat a classe i, per tant, sé com treballar amb elles.

2) Són lo suficientment potents com per optimitzar `joins` complexos.

3) Se li poden realitzar milers de consultes amb milers d'usuaris i segueixen funcionant perfectament.

4) Estan dissenyades per executar-se en `clusters` de manera eficient.

5) Són generalment de programari lliure.

6) Incorporen el concepte de `clau forana`, que em serà molt útil per determinar les relacions entre les diferents taules del projecte.

Per tant, tenint totes aquestes raons en compte, per desenvolupar el projecte faré servir la base de dades de model relacional, concretament `SQLite`.

1.2. Frontend

El `frontend` és el software amb el que interactua l'usuari a través de la web. Per desenvolupar el `frontend`, un cop vaig escollir que faria servir el *framwework* Django, vaig investigar amb quin llenguatge s'escrivien els templates. Un cop vaig investigar-ho, vaig veure que utilitzava el llenguatge més estàndard, el `HTML`.
Un cop vaig descobrir això, vaig investigar una mica més el `HTML`, i els complements que podia fer servir amb ell, que eren el `CSS`, `JavaScript`, `jQuery` i `Bootstrap`.

1.2.1. HTML

HTML és un acrònim de *HyperText Markup Language* i fa referència al llenguatge de marxat per a la elaboració de pàgines web [Wik16i]. És un estàndard que serveix de referència per la elaboració de pàgines web i serveix una estructura bàsica i un codi per a la definició de contingut d'una pàgina web; com texts, imatges, vídeos, entre d'altres.
Aquest llenguatge basa la seva filosofia de desenvolupament en la diferenciació; per afegir un element extern a la pàgina (imatge, vídeo, *script*, ...) aquest no s'incrusta directament en el codi de la pàgina, sinó que es fa una referència a la ubicació d'aquest element mitjançant text. D'aquesta manera, la pàgina web conté només text mentre que recau en el navegador web (l'intèrpret del codi) la tasca d'unir tots els elements i visualitzar la pàgina final. Al ser un estàndard, `HTML` busca ser el llenguatge que permeti a qualsevol pàgina web escrita en una determinada versió, pugui ser interpretada de la mateixa forma per qualsevol navegador actualitzat.

1.2.2. CSS

CSS és un acrònim de *Cascading Style Sheets* (fulla d'estil en cascada) i és un llenguatge utilitzat per definir i crear la presentació d'un document estructurat escrit en HTML o XML [Wik16h]. La informació d'estil pot ser definida en un document separat o en el mateix document HTML. En l'últim cas, podrien definir-se estils generals amb l'element `<style>` o en cada etiqueta particular mitjançant l'atribut `style="...`. Algunes de les seves característiques són les següents:

- Proporciona un control centralitzar de la presentació d'una aplicació web complet amb el que s'agilitza de forma considerable l'actualització del mateix.
- Optimitza l'ample de banda de la connexió, ja que poden definir-se els mateixos estils per molts elements amb només un selector i perquè un mateix arxiu CSS pot servir per una multitud de documents.

1.2.3. JavaScript

JavaScript és un llenguatge de programació interpretat. Es defineix com un llenguatge orientat a objectes, basat en prototips, imperatiu i dinàmic [Wik16j].
S'utilitza principalment en la seva forma del costat del client, implementat com part d'un navegador web permeten millores en la interfície d'usuari i pàgines web dinàmiques, tot i que existeix un JavaScript del costat del servidor.
Té una sintaxi similar a C, tot i que adopta noms i convencions del llenguatge Java. Tot i això, Java i JavaScript no estan relacionats i tenen propòsits diferents.
Per poder interactuar amb una pàgina web, JavaScript té una implementació del Document Object Model (DOM). Les avantatges que aporta són les següents:

1) És imperatiu i estructurat
2) Dinàmic
3) Funcional
4) Prototípic

5) Té un entorn de execució

6) Funcions amb mètodes

7) Taules i definició literal d'objectes

8) Expressions regulars

1.2.4. jQuery

jQuery és una biblioteca de software lliure i de codi obert de JavaScript que permet simplificar la manera d'interactuar amb els documents HTML, permetent manipular l'arbre DOM, manipular events, desenvolupar animacions i afegir interaccions a pàgines web [Wik16k].
Les avantatges que aporta són les següents:

1) És flexible i ràpid pels desenvolupadors web

2) Ve amb llicència MIT i es Open Source

3) Té una excel·lent comunitat de suport

4) Té plug-ins

5) Els bugs se solucionen ràpidament

1.2.5. Bootstrap

Bootstrap és un *framework* CSS o conjunt d'instruments de codi obert per dissenyar lloc i aplicacions web [Wik16y]. Conté plantilles de disseny amb tipografia, formularis, botons, quadres menús de navegació i altres elements de disseny basats en HTML i CSS, i també té extensions de JavaScript opcionals addicionals. Bootstrap és caracteritza per:

- Compatible per la majoria de navegadors webs.
- El disseny gràfic de la pàgina s'ajusta dinàmicament.
- És de codi obert.
- Bootstrap ofereix les eines necessàries per crear qualsevol lloc web.
- Sistema GRID que permet dissenyar fent servir un GRID de dotze columnes.

1.3. Servidor

1.3.1. Què és un Servidor?

Un servidor és una aplicació de software capaç d'atendre les peticions d'un client i tornar-li una resposta en concordança [Wik16v]. Els servidors es poden executar en qualsevol tipus de computadora. Existeixen computadores que es dediquen a fer exclusivament de servidor. En la majoria de casos una mateixa computadora pot proveir múltiples serveis i tenir diferents servidors en funcionament.
Els servidors operen a través de l'arquitectura `client-servidor`. Els servidors són programes de computadora en execució que atenen les peticions d'altres programes, els clients. Per tant, el servidor realitza altres tasques per benefici dels clients. Ofereix al client les possibilitats de:

- Compartir dades.
- Compartir informació.
- Compartir recursos de `hardware` i `software`.

Els clients usualment es connecten al servidor a través de la xarxa, però també poden accedir a ell a través de la computadora on està funcionant. En el context de xarxes Internet Protocol (IP), un servidor és un programa que opera amb `sockets`. Un socket és un concepte abstracte amb el qual dos programes (normalment en computadores diferents) poden intercanviar qualsevol flux de dades, generalment de manera fiable i ordenada.

1.3.1.1. Tipus de Servidors

Els servidors proveeixen serveis essencials dintre d'una xarxa, ja sigui per usuaris privats dintre d'una organització o companyia, o usuaris públics a través d'Internet. Els tipus de servidors més comuns són:

Servidor de Base de Dades Proveeix serveis de base de dades a altres programes o altres computadores, com està definit en el model `client-servidor`. També pot fer referència a aquelles computadores servidores dedicades a executar programes, prestant atenció al servei.

Servidor d'Arxius És el servidor que emmagatzema varis tipus d'arxius i els distribueix a altres clients en la xarxa.

Servidor de Correu Servidor que emmagatzema, envia, rep, enruta i realitza altres operacions relacionades amb el correu electrònic per els clients de la xarxa.

Servidor d'Impressió Servidor que controla una o més impressores i accepta treballs d'impressió d'altres clients de la xarxa, posant en cua altres treballs d'impressió i realitzant la majoria o totes les altres funcions que un lloc de treball es realitzarien per aconseguir una tasca d'impressió si la impressora estigués connectada directament amb el port d'impressora del lloc de treball.

Servidor Web Servidor que emmagatzema documents `HTML`, imatges, arxius de texts, escriptures, a més de material Web composats per dades, i distribueix aquest contingut a clients que el demanen en la xarxa.

Servidor de Joc Servidor que utilitzen els clients de videojocs per jugar a videojocs multijugador.

Servidor d'Aplicacions Servidor que proporciona serveis d'aplicació a les computadores client.

Jo pel projecte que vull dur a terme, faré servir el **Servidor Web**.

1.3.2. Funcionament d'un Servidor Web

El funcionament d'una petició HTTP a un servidor és la següent [K W12]:

1) El client HTTP inicia la connexió **Protocol de Control de Transmissió** (`TCP`) fent una petició a la `URL` del servidor HTTP.

2) El servidor HTTP rep la petició de la `URL TCP` i accepta la connexió, notificant-ho al client.

3) El client HTTP envia un missatge (que conté la `URL`) dintre la connexió `TCP`.

4) El servidor HTTP rep el missatge, prepara la resposta i l'envia al client.

5) El servidor HTTP tanca la connexió `TCP`

6) El client HTTP rep la resposta en el fitxer `HTML`, que veu en el seu navegador.

7) Es tornen a repetir els passos anteriors per un altre petició.

L'esquema del procés descrit abans és el següent:

Figura 1.4.: Esquema funcionament petició HTTP

1.3.3. Elecció del Servidor

Després d'haver investigat que és un servidor web i com funciona, vaig haver d'investigar quins tipus de programes podrien funcionar per poder emmagatzemar el projecte dintre seu i que el serveixi correctament.

1.3.3.1. Apache

Apache és un servidor web de codi lliure, totalment configurable. Destaca per ser multi-plataforma, molt robust i per la seva seguretat i rendiment. S'utilitza principalment per realitzar el servei de pàgines web, ja siguin estàtiques o dinàmiques [Cul16]. Entre les principals característiques d'Apache, les més importants són:

- Suport de seguretat `SSL` i `TLS`.

- Pot realitzar autentificació de dades utilitzant `SGDB`.
- Dóna suport a diferents llenguatges com Perl, PHP, Python i tcl.
- Virtual Hosting. Pot allotjar varis dominis en una mateixa IP.

L'esquema de com estan estructurats els arxius en l'Apache és el següent:

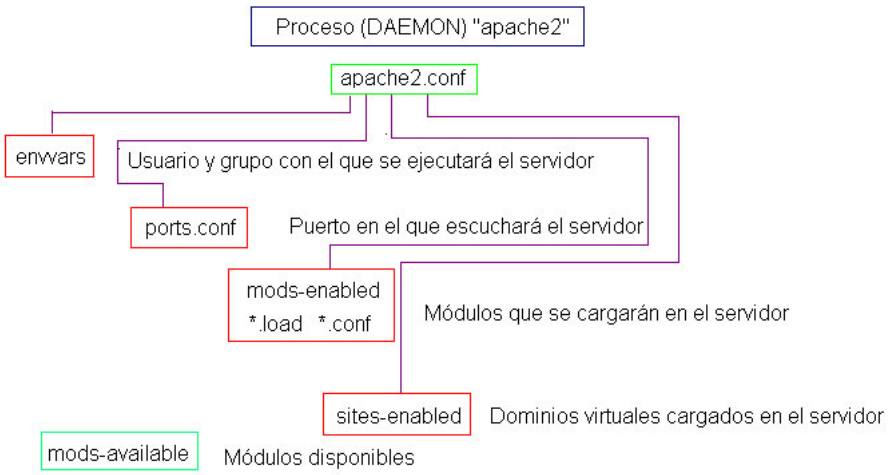

Figura 1.5.: Esquema de fitxers d'Apache

L'arquitectura d'Apache és modular. Hi ha diversos mòduls que aporten moltes funcionalitats. Els mòduls més destacats són els següents [Wik16w]:

mod_ssl Comunicacions segures via **Seguretat de la capa de Transport** `(TLS)`.

mod_rewrite Reescriptura de direccions.

mod_dav Suport del protocol `WebDAV` (Creació i Control de versions distribuïdes en la web).

mod_deflate Compressió transparent amb l'algoritme `deflate` del contingut enviat al client.

mod_auth_ldap Permet autentificar els usuaris contra un servidor `LDAP`.

mod_proxy_ajp Connector per enllaçar amb el servidor amb el servidor `Jakarta Tomcat` de pàgines dinàmiques en `Java`.

mod_cfml Connector `CFML` fet servir per `Railo`.

El servidor de base pot ser estès amb la inclusió de mòduls externs entre els que hi ha:

mod_cband Control de tràfic i limitador d'ample de banda.

mod_perl Pàgines dinàmiques en `Perl`.

mod_php Pàgines dinàmiques en `PHP`.

mod_python Pàgines dinàmiques en `Python`.

mod_rexx Pàgines dinàmiques en REXX i Object REXX.

mod_ruby Pàgines dinàmiques en Ruby.

mod_mono Pàgines dinàmiques en Mono.

mod_security Filtrat a nivell d'aplicació, per seguretat

1.3.3.2. Lighttpd

Lighttpd és un servidor web dissenyat per ser ràpid, segur, flexible i segueix els estàndards. Està optimitzat per entorns on la velocitat és molt important. Pot tenir aquesta velocitat ja que consumeix menys CPU i memòria RAM que altres servidors [Lin16].
Lighttpd és apropiat per qualsevol servidor que tingui problemes de càrrega. És de software lliure. Funciona en GNU/Linux i UNIX.
Les característiques principals són:

- Virtual Hosting.
- CGI, SCGI, FastCGI. Amb aquests programes, pot donar suport a casi qualsevol llenguatge de programació.
- Suport per PHP, Ruby, Python i altres.
- Consum de memòria constant.
- Redireccions HTTP, i reescriptures de URL.
- Xifrat SSL.

L'arquitectura de Lighttpd és modular. Hi ha diversos mòduls que aporten moltes funcionalitats. Els mòduls més destacats són els següents [LIG16]:

SSL Configuració de les comunicacions segures.

mod_access Per restringir l'accés.

mod_auth Configuració de les autentificacions.

mod_cgi Configuració del CGI.

mod_deflate Compressió dinàmica del contingut enviat al client.

mod_evhost Per fer més gran el virtual host.

mod_fastcgi Configuració del FastCGI.

mod_mysql_vhost Virtual host de la base de dades MySQL.

mod_redirect Per redirigir les peticions HTTP.

mod_SCGI Configuració del SCGI.

1.3.3.3. Nginx

Nginx és un servidor web dissenyat per tenir un rendiment alt. A més a més, disposa d'un proxy per protocols de correu electrònic. És de software lliure i de codi obert. Nginx és un sistema Multi-plataforma. Les seves principals característiques són [Wik16p]:

- Servidor d'arxius estàtics i auto-indexat.
- Proxy invers amb opcions de memòria cau.
- Balanceig de la càrrega. Comparteix el treball per realitzar-lo entre varis processos, ordinadors o altres recursos.
- Tolerància a fallades. El sistema pot accedir a la informació tot i que hi hagi alguna anomalia o fallada en el sistema.
- Suport de `SSL`.
- Suport per `FastCGI` amb opcions de memòria cau.
- Virtual Hosting basat en nom o en direcció `IP`.
- Streaming per arxius `FLV` i `MP4`.

L'arquitectura de Nginx és modular. Hi ha diversos mòduls que aporten moltes funcionalitats. Els mòduls més destacats són els següents:

nginx.conf Arxiu on és fa la configuració principal del servidor.

proxy_conf Fitxer on és defineix com a d'actuar el proxy.

fastcgi_conf Configuració del `FastCGI`

mime_types Configuració dels tipus de fitxers que volem que reconegui i treballi el servidor.

1.3.3.4. Comparació i Tria del Servidor

Un cop vaig investigar els servidors, vaig fer la següent taula per comparar les seves prestacions:

Nom	Apache	Lighttpd	Nginx
Sistema Multi-usuari	Si	Si	Si
Llicència	Apache License 2	BSD	BSD
Sistemes Operatius	Multi-plataforma	GNU/Linux, UNIX	Multi-plataforma
Suport Seguretat SSL	Si	Si	Si
Virtual Hosting	Si	Si	Si
Proxy	Si		Si
Mime types	Si	Si	Si

Nom	Apache	Lighttpd	Nginx
Autentificacions	Si	Si	No
Compressió dinàmica	Si	Si	No
Redirecció de peticions HTTP	Si	Si	Si
Filtrat d'aplicació	Si	No	No
Pàgines dinàmiques	Si	Si	Si
Control de tràfic	Si	No	No
Limitador de banda ample	Si	No	No

Taula 1.4.: Comparació dels Servidors

Observant la taula anterior, vaig determinar que el servidor que millor s'adaptarà al projecte serà Apache, per els motius següents:

- És un servidor més complert que els altres dos.
- És un sistema Multi-plataforma.
- Ja hi he treballat abans amb ell, el que fa que pugui treballar millor amb ell i configurar-lo millor.

Per tant per allotjar el projecte, faré servir un servidor Apache.

2. Conceptes Pràctics

Després d'haver investigat totes les eines que necessito utilitzar per desenvolupar aquest projecte (Django, SQLite, HTML, CSS, JavaScript, jQuery, Bootstrap, Apache), vaig poder procedir a desenvolupar el projecte. Per desenvolupar el projecte correctament, vaig tenir de fer les següents implementacions:

1) Creació del projecte `Django`.

2) Creació i implementació de la `Base de Dades`.

3) Creació i implementació del `backend`.

4) Creació i implementació del `frontend`.

5) Exportar el projecte al servidor `Apache`.

Abans de començar el projecte, vaig definir com volia que fos el resultat final de la web, que seria el següent:

- Una part d'administració, des de on l'administrador pot afegir, modificar o eliminar les màquines, fases o vídeos que apareixen en la part d'usuari.
- Una part d'usuari, des de on els usuaris poden visualitzar els tutorials de com es munten les diferents màquines de l'empresa.

A continuació, explicaré com he portat a terme aquestes tasques pas a pas.

2.1. Projecte Django

Per poder començar a desenvolupar el projecte, el primer que vaig haver de fer va ser crear el projecte de `Django`.
Per crear el projecte de Django on emmagatzemaré el projecte, vaig anar al directori on volia que es crees el projecte Django, vaig obrir un terminal i vaig escriure la següent comanda:

```
$ django-admin.py startproject tutorials
```

Aquesta comanda ens crearà un directori anomenat tutorials, i dintre trobem els fitxers `db.sqlite3`, `manage.py` i un altre directori que també s'anomena tutorials, on hi trobem els fitxers `settings.py`, `urls.py` i el `wsgi.py`.
Un cop he creat el projecte Django, he de crear l'aplicació que servirà per desenvolupar la gestió de vídeos del projecte.
Per cada funcionalitat nova que es vulgui que tingui el projecte Django, s'ha de crear una nova aplicació. Per tant, aquest projecte va creixent horitzontalment, i en un futur es podrà anar fent més gran sense problemes.
Per l'aplicació dels vídeos, vaig crear l'aplicació `videos`, amb la comanda:

```
$ python manage.py startapp videos
```

Un cop executada aquesta comanda, es crea un directori nou amb el nom `videos`, on hi haurà els fitxers `urls.py`, `models.py`, `views.py` i `admin.py`. També hi el directori migrations.
Un cop hi ha creats el projecte i l'aplicació, vaig configurar el fitxer `settings.py` [1]. A continuació, explicaré pas a pas la configuració d'aquest arxiu:

```
DEBUG = True

TEMPLATE_DEBUG = True
```

Les variables `DEBUG` i `TEMPLATE_DEBUG` han de ser igual a `True` mentre s'estigui desenvolupant el projecte. Així, quan hi hagi alguna fallada, enlloc de mostrar la pàgina estàndard d'error 404, mostrara la raó del error i de quin fitxer ve aquest error.

```
INSTALLED_APPS = (
    #admin stuff
    'grappelli',
    'grappelli_modeltranslation',

    #'filebrowser',
    'djrill',

    #plugins
    'modeltranslation',
    'rosetta',

    #admin default
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    #own applications
    'videos',
)
```

L'apartat `INSTALLED_APPS`, és on es tenen d'afegir totes les aplicacions que es necessiten i/o es vulguin fer servir per el desenvolupament del projecte. Les aplicacions que faré servir per aquest projecte es poden dividir en cinc parts:

admin stuff Aplicació que servirà per gestionar la part d'administració de la pàgina. En aquest cas, faré servir l'aplicació de `grappelli`. Més endavant, explicaré que he tingut de fer i descarregar per poder utilitzar el `grappelli`.

filebrowser És l'explorador d'arxius que faré servir. En aquest cas, faré servir el `djrill`.

plugins Són les funcionalitats addicionals que es vol que tingui la web. En aquest cas li he afegit `modeltranslation`, per si en un futur, és vol fer la web en més d'un idioma. I el `rosetta`, que serveix per fer traduccions d'un idioma a un altre.

admin default Aplicacions que faig servir perquè `Django` crei l'administrador amb totes les funcionalitats que es desitgen. En aquest cas, li he posat les aplicacions:

- `django.contrib.admin` Per activar la interfície d'administrador que proporciona `Django`.
- `django.contrib.auth` Per activar el sistema d'autentificació de `Django`.
- `django.contrib.contenttypes` Un `framework` per tots els tipus de contingut.
- `django.contrib.sessions` Un `framework` per controlar les sessions dels usuaris.
- `django.contrib.messages` Un `framework` per la missatgeria.
- `django.contrib.staticfiles` Un `framework` per la gestió dels arxius estàtics.

own applications És el lloc on s'han d'afegir les aplicacions que hagi creat personalment. En aquest cas, només hi ha una, `videos`.

```
MIDDLEWARE_CLASSES = (
    'django.middleware.gzip.GZipMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
)
```

L'apartat `MIDDLEWARE_CLASSES` [Doc16a], és un marc pel processament de la petició/resposta de `Django`. Cada component del `middleware` és responsable de fer alguna funció específica. Els `middlewares` que faig servir en aquest projecte són:

- `django.middleware.gzip.GZipMiddleware` Comprimeix el contingut per a navegadors que entenguin la compressió `gzip` (tots els navegadors moderns). El contingut no va comprimit si qualsevol de les següents situacions és compleix:
  - El cos del contingut és de menys de 200 bytes de longitud.
  - La resposta ja ha establert la capçalera del `Content-Encoding`.
  - La sol·licitud (navegador) no ha enviat un `Accept-Encoding` de la capçalera que conté la codificació `gzip`.
- `django.contrib.sessions.middleware.SessionMiddleware` Habilita el suport a les sessions dels usuaris.
- `django.middleware.locale.LocaleMiddleware` Habilita la selecció dels llenguatges basat en les dades que rebem de la sol·licitud. El contingut està personalitat per cada usuari.
- `django.middleware.common.CommonMiddleware` Afegeix unes quantes comoditats:
  - Prohibeix l'accés als agents de l'usuari en la configuració `DISALLOWED_USER_AGENTS`, que ha de ser una llista d'objectes d'expressions regulars compilades.

  - Realitza la reescriptura de la URL basat en al configuració APPEND_SLASH i PREPEND_WWW. Si APPEND_SLASH és True i l'adreça URL inicial no acaba amb una barra (/), i aquesta no es troba declarada amb cap fitxer urls.py, la barra s'afegeix automàticament. En canvi, si aquesta URL és troba declarada sense barra, se'ns redirecciona a com de costum.
    Si PREPEND_WWW és True, les URL que no escrivim www, serem redirigits a la mateixa URL amb www.

- `django.middleware.csrf.CsrfViewMiddleware` Afegeix protecció contra Cross Site Request Forgeries (CSRF) mitjançant l'addició de camps de formularis ocults als formularis POST i comprova les sol·licituds perquè tinguin el valor correcte.
- `django.contrib.auth.middleware.AuthenticationMiddleware` Afegeix l'atribut usuari, la qual cosa suposa que l'usuari ha iniciat sessió actualment, i queda autentificat a cada objecte HttpRequest entrant.
- `django.contrib.auth.middleware.SessionAuthenticationMiddleware` Habilita que les sessions dels usuaris quedin invalidades quan les contrasenyes canvien.
- `django.contrib.messages.middleware.MessageMiddleware` Habilita els cookies i sessions basades en el suport de missatges.
- `django.middleware.clickjacking.XFrameOptionsMiddleware` És una protecció per les capçaleres que arribin via X-Frame-Options.

```
ROOT_URLCONF = 'tutorials.urls'

WSGI_APPLICATION = 'tutorials.wsgi.application'
```

La variable ROOT_URLCONF, configura on ha d'anar el projecte de Django a buscar les URL quan hi ha una sol·licitud d'alguna d'elles per part d'algun client. Per tant, quan se li demana alguna URL, Django el va a buscar al fitxer urls.py, que en aquest cas es troba en el directori tutorials. Quan es troba en aquest arxiu, soluciona la sol·licitud que li ha arribat. Si no pot trobar cap resultat que coincideixi amb la URL, dóna l'error 404.
La variable WSGI_APPLICATION, serveix per fer la configuració del mòdul WSGI perquè el servidor i el propi Django pugui configurar-se amb els estàndards de Python.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

L'apartat DATABASES és on defineixo la base de dades que faré servir pel projecte. En aquest cas, com faré servir la base de dades SQLite, ja puc aprofitar la configuració que crea el Django al crear al projecte, ja que per defecte ja fa servir la base de dades SQLite.

```
LANGUAGE_CODE = 'en'

DEFAULT_CHARSET = 'utf-8'
```

```
FILE_CHARSET = 'utf-8'

TIME_ZONE = 'Europe/Madrid'

USE_I18N = True
USE_L10N = True
USE_TZ = True

MODELTRANSLATION_DEFAULT_LANGUAGE = 'en'
LANGUAGES = (
    ('en', u'English'),
    ('ca', u'Catalan'),
)

LOCALE_PATHS = (
    os.path.join(BASE_DIR, 'locale'),
)
```

La variable `LANGUAGE_CODE`, es defineix el llenguatge amb que es farà la implementació del projecte. Aquesta variable només és eficient si es combina amb les variables `USE_I18N` i `USE_L10N`.
Les variables `DEFAULT_CHARSET='utf-8'` i `FILE_CHARSET='utf-8'` són per definir el format de codificació que faré servir per la web i la programació del projecte.
La variable `TIME_ZONE` serveix per definir la zona horaria en que es troba ubicat el projecte, i que en tots els missatges que ens proporcioni surti l'hora correcta en que s'han produït. Aquests missatges poden d'accés, d'error, etc. Així podem saber amb exactitud quan s'ha produït cada esdeveniment relacionat amb el projecte.
La variable `MODELTRANSLATION_DEFAULT_LANGUAGE` serveix per definir el llenguatge per defecte que ha de fer servir la pàgina. Jo hi he configurat el anglès. Però en aquests moments, en totes les pàgines es mostraran en el mateix idioma, ja que no he implementat més d'un llenguatge.
En la variable `LANGUAGES`, es defineixen tots els llenguatges que volem que disposi la web.
I la variable `LOCALE_PATHS` serveix per definir on es troben els fitxers del projecte.

```
STATIC_ROOT = os.path.join(BASE_DIR, 'wsgi', 'static', 'root')
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'wsgi/static'),
)

STATIC_URL = '/static/'

STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)
```

Totes aquestes variables serveixen per gestionar els arxius estàtics [Doc16b] i treballar amb ells:

- `STATIC_ROOT` És el directori on es guardaran els arxius estàtics després de fer un `python manage.py collectstatic`, que serveix per posar els arxius estàtics dintre aquest directori, i crear correctament tot lo relacionat amb els arxius estàtics. En aquest cas, es

trobaran en el `path`, desde la base del projecte en `wsgi/static/root`. La carpeta `wsgi` i els directoris que van dintre seu els he de crear jo manualment.

- `STATICFILES_DIRS` És el directori on es trobaran tots els arxius estàtics que necessitem per el projecte. En aquest cas, es trobaran en el directori `wsgi/static`.
- `STATIC_URL` Variable que serveix per declarar el `path` fins als fitxers estàtics de manera més curta. És a dir, enlloc d'escriure el `path` absolut fins als fitxers estàtics, escrivint `static` ja és suficient.
- `STATICFILES_FINDERS` Llista de cercadors del `backend` per poder trobar els arxius estàtics en vàries localitzacions. En faig servir dos:
  - `django.contrib.staticfiles.finders.FileSystemFinder` Cercador que busca en els arxius que es troben en el directori `STATICFILES_DIRS`.
  - `django.contrib.staticfiles.finders.AppDirectoriesFinder` Cercador que busca els arxius estàtics en els subdirectoris de cada aplicació.

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'wsgi', 'media')

MEDIA_URL = '/media/'

VIDEO_TYPES = 'mp4'

IMAGE_TYPES = 'jpg'
```

La variable `MEDIA_ROOT` serveix per definir on es guardaran els arxius dinàmics, és a dir, els que podran anar afegint els administradors de la web, que seran les imatges i vídeos. En aquest cas, aquests arxius es trobaran en el directori `wsgi/media`.
La variable `MEDIA_URL` serveix per declarar el `path` fins als fitxers dinàmics sense tenir de escriure el `path` absolut, si no escrivint només `media`.
La variable `VIDEO_TYPES`, serveix per declarar els tipus d'arxius de vídeo que es podran guardar en el nostre projecte. En aquest cas, només s'accepten els que són de tipus `mp4`.
La variable `IMAGE_TYPES`, serveix per declarar els tipus d'arxius d'imatge que es podran guardar en el nostre projecte. En aquest cas, només s'accepten els que són de tipus `jpg`.

```
TEMPLATE_DIRS = (
    os.path.join(BASE_DIR, 'templates'),
)

TEMPLATE_CONTEXT_PROCESSORS = TCP + (
    'django.core.context_processors.i18n',
    'django.core.context_processors.request',
    'django.contrib.auth.context_processors.auth',
    'django.core.context_processors.media',
)
```

La variable `TEMPLATE_DIRS` serveix per definir on es guardaran els arxius de frontend, és a dir, el que visualitzarà un usuari quan faci servir la web. En aquest cas, aquests arxius es guardaran en el directori `templates`, que he de crear manualment.

La variable TEMPLATE_CONTEXT_PROCESSORS serveix per configurar com respondran els templates quan s'interactuï amb ells. Per aquest projecte, he fet la següent configuració:

- `django.core.context_processors.i18n` Per definir el llenguatge que és farà servir en la web.
- `django.core.context_processors.request` Per processar les sol·licituds dels clients.
- `django.contrib.auth.context_processors.auth` Per gestionar les autentificacions.
- `django.core.context_processors.media` Per gestionar els arxius dinàmics que s'han de mostrar en la pàgina que es trobi l'usuari en aquell moment.

```
MANAGERS = ('Daniel Varo Garcia', 'danivaro35@gmail.com')

AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)
```

La variable MANAGERS serveix perquè, si en algun moment no es pogués accedir a la web per algún motiu, surt un missatge de que s'ha produït un error, i posa que pots contactar amb el responsable de la web, posant el seu email. En aquest cas, el email que apareixeria seria danivaro35@gmail.com.
La variable AUTHENTICATION_BACKENDS, és una llista de classes que es fan servir quan s'intenta autentificar un usuari.
En aquest projecte, només cal afegir la classe `django.contrib.auth.backends.ModelBackend` per fer aquesta feina d'autentificar l'usuari.

```
EMAIL = 'danivaro35@gmail.com'
PASSWORD = '********'
```

Les variables EMAIL i PASSWORD serveixen per configurar el correu on es volen rebre els missatges que enviïn els usuaris des de la plataforma web. En aquest cas, el email on s'enviarien seria danivaro35@gmail.com.

```
LOGIN_URL = "/en/login"
LOGIN_REDIRECT_URL = "/en/login"
```

Les variables LOGIN_URL i LOGIN_REDIRECT_URL serveixen per, si s'intenta accedir a una URL que necessita que l'usuari estigui autentificat, no el deixarà veure l'adreça introduïda sense abans autentificar-se.
Un cop feta la configuració del `settings.py`, vaig crear la base de dades en SQLite, tal com està definida en el `settings.py`. Per crear la base de dades, s'ha de fer la comanda:

```
$ python manage.py syncdb
```

Al acabar de crear la base de dades, ens pregunta si volem crear un superusuari. Vaig crear-lo, ja que sense ell no podria entrar a la part d'administració de Django.
Un cop creada la base de dades, vaig configurar el arxiu `urls.py` que hi ha en el directori `tutorials`. La web disposarà de dues parts:

- Part d'usuari Serà la part web que podran visualitzar els usuaris estàndards.

- Part d'administració Serà la part on només podran accedir els administradors de la web, per afegir màquines, vídeos, fases i usuaris.

Des de el directori de l'aplicació (`videos`), s'han de crear els atributs que es volen que es creïn en la base de dades. Per crear aquests atributs, és fa des de l'arxiu `models.py`. En l'apartat 2.2.6, explicaré la configuració d'aquest arxiu i la creació d'aquests atributs per la base de dades.
Després d'haver creat tots els atributs que es vol que apareguin en la base de dades, s'han de fer les següents comandes:

```
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py syncdb
```

Aquestes comandes serveixen per crear en la base de dades els atributs que s'han definit en el `models.py`.
Després de configurar l'arxiu `models.py`, vaig crear els directoris que havia definit anteriorment en el `settings.py` i que encara no estaven creats. Aquests directoris són:

- templates
- wsgi. I dintre d'aquest els directoris:
  - root
  - css
  - js
  - images
  - media

Per poder visualitzar un administrador de `Django` que estigui customitzat, vaig fer servir `grappelli`. `Grappelli` és una aplicació de `Django` que et serveix per crear un entorn d'administració. Per fer-lo servir, s'ha de posar en l'apartat `INSTALLED_APPS` del `settings.py`, i també s'ha de posar en el directori `wsgi/static`. S'han d'afegir els directoris `grappelli` i `grappelli_modeltranslation`. El primer és per l'entorn d'administració, i el segon per poder fer la web en més d'un llenguatge.
Al acabar de afegir aquests dos directoris, el projecte de `Django` ja està preparat perquè es puguin començar a fer les configuracions per la web.

Al acabar de crear el projecte `Django`, l'estructura de fitxers és la següent:

```
templates/
tutorials/
    __init__.py
    settings.py
    urls.py
    wsgi.py
    views.py
videos/
    migrations/
    models.py
    tests.py
    urls.py
    __init__.py
    admin.py
    views.py
wsgi/
    media/
        fotos_maquines/
        fotos_video/
        videos/
    static/
        admin/
        css/
        filebrowser/
        grappelli/
        grappelli_modeltranslation/
        img/
        js/
        root/
db.sqlite3
manage.py
```

2.2. Base de Dades

Un cop vaig tenir creada la base del projecte `Django`, vaig tenir de crear les taules i els atributs de cada una d'elles per gestionar les dades en la base dades.
Com `Django` ja proporciona una taula per administrar els usuaris, d'aquesta part de la base de dades no m'havia de preocupar. El que tenia d'implementar jo era com gestionar les màquines, les fases i els vídeos que es podrien afegir a la web.
Per crear les taules i els seus atributs, es té de fer en el fitxer `models.py` [2] que es troba en el directori `videos`. Aquest arxiu l'explicaré pas a pas en l'apartat [2.2.6].

2.2.1. Usuaris

Com que aquesta taula i els seus atributs ja me'ls proporciona l'administrador de `Django`, no hem tinc de preocupar de crear la taula i els seus atributs.

Django ja crea la taula `User`, amb els següents atributs [Doc16c]:

username El nom d'usuari que tindrà cada treballador per accedir a la web. És obligatori i ha de tenir com a màxim 30 caràcters. En el cas del projecte, he definit que el nom d'usuari sigui un email. Per tant, cada nom d'usuari ha de ser un email.

first_name És el nom real de l'usuari. Tot i que el `Django` el considera opcional, per el projecte és obligatori omplir aquest camp, ja que l'utilitzaré per enviar correus quan l'usuari estigui autentificat, com explicaré en l'apartat [2.4.3.11].

last_name Els cognoms reals de l'usuari. Tot i que el `Django` el considera opcional, per el projecte és obligatori omplir aquest camp, ja que el farem servir per enviar correus quan l'usuari estigui autentificat, com explicaré en l'apartat [2.4.3.11].

email El correu de l'usuari. Tot i que el `Django` el considera opcional, per el projecte és obligatori omplir aquest camp, ja que l'utilitzaré per enviar correus quan l'usuari estigui autentificat, com explicaré en l'apartat [2.4.3.11].

password La contrasenya que tindrà l'usuari per entrar en la web. Aquest camp és obligatori.

groups Per posar l'usuari en algun grup que haguem creat. Per aquest projecte, aquest camp no el faré servir.

user_permissions Camp per donar permisos a un usuari. Aquests permisos poden ser:

- **is_staff** Camp per designar que aquell compte d'usuari pot accedir a la part web d'administració.
- **is_active** Camp per designar si aquell usuari està actiu per accedir a la web, tant a la part d'usuari com de l'administració.
- **is_superuser** Camp per designar si aquell usuari té tots els permisos sense tenir d'assignar-los explícitament.

last_login Camp que ens informa de l'últim cop que s'ha autentificat l'usuari.

date_joined Camp que ens diu la data de quan s'ha creat l'usuari.

2.2.2. Màquines

Per gestionar les màquines que podria haver en la base de dades, vaig crear la taula `Màquina`. De cada màquina, per gestionar-les correctament, necessito guardar:

- L'identificador únic de la màquina. Aquest el crea automàticament `Django` i no cal que el crei jo.
- Nom de la màquina
- Una imatge de la màquina. En la base de dades per això, guardaré una referència a la imatge, no la imatge en si. La imatge és guardarà en el directori `wsgi/media/fotos_maquines`.

Per tant, crearé la taula `Màquina`, amb els atributs `name` i `image`.

2.2.3. Fases

Per gestionar les fases, que són les parts en que es divideix el muntatge de la màquina, vaig crear la taula `Fase`.
De cada fase, per gestionar-les correctament, necessito guardar:

- L'identificador únic de la fase. Aquest el crea automàticament `Django` i no cal que el crei jo.
- Nom de la fase.
- Nom de la màquina de la que forma part la fase.
- Posició que ocupa aquella fase respecte les altres fases d'aquella màquina.

Per tant, crearé la taula `Fase`, amb els atributs `name_phase, name_machine` i `num_ordre`. La relació que he establit entre la taula `Màquina` i `Fase` és de `1:N`, ja que una màquina pot tenir N fases, però una fase només pot estar un una màquina.

2.2.4. Vídeos

Per gestionar els vídeos, que són les parts en que es divideixen les diferents fases d'una màquina, vaig crear la taula `Video`.
De cada vídeo, per gestionar-lo correctament, necessito guardar:

- L'identificador únic del vídeo. Aquest el crea automàticament `Django` i no cal que el crei jo.
- Nom del vídeo.
- L'arxiu de vídeo. En la base de dades guardaré una referència al vídeo. L'arxiu de vídeo el guardaré en el directori `wsgi/media/videos`.
- Imatge que representi el vídeo. En la base de dades guardaré una referència a la imatge. La imatge la guardaré en el directori `wsgi/media/fotos_videos`.

De cada vídeo, a més a més de lo especificat abans, també necessito saber en quina fase es troba, i en quina posició dintre d'aquella fase. Com que un vídeo pot estar en més d'una fase, i una fase pot tenir més d'un vídeo, aquesta relació és de `M:N`. Per tant, al ser una relació `M:N`, no puc posar un altre atribut a la taula `Video` per dir quina posició es trobarà dintre de la fase. Tinc de crear un altre taula per controlar aquesta relació, la qual serà la taula `Muntatge`.

2.2.5. Muntatge

La taula `Muntatge` serveix per gestionar en quines fases es troba un vídeo. Com he dit en l'apartat anterior, aquesta relació és de `1:N`. També es pot observar que que la taula Muntatges té una relació de `1:N` amb la taula Maquina, ja que una màquina pot tenir N muntatges, i un muntatge pot tenir una màquina. I en la relació entre la Fase i Muntatge també és de `1:N`, ja que una fase pot estar en N muntatges, però un muntatge només pot tenir una fase. Per tant, aquesta relació és una relació ternària, entre les taules Maquina, Fase i Video. Per tant, per poder gestionar bé aquesta relació, i per poder obtenir un resultat únic per cada consulta que fem en aquesta taula, necessito guardar:

- L'identificador únic del vídeo. Aquest el crea automàticament `Django` i no cal que el crei jo.
- L'identificador de la màquina a la que pertany el vídeo.
- L'identificador de la fase de la qual forma part el vídeo.
- L'identificador del vídeo.
- Posició que ocupa el vídeo dintre d'aquella fase.

Creant aquesta taula, ja puc gestionar on va cada vídeo, i posar un mateix vídeo en més d'un lloc.

2.2.6. Models.py

En aquests apartat explicaré pas a pas la configuració que he fet en l'arxiu `models.py` [2], per crear les taules de la base de dades. A les taules de la base de dades, en `Django` s'anomenen models, així que a partir d'ara m'hi referiré d'aquesta manera.

```
class Maquina(models.Model):
    name = models.CharField(verbose_name=u"Nom", max_length=128)
    image = models.ImageField(storage=OverwriteStorage(),
        upload_to='fotos_maquines', verbose_name=u"Imatge")

    def __str__(self):
        return "%s"%self.name

    class Meta:
        verbose_name = u"Maquina"
        verbose_name_plural = u"Maquines"
```

La classe `Maquina` és el model que es crearà a la base de dades que es dirà `Maquina`. Els atributs que té són els següents:

name És un `CharField`, que vol dir que és un *string*, i fa referència al nom de la màquina. El `verbose_name` és el que ens apareixerà en l'administrador quan vulguem afegir una màquina. El `max_length` és la longitud màxima del nom és de 128 caràcters.

image És un `ImageField`, que vol dir que és una imatge. Per tant, aquí qualsevol arxiu que no tingui extensió d'imatge serà rebutjat i sortirà un missatge d'error en l'administrador. `storage=OverwriteStorage()` és una classe per sobreescriure una imatge amb el mateix nom que explicaré més endavant. `upload_to` és el directori on és guardaran les imatges de la màquina, que en aquest cas serà `fotos_maquina`.

def __str__(self) Funció que serveix a l'administrador per dir el nom de la màquina que s'ha creat, modificat o eliminat.

class Meta Classe que serveix per quan s'interactua amb el model `Maquina` des de l'administrador, que ens mostri el nom del model amb la que s'està interactuant.

```
class Fase(models.Model):
    name_phase = models.CharField(verbose_name=u"Nom de la fase", max_length=128)
    name_machine = models.ForeignKey(Maquina, verbose_name=u"Maquina on es troba",
        related_name='phase_machine')
    num_ordre = num_ordre = models.CharField(verbose_name=u"Posicio de la fase
        dintre la maquina", max_length=128)

    def __str__(self):
        return "%s"%self.name_phase

    class Meta:
        verbose_name = u"Fase"
        verbose_name_plural = u"Fases"
```

La classe `Fase` és el model que es crearà a la base de dades que es dirà `Fase`. Els atributs que té són els següents:

name_phase És un `CharField` que fa referència al nom de la fase.

name_machine És un `ForeignKey` (clau forànea), que vol dir que apunta a un altre model, en aquest cas, el model `Maquina`.

num_ordre És un `CharField` que indica quina posició ocupa aquella fase dintre de la màquina. S'ha d'anar amb compte de que dintre d'aquella màquina no se li posi la mateixa posició a dues fases, ja que si no no s'ordenaran correctament. No he pogut fer aparèixer un missatge d'error, perquè el valor que s'introdueix s'ha de poder repetir per altres màquines. Per tant, s'ha d'anar en compte al introduir aquest valor.

def __str__(self) Funció que serveix a l'administrador per dir el nom de la fase que s'ha creat, modificat o eliminat.

class Meta Classe que serveix per quan s'interactua amb el model `Fase` des de l'administrador, que ens mostri el nom del model amb la que s'està interactuant.

```
class Video(models.Model):
    name = models.CharField(verbose_name=u"Nom", max_length=128)
    video = models.FileField(storage=OverwriteStorage(), upload_to='videos',
        validators=[validate_file_type],verbose_name=u"Video")
    photo = models.ImageField(storage=OverwriteStorage(), upload_to='fotos_video',
        validators=[validate_image_type], verbose_name=u"Imatge")
    instruccions = models.TextField(verbose_name=u"Instruccions")

    def __str__(self):
        return "%s"%self.name

    class Meta:
        verbose_name = u"Video"
        verbose_name_plural = u"Videos"
```

La classe `Video` és el model que es crearà a la base de dades que es dirà `Fase`. Els atributs que té són els següents:

name És un `CharField` que fa referència al nom del vídeo.

video És un `FileField`, que vol dir que es pot qualsevol tipus d'arxiu. Per acceptar només els arxius `mp4`, que són els únics que s'han de poder afegir en aquest atribut, faré servir el `validators=[validate_file_type]`, que és una funció que explicaré més endavant. I el directori on es guardaran els vídeos serà `videos`.

photo És un `ImageField`. `storage=OverwriteStorage()` és una classe per sobreescriure una imatge amb el mateix nom que explicaré més endavant. `upload_to` és el directori on és guardaran les imatges del vídeo, que en aquest cas serà `fotos_video`.

instruccions És un `TextField`, que vol dir que aquí es pot escriure tot els text que es desitgi, a diferència del `CharField`, que es per escriure només un nom o frase.

def __str__(self) Funció que serveix a l'administrador per dir el nom del vídeo que s'ha creat, modificat o eliminat.

class Meta Classe que serveix per quan s'interactua amb el model `Video` des de l'administrador, que ens mostri el nom del model amb la que s'està interactuant.

```
class Muntatge(models.Model):
    id_maquina = models.ForeignKey(Maquina, verbose_name=u'Nom de la maquina',
        related_name='id_machine')
    id_fase = models.ForeignKey(Fase, verbose_name=u'Nom de la fase',
        related_name='id_phase')
    id_video = models.ForeignKey(Video, verbose_name=u'Nom del video',
        related_name='id_video')
    num_ordre = models.CharField(verbose_name=u"Posicio on va aquest video",
        max_length=128)

    def __str__(self):
        return "%s"%self.num_ordre

    class Meta:
        verbose_name = u"Muntatge"
        verbose_name_plural = u"Muntatges"
```

La classe `Muntatge` és el model que es crearà a la base de dades que es dirà `Muntatge`. Els atributs són els següents:

id_maquina És una `ForeignKey`, que en aquest cas apunta al model `Maquina`.

id_fase És una `ForeignKey`, que en aquest cas apunta al model `Fase`.

id_video És una `ForeignKey`, que en aquest cas apunta al model `Video`.

num_ordre És un `Charfield` que indica quina posició ocupa aquell vídeo dintre de la fase. S'ha d'anar amb compte de que dintre d'aquella fase no se li posi la mateixa posició a dos vídeos, ja que si no, no s'ordenaran correctament. No he pogut fer aparèixer un missatge d'error, perquè el valor que s'introdueix s'ha de poder repetir per altres fases. Per tant, s'ha d'anar en compte al introduir aquest valor.

def __str__(self) Funció que serveix a l'administrador per dir el nom del muntatge que s'ha creat, modificat o eliminat.

class Meta Classe que serveix per quan s'interactua amb el model `Muntatge` des de l'administrador, que ens mostri el nom del model amb la que s'està interactuant.

```
def validate_file_type(upload):
    #Fem que el fitxer de pujada sigui accessible per analitzar-lo per guardar-lo
        com a arxiu temporal
    file_name = str(upload)
    #Agafem la extensio del fitxer per veure si es el que ens interessa
    extension = file_name.split('.', 1)[1]
    #Fer sortir un error de validacio si el fitxer que pujem no es del tipus que
        volem acceptar
    if extension not in settings.VIDEO_TYPES:
        raise ValidationError('File type not supported. MP4 recommended')
```

La funció `validate_file_type(upload)` serveix per quan s'afegeix un arxiu en l'atribut `video` del model `Video`, comprovar si és del tipus que ens interessa. Això s'ha de fer amb aquest atribut, perquè accepta qualsevol tipus de arxiu. Per tant, perquè només accepti els arxius `mp4`, he creat aquest validador, que el que fa és, abans de guardar l'arxiu, mira la seva extensió. Sinó és `mp4`, fa sortir un missatge d'error en l'administrador, dient que només acceptarà arxius en format `mp4`.

```
def validate_image_type(upload):
    name = str(upload)
    extension = name.split('.', 1)[1]
    if extension not in settings.IMAGE_TYPES:
        raise ValidationError('File type not supported. JPG recommended')
```

La funció `validate_image_type(upload)` serveix per quan afegim un arxiu en l'atribut `image` del model `Maquina` o `photo` del model `Video`, comprovar si és del tipus que ens interessa. Faig aquesta comprovació perquè ser que les imatges en `jpg` es mostren correctament en la web, per tant no deixo cap altre format perquè no hi aparegui cap problema.

```
class OverwriteStorage(FileSystemStorage):

    def get_available_name(self, name):
        # Si el nom del fitxer existeix, borra el que hi havia i guarda la nova imatge
        if self.exists(name):
            os.remove(os.path.join(settings.MEDIA_ROOT, name))
        return name
```

La classe `OverwriteStorage(FileSystemStorage)` serveix per quan es vol guardar una imatge o vídeo, primer mira si una imatge amb el mateix nom existeix en aquell directori. Si existeix, elimina la imatge o vídeo antic i guarda la nova imatge o vídeo. Aquesta classe és necessària de fer ja que sinó en aquell directori hi hauria imatges o vídeos que no són necessaris, ja que no s'utilitzarien.

```
@receiver(pre_delete, sender=Maquina)
def maquina_delete(sender, instance, **kwargs):
    instance.image.delete(False)
```

```
@receiver(pre_delete, sender=Video)
def video_delete(sender, instance, **kwargs):
    instance.video.delete(False)
    instance.photo.delete(False)
```

La funció `maquina_delete` serveix per quan s'elimina una màquina, que la seva imatge també s'elimini del directori `fotos_maquina`. I la funció `video_delete` serveix per eliminar el vídeo i l'imatge d'aquell vídeo quan s'elimina el vídeo. Sense aquestes dues funcions, hi hauria imatges i vídeos que no es farien servir guardats en els directoris.

2.2.7. Esquema UML de la Base de Dades

Un cop vaig definir totes els models que necessitava per desenvolupar el projecte i les seves relacions, vaig poder fer el `Diagrama Entitat-Relació`, que representa la relació entre els diferents models de la base de dades, és el següent:

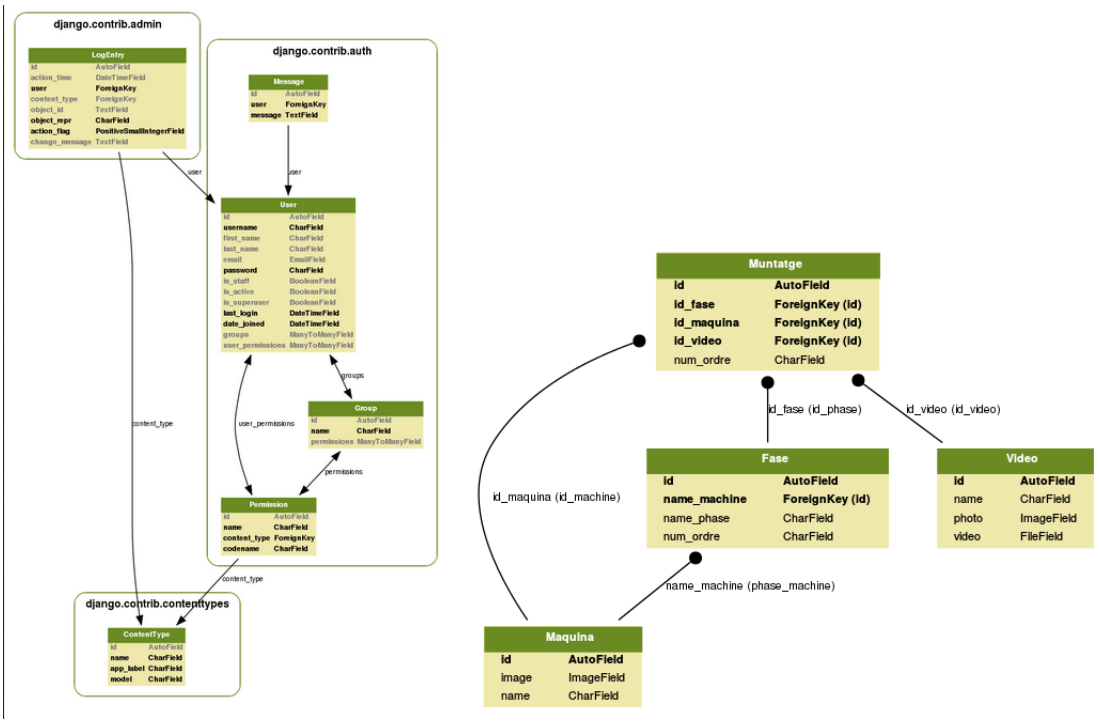

Figura 2.1.: Diagrama Entitat-Relació

Com es pot observar, hi ha dues representacions:

- Una per la part d'usuari, amb totes les relacions necessàries per per poder interactuar correctament amb totes les parts de la web que necessiten el model `User`.
- I una altre per la part dels vídeos, on es veuen totes les relacions que he comentat anteriorment entre els models `Maquina, Fase, Video` i `Muntatge`.

2.3. Backend

Un cop creat el projecte de `Django` i la base de dades, ja vaig poder implementar el `backend`. El `backend` que crearé es pot dividir en tres parts:

- El `backend` del projecte `tutorials`.
- El `backend` de l'aplicació `videos`.
- El directori `wsgi`, on hi guardaré els fitxers estàtics i dinàmics.

2.3.1. Projecte Tutorials

Per fer les configuracions pertinents perquè el projecte funcioni correctament, en el directori `tutorials` s'han de configurar els fitxers `views.py`, `urls.py` i `forms.py`. Els fitxers `wsgi.py` i `settings.py` no els configuraré en aquest apartat, el primer perquè ja el configura automàticament `Django`, i el segon perquè ja l'he configurat en l'apartat 2.1.

2.3.1.1. views.py

En el fitxer `views.py` [17], el que faig és treballar amb la base de dades per satisfer les peticions dels usuaris, i poder presentar les dades dinàmicament. En aquest views es treballen les parts genèriques de la pàgina web, que són la pàgina d'autentificació, el `logout` i les pàgines de contacte. Aquestes parts són les següents:

```
class HomeView(TemplateView):
    template_name = 'index.html'

    def get(self, request, *args, **kwargs):
        return render(request, 'index.html')

    def post(self, request, *args, **kwargs):
        email = request.POST['email']
        password = request.POST['password']
        user = authenticate(username=email, password=password)

        # Si l'usuari existeix
        if user is not None:
            # Si l'usuari esta actiu
            if user.is_active:
                login(request, user)
                redirection = reverse('machines_list')
                try:
                    if request.GET["next"]:
                        redirection = request.GET["next"]
                except:
                    pass
                return HttpResponseRedirect(redirection)
            else:
                return render(request, 'index.html', {'error': 'Compte no actiu:
                    Contactar amb el gestor de la web'})
        else:
```

```
            return render(request, 'index.html', {'error': 'Usuari i/o contrasenya
                incorrectes'})
```

La classe `HomeView` el que fa és presentar la pàgina d'autentificació. Et demana l'usuari i la contrasenya. Un cop s'han introduït les dades, comprova que l'usuari existeixi i que estigui actiu. Si es compleixen les dues coses, l'usuari serà redirigit a la pàgina principal de la web perquè pugui consultar el que desitgi. Si no es compleix alguna d'aquestes dues condicions, es mostra un missatge de l'error que s'ha produït i no et deixa avançar a la pàgina següent. Totes aquestes dades es processen en el template `index.html` [2.4.3.2].

```
class Logout(TemplateView):
    # Logout and redirect
    def get(self, request, *args, **kwargs):
        logout(request)
        return HttpResponseRedirect('/en/login/')
```

La classe `Logout` el que fa és que si un usuari ha iniciat sessió, aquest la pugui tancar.

```
class ContactView(TemplateView):
    template_name = 'contactar.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        user=request.user
        nom_treballador = user.first_name
        cognom_treballador = user.last_name
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'contactar.html', {'nom_treballador': nom_treballador,
            'cognom_treballador':cognom_treballador,})

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data["name"]
            surname = form.cleaned_data["surname"]
            email = form.cleaned_data["email"]
            text = form.cleaned_data["text"]

            mes = "Nou missatge"
            mes += "\n Nom: %s" % name
            mes += "\n Cognoms: %s" % surname
            mes += "\n Email: %s" % email
            mes += "\n Text: %s" % text

            server = smtplib.SMTP('smtp.gmail.com',587)
            server.starttls()
            server.login(settings.EMAIL, settings.PASSWORD)
            server.sendmail(email, settings.EMAIL, mes)
            server.quit()
            return render(request, 'contactar.html', {'send':True,'nom_treballador':
                name, 'cognom_treballador':surname,})
        else:
```

```
        errors = []
        for field in form:
            for error in field.errors:
                errors += error
        return render(request, 'contactar.html', {'errors': errors,})
```

El que fa la classe `ContactView` és permetre que, quan un usuari ha iniciat sessió, és pugui posar en contacte amb l'administrador per comunicar-li qualsevol dubte, problema o observació que pugui tenir respecte la pàgina web.
El `@method_decorator(login_required)` serveix perquè aquesta pàgina només es mostri quan l'usuari està autentificat, si no ho està, se li mostra la pàgina d'autentificació. Un cop s'autentifica l'usuari, aquest serà redirigit a aquesta pàgina automàticament.
En la funció `get`, el que faig es agafar les dades de l'usuari (nom, cognoms i email) perquè sàpiga el que es mostrarà en el correu que rebi l'administrador.
I en la funció `post`, el que faig és agafar les dades que apareixen en el `template` (nom, cognoms, email i text) i enviar-li el correu amb tota aquesta informació a l'administrador. En aquests moments, fins que en l'empresa ho posi en funcionament, per provar-ho, m'envio els correus al meu email personal. Les dades que es processen són les del template `contactar.html` [2.4.3.11].

```
class Contact2View(TemplateView):
    template_name = 'contactar2.html'

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data["name"]
            surname = form.cleaned_data["surname"]
            email = form.cleaned_data["email"]
            text = form.cleaned_data["text"]

            mes = "Nou missatge"
            mes += "\n Nom: %s" % name
            mes += "\n Cognoms: %s" % surname
            mes += "\n Email: %s" % email
            mes += "\n Text: %s" % text

            server = smtplib.SMTP('smtp.gmail.com',587)
            server.starttls()
            server.login(settings.EMAIL, settings.PASSWORD)
            server.sendmail(email, settings.EMAIL, mes)
            server.quit()
            return render(request, 'contactar2.html', {'send':True,})
```

El que fa la classe `Contact2View` és permetre que, quan un usuari no pot accedir a la pàgina web, li pugui enviar un email a l'administrador, escrivint el seu nom, cognoms i email, perquè el pugui informar. També l'usuari ha d'escriure el missatge que li apareixerà a l'administrador, perquè aquest sàpiga exactament el problema que té. Les dades que es processen es troben en el template `contactar2.html` [2.4.3.3].

2.3.1.2. forms.py

En el fitxer `forms.py` [18], el que faig és definir quines variables aniran en els formularis de contacte, per si en falta alguna per introduir, que li aparegui un error a l'usuari perquè aquest el corregeixi. La definició del `forms.py` és la següent:

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(max_length=128, required=True)
    surname = forms.CharField(max_length=128, required=True)
    email = forms.EmailField(required=True)
    text = forms.CharField(max_length=512, required=True)
```

Com es pot observar, les variables que demano per el formulari de contacte són el nom, cognoms, email i el comentari que se li vulgui escriure a l'administrador.

2.3.1.3. urls.py

En el fitxer `urls.py` [19], el que faig és definir les `URL's` que tindrà el projecte. En aquest `urls.py`, definim les `URL's` generals del projecte. Per tant defineixo les següents `URL's`:

```
urlpatterns += i18n_patterns('',

    (r'^grappelli/', include('grappelli.urls')),
    url(r'^admin/', include(admin.site.urls)),

    #pagina principal
    url(r'^login/', HomeView.as_view(), name='home'),
    url(r'^tutorials/', include('videos.urls')),

    url(r'^contactar/', ContactView.as_view(), name='contactar'),
    url(r'^contactar2/', Contact2View.as_view(), name='contactar2'),
    url(r'^logout/', Logout.as_view(), name='logout'),
) + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

En `urlpatterns`, el que faig és definir tots els patrons de `URL` que vull que contingui el projecte. Aquí defineixo els `URL's` de les views, perquè es mostrin quan la demani. En aquest `URL's` defineixo la pàgina d'autentificació, el logout i les pàgines de contactar. També faig un redirecció, que quan es demani una `URL` que al davant tingui `tutorials`, aquest es redirigeix cap al `urls.py` de l'aplicació `videos`.

2.3.2. Aplicació Vídeos

Per fer les configuracions pertinents perquè l'aplicació `videos` funcioni perfectament dintre de el projecte `tutorials`, s'han de configurar els fitxers `views.py` i `urls.py`, a part dels fitxers `admin.py` que està explicat en l'apartat 2.4.1 i `models.py`, que està explicat en l'apartat 2.2.6.

2.3.2.1. views.py

En el fitxer `views.py` [20], el que faig és treballar amb la base de dades per satisfer les peticions dels usuaris, i poder presentar les dades dinàmicament. En aquest `views` es treballen les parts

concretes de l'aplicació `videos`. Aquestes parts són:

- Presentar les màquines de les que es disposa tutorials.
- Presentar les fases que formen la màquina.
- Presentar els vídeos que formen part d'una fase.
- Presentar una pàgina on es reprodueixen els vídeos.
- Pàgina per poder observar les dades de l'usuari.
- Pàgina per modificar les dades de l'usuari.

L'explicació de cada part és la següent:

```
class MachinesDataView(TemplateView):
    template_name = 'maquines.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        #Fer una llista que contingui la llista de videos i muntatge per recorre-la a
            la vegada
        return render(request, 'maquines.html', {'current_machines':
            current_machines})
```

La classe `MachinesDataView` el que fa es agafar totes les màquines que hi ha creades en la base de dades. Un cop ha consultat totes les màquines que existeixen en la base de dades, les guarda en la variable `current_machines`, i aquesta variable se li passa al template `maquines.html` [2.4.3.4], i aquest ja s'encarrega de presentar les dades.

```
class PhasesLinkView(TemplateView):
    template_name = 'link_fases.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        current_fases = Fase.objects.all().order_by('num_ordre')
        return render(request, 'link_fases.html',
            {'current_machines':current_machines, 'current_fases':current_fases})
```

La classe `PhasesLinkView` el que fa es agafar totes les màquines i fases existents, i passar-li al template `link_fases.html` [2.4.3.8] perquè pugui presentar les dades.

```
class VideosLinkView(TemplateView):
    template_name = 'link_videos.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        current_fases = Fase.objects.all().order_by('num_ordre')
        current_videos = Video.objects.all()
        current_muntatges = Muntatge.objects.all().order_by('num_ordre')
```

```
        return render(request, 'link_videos.html',
            {'current_machines':current_machines, 'current_fases':current_fases,
            'current_videos': current_videos, 'current_muntatges': current_muntatges})
```

La classe `VideosLinkView` el que fa es agafar totes les màquines, fases, vídeos i muntatges existents i passar-li al template `link_videos.html` [2.4.3.9] perquè pugui presentar les dades.

```
class PhasesDataView(ListView):
    template_name = 'fases_maquina.html'
    context_object_name = 'maquina_actual'
    model = Maquina

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        pk_actual = int(self.kwargs['pk'])
        maquina_actual = Maquina.objects.filter(id=pk_actual)
        #Per obtenir els noms de les fases
        fases = Fase.objects.filter(name_machine_id=pk_actual).order_by('num_ordre')
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'fases_maquina.html', {'fases': fases,
            'maquina_actual':maquina_actual})
```

La classe `PhasesDataView` el que fa es agafar les fases que coincideixen amb la màquina que s'ha escollit. El que fa es agafar aquestes fases, i li passa al `template fases_maquina.html` [2.4.3.5] perquè aquesta les presenti correctament.

```
class MiniVideosDataView(ListView):
    template_name = 'minivideos.html'
    context_object_name = 'minivideos'
    model = Video

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        pk_actual = int(self.kwargs['pk'])
        nom_fase_actual =
            str(Fase.objects.filter(id=pk_actual).values('name_phase')[0]['name_phase'])
        id_fase = str(Fase.objects.filter(id=pk_actual).values('id')[0]['id'])
        id_maquina =
            str(Fase.objects.filter(id=pk_actual).values('name_machine_id')[0]['name_machine_id'])
        nom_maquina_actual =
            Maquina.objects.filter(id=id_maquina).values('name')[0]['name']
        #Obtenim els id dels videos que hem de mostrar, pero en un diccionari
        id_videos = Muntatge.objects.filter(id_maquina=id_maquina,
            id_fase=id_fase).values('id_video').order_by('num_ordre')
        #Obtenim dels id dels muntatges en que es troben els videos que hem de
            mostrar per poder passar a la pagina de video correcte
        id_muntatges = Muntatge.objects.filter(id_maquina=id_maquina,
            id_fase=id_fase).values('id').order_by('num_ordre')
        #Agafem els id dels videos que hem de mostar i els posem en una llista
        x = 0
        id_videos2 = []
        for video in id_videos:
            id_videos2.insert(x, id_videos[x]['id_video'])
```

```
            x += 1
        #Agafem els id dels muntatges que hem de mostrar i els posem en una llista
        z = 0
        id_muntatges2 = []
        for muntatge in id_muntatges:
            id_muntatges2.insert(z, id_muntatges[z]['id'])
            z += 1
        # La variable vid serveix al template per saber si hi ha videos per mostrar o
            no
        y = 0
        vid = 0
        videos_mostrar = []
        while (y < len(id_videos2)):
            videos_mostrar.insert(y, Video.objects.filter(id=id_videos2[y])[0])
            vid = 1
            y += 1
        #Llista de diccionaris on passem el id del muntatge i el video que s'ha de
            mostrar per cada link
        videos_mostrar2 = []
        z = 0
        for x,y in zip(id_muntatges2, videos_mostrar):
            d = {}
            d['id_muntatge'] = x
            d['video'] = y
            videos_mostrar2.insert(z, d)
            z += 1
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'minivideos.html', {'vid':vid,
            'videos_mostrar2':videos_mostrar2, 'nom_maquina_actual':
            nom_maquina_actual, 'nom_fase_actual':nom_fase_actual})
```

La classe `MiniVideosDataView` el que fa és agafar els vídeos que formen part de la fase i la màquina que s'ha escollit. El que fa es agafar els vídeos que s'han de mostrar, la màquina i la fase on es troben, i li passa tota aquesta informació al `template minivideos.html` [2.4.3.6] perquè presenti les dades correctament a l'usuari.

```
class VideoDataView(TemplateView):
    template_name = 'video.html'


    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_pk = int(self.kwargs['pk'])
        #Per saber en el muntatge que volem reproduir el video, i saber la maquina i
            la fase on es troba
        muntatge = Muntatge.objects.filter(id=current_pk)
        #Obtenim la fase del video que hem de reproduir
        fase =
            str(Muntatge.objects.filter(id=current_pk).values('id_fase_id')[0]['id_fase_id'])
        #Obtenim la maquina del video que hem dereproduir
        maquina =
            str(Muntatge.objects.filter(id=current_pk).values('id_maquina_id')[0]['id_maquina_id'])
```

```
#Per obtenir els id dels videos que hem de mostrar, pero en diccionari
id_videos = Muntatge.objects.filter(id_maquina=maquina,
    id_fase=fase).values('id_video').order_by('num_ordre')
#Agafem els id dels videos que hem de mostar i els posem en una llista
x = 0
id_videos2 = []
for video in id_videos:
    id_videos2.insert(x, id_videos[x]['id_video'])
    x += 1
#Agafem els id dels muntatges que s'han de mostrar i els posem en una llista
z = 0
id_muntatges2 = []
for i in id_videos2:
    m = Muntatge.objects.filter(id_video=i,
        id_fase_id=fase).values('id')[0]['id']
    id_muntatges2.insert(z, m)
    z += 1
#Agafem els videos que van en aquella fase en el ordre correcte
y = 0
videos_mostrar = []
while (y < len(id_videos2)):
    videos_mostrar.insert(y, Video.objects.filter(id=id_videos2[y])[0])
    vid = 1
    y += 1
#Numero que serveix per fer la reproduccio automatica
num_videos = len(id_muntatges2)
current_position = id_muntatges2.index(current_pk)
#Per agafar la posicio de la llista de pk al qual tenim d'anar a continuacio
position_seguent_pk = id_muntatges2.index(current_pk) + 1
#Si el seguent pk no existeix
if (current_position >= (len(id_muntatges2))-1):
    next_pk = current_pk
#Si existeix seguent pk
else:
    next_pk = id_muntatges2[position_seguent_pk]
#Adjunto les llistes id_muntatges2 i videos_mostrar per poder posar a
    reproduir el video que toca
i2 = 0
videos = []
for x,y in zip(id_muntatges2, videos_mostrar):
    d = {}
    d['id_muntatge'] = x
    d['video'] = y
    videos.insert(i2, d)
    i2 += 1
#Agafo el video que hem de reproduir, i agafo les seves instruccions per
    mostrar-les en el template
i = videos_mostrar[current_position].instruccions
instruccions = i.split('\r\n')
#Li passo els objectes al html per poder mostrar-los
return render(request, 'video.html', {'videos':videos, 'num_videos':
    num_videos, 'pagina_actual': current_pk, 'pagina_seguent': next_pk,
    'current_position': current_position, 'instruccions': instruccions})
```

La classe `VideosDataView` el que fa és agafar el vídeo de la fase que es reproduir. Els passa al template `video.html` [2.4.3.7], i allà es posa a reproduir el vídeo que s'ha escollit per veure, i al costat dret posa una llista amb tots els vídeos que formen la fase junt amb aquell vídeo. També li passo la posició actual que ocupa el vídeo dintre de la fase, per si es vol fer la reproducció automàtica, el template sàpiga si ha de posar a reproduir un altre vídeo o no.

```
class SearchResultsView(TemplateView):
    template_name = 'search.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        mrname = request.GET.get('mrname')
        #Llista on guardo totes les paraules que ha introduit l'usuari per buscar-les
        mrname_list = mrname.split(',')
        #Llista on guardo les maquines per mostrar al template
        maquines = []
        #Per saber si hi ha mes d'una maquina en el template
        m = 0
        #Llista on guardo les fases per mostrar al template
        fases = []
        #Per saber si hi ha mes d'una fase en el template
        f = 0
        ldf = []
        #Llista on guardo els videos per mostrar al template
        videos = []
        v = 0
        ldv = []
        #Variable per comprovar que no es passen mes de dues consultes a la vegada.
            Si passa aixo, en el template surtira el misstage d'error de masses
            variables
        consultes = 0
        variables_buscades = len(mrname_list)
        if variables_buscades > 2:
            consultes = 1
        #Si la variable consultes es 1, surt error perque nomes es podran cercar com
            a maxim dues paraules a la vegada. S
        if consultes == 1:
            return render(request, 'search.html', {'consultes':consultes})
        else:
            if variables_buscades == 2:
                if mrname_list[0] == mrname_list[1]:
                    del(mrname_list [0])
            for mrname in mrname_list:
                #Consulta per mostar les maquines que coincideixen amb la cerca
                maquines += Maquina.objects.filter(name__contains=mrname)
                #Consulta per buscar les fases que coincideixen amb la cerca
                fases += Fase.objects.filter(name_phase__contains=mrname)
                fases_final = Fase.objects.filter(name_phase__contains=mrname)
                #Fa el mateix que la consulta de dalt, pero ens servira per trobar el nom
                    que correspon a cada maquina, no per buscar les fases
                fases2 = Fase.objects.filter(name_phase__contains=mrname).values()
                #Llista del id de les fases que ens coincideixen amb la cerca, en format
```

```
    diccionari
id_fases_d = Fase.objects.filter(name_phase__contains=mrname).values('id')
s = 0
#Llista on tenim nomes els id
id_fases = []
for e in id_fases_d:
    id_fases.insert(s, id_fases_d[s]['id'])
    s += 1
#Llista de diccionaris que ens servira per passar junts la fase i el nom
    de la maquina on es troba
ld = [{'x':1,'y':2}]
id_maquines = []
nom_maquines = []
x = 0
if len(fases2) > 0:
    for fase2 in fases2:
        id_maquines.insert(x, fases2[x]['name_machine_id'])
        x += 1
    if len(id_maquines) > 0:
        y = 0
        for id_maquina in id_maquines:
            nom =
                str(Maquina.objects.filter(id=id_maquina).values('name')[0]['name'])
            nom_maquines.insert(y, nom)
            y += 1
#Llista de diccionaris on relacionarem la fase amb el nom de la maquina
    on es troba
index_ldf = 0
for x,y,w in zip(id_fases, fases_final, nom_maquines):
    d={}
    d['nom_fase'] = y
    d['id_fase'] = x
    d['maquina'] = w
    ldf.insert(index_ldf, d)
    index_ldf += 1
#Consulta per mostar els videos que coincideixen amb la cerca
videos += Video.objects.filter(name__contains=mrname)
#Ens quedem amb els id dels videos per buscar les fases i maquines a les
    que pertany. En format diccionari
videos2 = Video.objects.filter(name__contains=mrname).values('id')
#Llista on hi ha guardat els numeros dels id dels videos en el format
    correcte
videos_id = []
#Per trobar tots els muntatges que coincideixen amb la cerca
muntatgesv = []
#Llista on hi guardo els id dels muntatges dels videos que coincideixen
    amb la cerca, en diccionari
muntatgesv_id = []
#Llista on hi guardo els id dels muntatges dels videos que coincideixen
    amb la cerca, en format correcte
muntatgesv_id2 = []
muntatgesf = []
muntatgesm = []
```

```
#Llista amb tots els id dels videos que es troben en els muntatges
videos_id2 = []
videos_name = []
#Llista on hi ha guardat els numeros de id de cada maquina a la que
    pertany cada video
maquines_id = []
maquines_name = []
#Llista on hi ha guardat els numeros de id de cada fase a la que pertany
    cada video
fases_id = []
fases_name = []
#Llista de diccionaris on relacionem els videos amb les fases i les
    maquines on es troben
x = 0
if len(videos2) > 0:
    for video2 in videos2:
        videos_id.insert(x, videos2[x]['id'])
        x += 1
    #Per saber els id dels videos, fases i maquines que tenen d'apareixer
        en la cerca, en format diccionari
    for i in videos_id:
        muntatgesv +=
            Muntatge.objects.filter(id_video_id=i).values('id_video_id')
        muntatgesv_id +=
            Muntatge.objects.filter(id_video_id=i).values('id')
        muntatgesf +=
            Muntatge.objects.filter(id_video_id=i).values('id_fase_id')
        muntatgesm +=
            Muntatge.objects.filter(id_video_id=i).values('id_maquina_id')
    #Per obtenir els id dels muntatges en una llista
    i = 0
    for e in muntatgesv_id:
        muntatgesv_id2.insert(i, muntatgesv_id[i]['id'])
        i += 1
    #Per obtenir els id dels videos en una llista
    i2 = 0
    for e in muntatgesv:
        videos_id2.insert(i2, muntatgesv[i2]['id_video_id'])
        i2 += 1
    #Per obtenir els id de les fases en una llista
    i3 = 0
    for e in muntatgesf:
        fases_id.insert(i3, muntatgesf[i3]['id_fase_id'])
        i3 += 1
    #Per obtenir els id de les maquines en una llista
    i4 = 0
    for e in muntatgesm:
        maquines_id.insert(i4, muntatgesm[i4]['id_maquina_id'])
        i4 += 1
    #Recorro les llistes anteriors per aconsegui el nom de cada id de
        maquina,de fase i de video i fer tres llistes noves
    y = 0
    for index in maquines_id:
```

```
                maquina =
                    str(Maquina.objects.filter(id=maquines_id[y]).values('name')[0]['name'])
                maquines_name.insert(y, maquina)
                y += 1
            y = 0
            for index2 in fases_id:
                fase =
                    str(Fase.objects.filter(id=fases_id[y]).values('name_phase')[0]['name_phase'])
                fases_name.insert(y, fase)
                y += 1
            y = 0
            for index3 in videos_id2:
                video = Video.objects.filter(id=videos_id2[y])[0]
                videos_name.insert(y, video)
                y += 1
            #Llista de diccionaris on relacionem els videos amb les fases i les
                maquines on es troben
            z = 0
            for v,w,x,y in zip(muntatgesv_id2, videos_name, maquines_name,
                fases_name):
                d={}
                d['id_muntatge'] = v
                d['video'] = w
                d['maquina'] = x
                d['fase'] = y
                ldv.insert(z,d)
                z += 1
    if len(maquines) > 0:
        m = 1
    if len(fases) > 0:
        f = 1
    if len(videos) > 0:
        v = 1
    #Li passo els objectes al html per poder mostrar-los
    return render(request, 'search.html', {'maquines':maquines, 'm':m,
        'fases':fases, 'f':f, 'ldf':ldf, 'videos':videos, 'ldv':ldv, 'v':v})
```

La classe `SearchResultsView` el que fa és, segons el que hagi escrit l'usuari en el cercador, busca en els noms de les màquines, fases o vídeos. Es poden buscar una o dues paraules a la vegada, separant-les amb una coma. Si troba alguna coincidència, li passa al template `search.html` [2.4.3.10] les màquines, fases o vídeos que coincideixen amb aquell resultat. Després el template ja se'n encarrega de presentar les dades correctament.

```
class PersonalDataView(TemplateView):
    template_name = 'dades_personals.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        return render(request, 'dades_personals.html', {})
```

La classe `PersonalDataView` el que fa només és cridar el `template dades_personals.html` [2.4.3.12], i aquest ja presenta les dades de l'usuari correctament.

```
class ModifyDataView(TemplateView):
    template_name = 'modify_data.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        return render(request, 'modify_data.html', {})

    def post(self, request, *args, **kwargs):
        user = self.request.user
        name = request.POST["name"]
        surname = request.POST["surname"]
        email = request.POST["email"]
        new_password = request.POST["new_password"]
        repeat_new_password = request.POST["repeat_new_password"]
        if name!="" and surname!="" and email!="":
            user.first_name = name
            user.last_name = surname
            user.email = email
            user.save()
            redirection = reverse('personal_data')
            if new_password!="":
                if repeat_new_password!="":
                    if repeat_new_password==new_password:
                        user.set_password(new_password)
                        user.save()
                        user = authenticate(username=user.username,
                            password=new_password)
                        login(request, user)
                    else:
                        error = "Les contrasenyes no coincideixen"
                        return render(request, 'modify_data.html', {'error':error,})
                else:
                    error = "Tornar a repetir contrasenya"
                    return render(request, 'modify_data.html', {'error':error,})

            return HttpResponseRedirect(redirection)

        else:
            error = "Omplir tots els camps obligatoris"
            return render(request, 'modify_data.html', {'error':error,})
```

La classe `ModifyDataView` el que fa és que l'usuari, des de el template `modify_data.html` [2.4.3.13] pugui interactuar amb les seves dades d'usuari que hi ha guardades en la base de dades. Des de aquest template, l'usuari pot modificar el seu nom, cognoms, email personal i la contrasenya per accedir a la web. El que no podrà canviar és el usuari, ja que he fet que aquest només el pugui canviar l'administrador.

```
class PhasesLinkView(TemplateView):
    template_name = 'link_fases.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
```

```
current_machines = Maquina.objects.all()
current_fases = Fase.objects.all().order_by('num_ordre')
return render(request, 'link_fases.html',
    {'current_machines':current_machines, 'current_fases':current_fases})
```

2.3.2.2. urls.py

En el fitxer `urls.py` [21], el que faig és definir les `URL's` que tindrà l'aplicació `videos`. En aquest fitxer, per cada view que s'hagi creat, s'ha de crear un `url` perquè aquesta `view` sigui accessible des de la web.

2.3.3. WSGI

Per guardar els fitxers estàtics i dinàmics del projecte, vaig crear el directori `wsgi`. Dintre d'aquest, vaig crear el directori `media`, per els fitxers dinàmics, i el directori `static`, per els fitxers estàtics.
Els fitxers estàtics són aquells que després de acabar el projecte, canviaran molt poc.
Els fitxers dinàmics són aquells que poden anar canviant constantment.

2.3.3.1. static

En el directori `static`, és on hi ha guardats els fitxers que serveixen per la configuració dels `templates` i de l'administrador.
Per tant en aquest directori, referent als `templates` hi ha els següents directoris:

1) **css**: És on hi ha definits els estils que faran servir els `templates`.

2) **img**: Són les imatges que faran servir els `templates`.

3) **js**: Són les llibreries de `JavaScript` que faran servir els templates per dur a terme alguna tasca.

I referent a l'administrador, els directoris són:

1) **grappelli**: Directori que serveix per crear l'entorn d'administració.

2) **grappelli_modeltranslation**: Directori que serveix per poder crear la web en diferents idiomes.

2.3.3.2. media

En el directori `media`, és on hi ha guardades les imatges i els vídeos que serveixen per poder presentar i reproduir els tutorials. Aquest directori, conté els següents directoris, per tenir les imatges i els vídeos ben ordenats:

1) **fotos_maquines**: És on es guarden totes les imatges de les màquines, per tenir una representació gràfica de com ha de quedar la màquina un cop finalitzada.

2) **fotos_videos**: És on es guarden les imatges dels vídeos, per tenir una representació gràfica de la feina que es fa en aquell vídeo.

3) **videos**: És on es guarden els fitxers de vídeo per després poder posar-los a reproduir des de la web.

Al acabar de configurar tots aquests fitxers, l'arbre de fitxers

2.4. Frontend

Un cop creat el projecte de `Django`, la base de dades i implementada la part de `backend`, ja vaig poder implementar el `frontend`. El `frontend` que crearé es pot dividir en dues parts:

- El `frontend` de la part d'administració. Aquest `frontend` és el que ens proporciona l'aplicació `grappelli`, per tant d'aquest frontend no hem tinc de preocupar de crear-lo.
- El `frontend` de la part d'usuari. Aquest és el frontend que he de crear jo.

Tot el `frontend` que crearé es trobarà en el directori `templates`.

2.4.1. Part d'Administració

Per el disseny de la part de l'administrador, utilitzant l'aplicació `grappelli`, junt amb el fitxer `models.py` explicat en l'apartat[2.2.6] i el fitxer `admin.py` [3] que explicaré a continuació, no m'he de preocupar de crear els `templates` ni els `views` en el `backend`, ja que l'aplicació `grappelli` ja se'n ocupa de crear tot l'entorn d'administració.
Les parts més importants del fitxer `admin.py` són les següents:

```
@admin.register(Maquina)
class MaquinaAdmin(admin.ModelAdmin):
    list_display = ('name',)
    list_filter = ('name',)
```

El `@admin.register(Maquina)` serveix perquè la taula `Maquina` aparegui en l'administrador, i poder interactuar amb ella.
La classe `MaquinaAdmin` serveix per dir-li a l'administrador el que vols que aparegui com a informació de cada màquina creada quan s'està veient la taula `Maquina`, amb `list_display`. En aquest cas només es mostra el nom de la màquina. I `list_filter`, serveix per definir els filtres que es podran utilitzar des de l'administració per veure unes màquines o unes altres. En aquest cas, només es pot filtrar per el nom de la màquina.

```
@admin.register(Fase)
class FaseAdmin(admin.ModelAdmin):
    list_display = ('name_phase', 'name_machine', 'num_ordre',)
    list_filter = ('name_phase', 'name_machine', 'num_ordre',)
```

El `@admin.register(Fase)` serveix perquè la taula `Fase` aparegui en l'administrador, i poder interactuar amb ella.
La classe `FaseAdmin`, té la mateixa funció que la classe `MaquinaAdmin`, però amb la taula `Fase`. La informació que es mostra de cada fase creada amb `list_display` és el nom de la fase, el nom de la màquina a la que pertany la fase i la posició que ocupa aquella fase dintre de la màquina. I amb el `list_filter`, defineixo que podré filtrar els resultats per el nom de la fase, el nom de la màquina o per la posició que ocupa dintre de la màquina.

```
@admin.register(Video)
class VideoAdmin(admin.ModelAdmin):
    list_display = ('name',)
    list_filter = ('name',)
```

El `@admin.register(Video)` serveix perquè la taula `Video` aparegui en l'administrador, i poder interactuar amb ella.
La classe `VideoAdmin` té la mateixa funció que la que la classe `MaquinaAdmin`, però amb la taula `Video`. La informació que es mostra de cada vídeo creat amb `list_display` és el nom del vídeo. I amb `list_filter`, defineixo que podré filtrar els resultats per el nom del vídeo.

```
@admin.register(Muntatge)
class MuntatgeAdmin(admin.ModelAdmin):
    list_display = ('id_maquina', 'id_fase', 'id_video', 'num_ordre',)
    list_filter = ('id_maquina', 'id_fase', 'id_video', 'num_ordre',)
```

El `@admin.register(Muntatge)` serveix perquè la taula `Muntatges` aparegui en l'administrador, i poder interactuar amb ella.
La classe `MuntatgeAdmin(admin.ModelAdmin)`, té la mateixa funció que la classe `MaquinaAdmin`, però amb la taula `Muntatge`. La informació que es mostra de cada muntatge creat amb `list_display` és el nom de la màquina, el nom de la fase, el nom del vídeo i la posició que ocupa aquell vídeo per aquella fase que es troba en aquella màquina. I amb el `list_filter`, defineixo que podré filtrar els resultats per el nom de la màquina, el nom de la fase, el nom del vídeo o la posició que ocupa el vídeo.

2.4.2. Part d'Usuaris

Per el disseny de la part d'usuari, primer de tot vaig fer el disseny de la pàgina mitjançant el diagrama de blocs. Aquests diagrama és el següent:

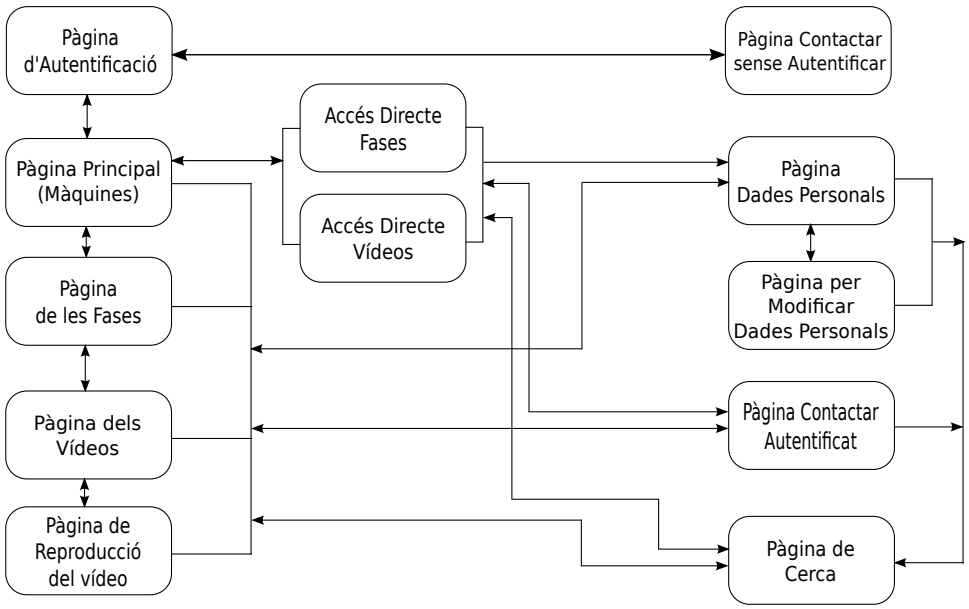

Figura 2.2.: Diagrama de blocs de la web d'usuari

Com es pot observar en el diagrama, la web es dividirà en deu `templates` diferents per els que podrà navegar un usuari:

- **Pàgina d'autentificació** És on s'hauran d'introduir l'usuari i la contrasenya per poder

accedir als tutorials.

- **Pàgina per contactar amb l'administrador de la web sense estar autentificat** Per utilitzar aquesta pàgina no cal estar autentificat, ja que precisament l'he creat per si hi ha algun problema al autentificar l'usuari.
- **Pàgina de les màquines** És la pàgina principal, és a dir la inicial quan ja s'ha identificat l'usuari. En aquesta pàgina és on es mostren les màquines de les quals es disposa tutorial i un navegador per poder accedir directament a una fase o un vídeo.
- **Pàgina de les fases** Un cop s'ha seleccionat una de les màquines, es mostren les fases que que té el muntatge de la màquina seleccionada.
- **Pàgina dels vídeos** Un cop s'ha seleccionat una de les fases, es mostren els vídeos que formen aquella fase.
- **Pàgina de Reproducció del Vídeo** És on es reprodueix el vídeo que s'ha seleccionat per la seva reproducció en la pàgina anterior.
- **Pàgina per contactar amb l'administrador quan l'usuari està autentificat** Serveix per els usuaris que es volen posar en contacte amb l'administrador per comentar-li algun dubte, problema o observació de la pàgina.
- **Pàgina de la cerca** Serveix per quan els usuaris fan una cerca, en aquesta pàgina se'ls hi mostri tots els resultats que coincideixen amb aquella cerca, ja siguin màquines, fases o vídeos.
- **Pàgina per visualitzar les dades personals** És on és poden veure les dades de l'usuari (nom, cognoms i email personal).
- **Pàgina per modificar les dades personals** Si alguna dada personal que es mostra no és correcta, l'usuari pot entrar en aquesta pàgina i modificar-la.

2.4.3. Templates

Un cop he explicat l'estructura de la web de la part d'usuari, explicaré les parts més importants de cada `template` per veure com està programat perquè presenti les dades dinàmicament, amb la incorporació dels tags de `Django`. El codi d'aquests fitxers està complert en la secció d'annexos.
Amb aquests tags s'utilitzen les variables i funcions que se li passen des de les views corresponents.
Els arxius de `frontend` que formen el projecte són:

1) base.html
2) index.html
3) contactar2.html
4) maquines.html
5) fases_maquina.html

6) minivideos.html
7) video.html
8) link_fases.html
9) link_videos.html
10) search.html
11) contactar.html
12) dades_personals.html
13) modify_data.html

2.4.3.1. base.html

El que fa el fitxer `base.html` [4] és crear la base que faré servir per crear els altres fitxers HTML, per així tenir l'esquelet de totes les pàgines web i totes les llibreries que es necessiten importar en el mateix lloc, i no haver-ho de repetir en cada fitxer HTML.
Les parts que formen el fitxer `base.html` són:

- {% load staticfiles %}: És una declaració que ens aporta `Django` que serveix per carregar tots els fitxers estàtics.
- `head`: És on s'importen totes les llibreries externes que es necessiten per a l'aplicació web. També s'hi defineix el que es mostrarà en el *title*, i s'importa el nostre fitxer `style.css`, per fer servir els estils propis que jo he creat.
- `<body>`: És on es defineixen i s'estructuren totes les parts de les que estan formades totes les pàgines web que he creat. Està formada per les següents parts:
  - `wrapper`: És la divisió principal, on anirà tot el contingut de la pàgina web.
  - `nav`: És el navegador. Si no s'ha iniciat sessió, només s'hi veurà el logo d'AUSA. Si s'ha iniciat sessió, s'hi veurà el logo d'AUSA, un link per anar a la pàgina principal, on es mostren totes les màquines de les quals hi ha tutorial, un link per veure les dades personals de l'usuari i la icona per tancar la sessió.
- {% block section %} {% endblock %}: És l'ordre de `Django` amb la qual se li diu el lloc on ha de posar les seccions que es defineixen en els templates següents.
- `footer`: És on és defineix el que està en la part inferior de la pàgina web, sota tota la informació que es vol mostrar.

2.4.3.2. index.html

El que fa el fitxer `index.html` [5] és demanar el nom d'usuari i la contrasenya per poder accedir als tutorials.
Les parts que formen el fitxer `index.html` són:

- {% extends 'include/base.html' %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definida.

- {% load staticfiles %}: És una declaració que aporta Django que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de Django amb la que es declara que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer base.html.
- formulari: És la divisió on està el formulari d'autentificació per poder accedir als tutorials.

La pàgina web que pot observar és la següent:

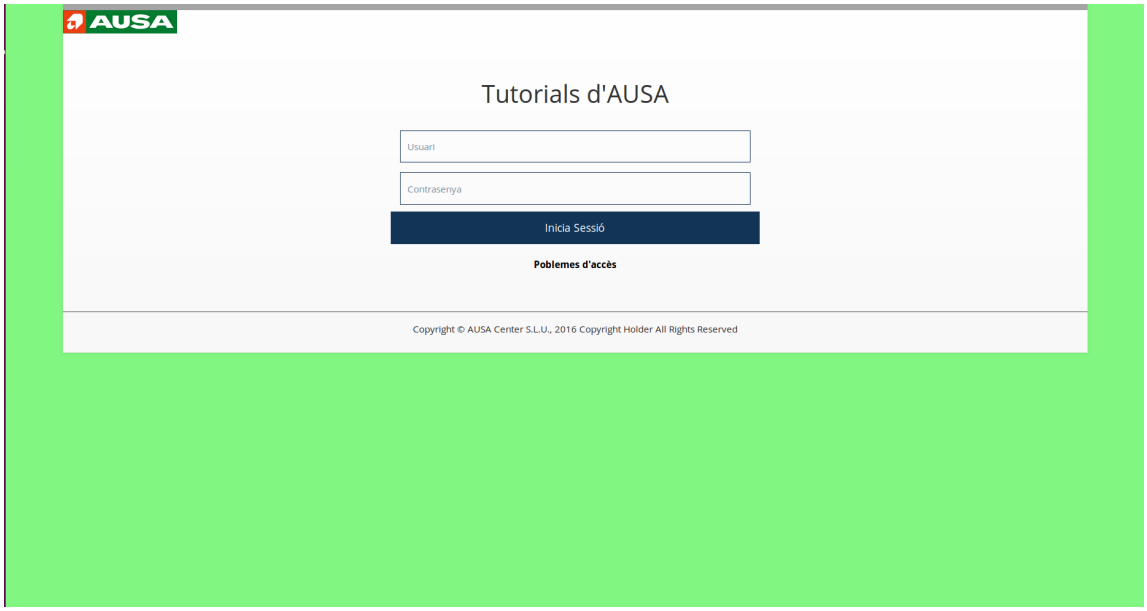

Figura 2.3.: Pàgina d'autentificació per accedir a la web

2.4.3.3. contactar2.html

El que fa el fitxer contactar2.html [10] és oferir un formulari de contacte per els usuaris que no puguin accedir a la pàgina web, perquè puguin comunicar-li a l'administrador que tenen problemes per accedir-hi.
Les parts que formen el fitxer contactar2.html són:

- {% extends 'include/base.html' %}: És l'ordre de Django amb la que s'importa el fitxer base.html, per fer servir l'estructura que hi ha definida.
- {% load staticfiles %}: És una declaració que aporta Django que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de Django amb la que es declara que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer base.html.

- {% for error in errors %}{% endfor %}: Si es produeix algun error al intentar enviar el email, aquest `for` el que fa es mostrar tots els errors que s'han produït en el `template`, perquè l'usuari sàpiga perquè no s'ha enviat.
- {% if send %}{% endif %}: Si el email s'ha enviat correctament, en el `template` apareix el missatge de que el correu s'ha enviat correctament.

La pàgina web que pot observar és la següent:

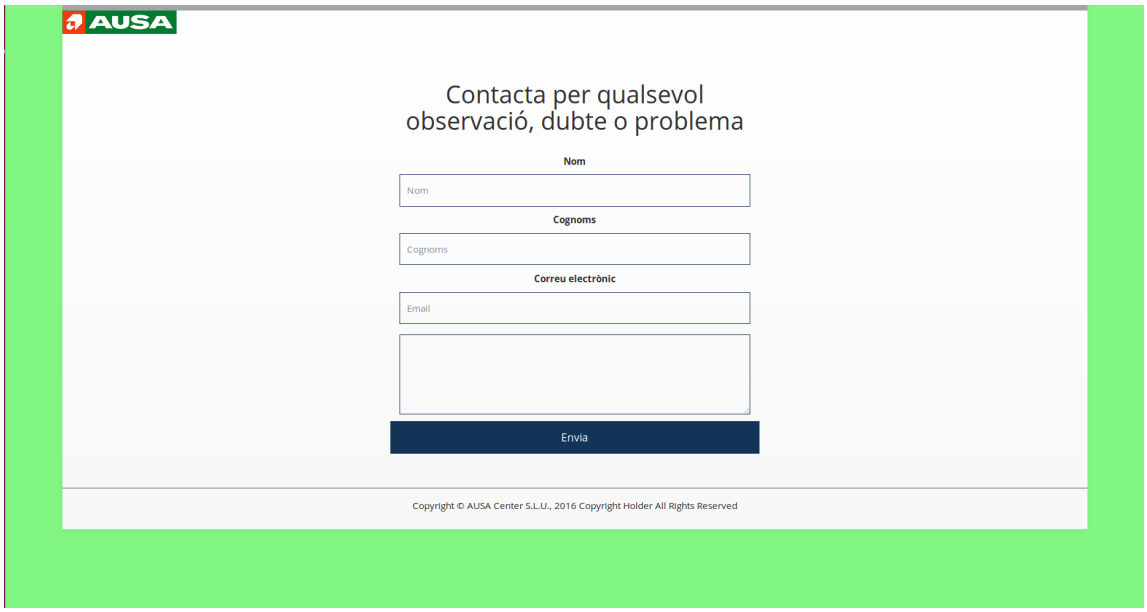

Figura 2.4.: Pàgina per contactar amb l'administrador sense autentificar

2.4.3.4. maquines.html

El que fa el fitxer `maquines.html` [6] és mostrar totes les màquines que hi ha en el tutorial, a més d'un navegador per poder accedir directament a:

1) Les fases
2) Els vídeos

Aquesta pàgina només es mostra si s'ha iniciat sessió. Les parts que formen el fitxer `maquines.html` són:

- {% extends 'include/base.html' %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definit.
- {% load staticfiles %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de `Django` amb la que se li diu que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.

- `main1`: És la divisió on es mostren totes les màquines de les que hi ha tutorial.
- `main2`: És la divisió on es mostra el navegador per poder accedir directament a les fases o vídeos. Aquesta divisió està formada per:
  - {% `url 'fases_list'` %}: Link per accedir a la pàgina `link_fases.html` [2.4.3.8], on es mostraran el nom de totes les màquines i les fases que formen cadascuna.
  - {% `url 'videos_list'` %}: Link per accedir a la pàgina `link_videos` [2.4.3.9], on es mostraran el nom de totes les màquines, les fases que formen cada màquina, i els vídeos que formen cada fase.

La pàgina web que es pot observar és la següent:

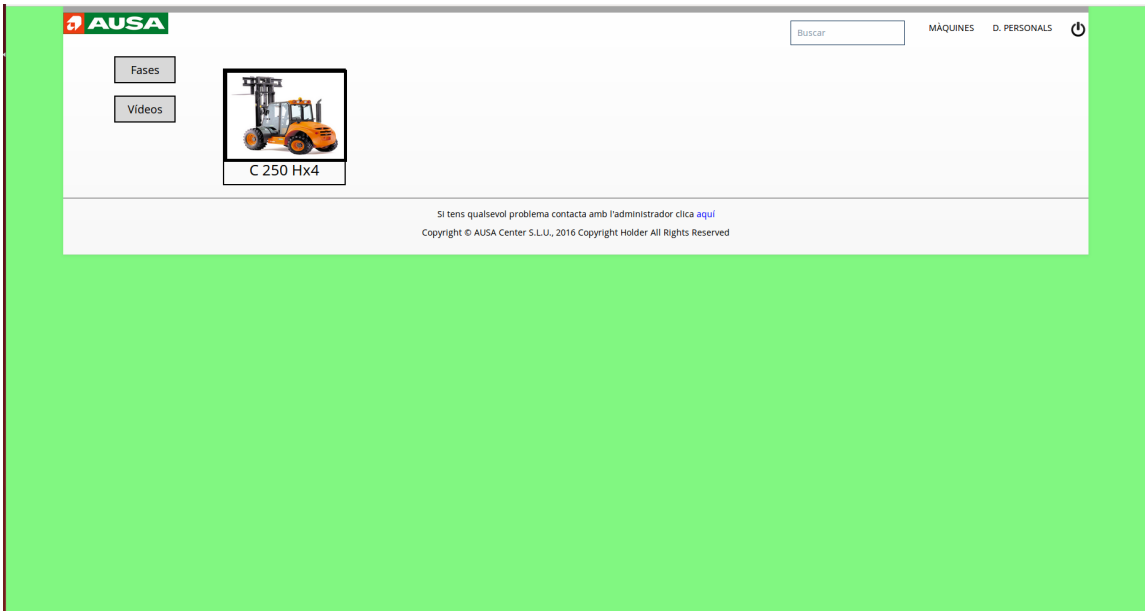

Figura 2.5.: Pàgina principal de la part d'usuari

2.4.3.5. fases_maquina.html

El que fa el fitxer `fases_maquina.html` [7] és mostrar les fases de muntatge que té la màquina que s'ha escollit, per triar els tutorials de quina fase es vol visualitzar. Aquesta pàgina només es podà veure si l'usuari s'ha autentificat.
Les parts que formen el fitxer `fases_maquina.html` són:

- {% `extends 'include/base.html'` %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definida.
- {% `load staticfiles` %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% `block section` %} {% `endblock` %}: És l'ordre de Django amb la que es diu que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.

- `main`: És la divisió on es mostren totes les fases de la màquina que hem escollit en la pàgina `maquines.html`

La pàgina web que podem observar és la següent:

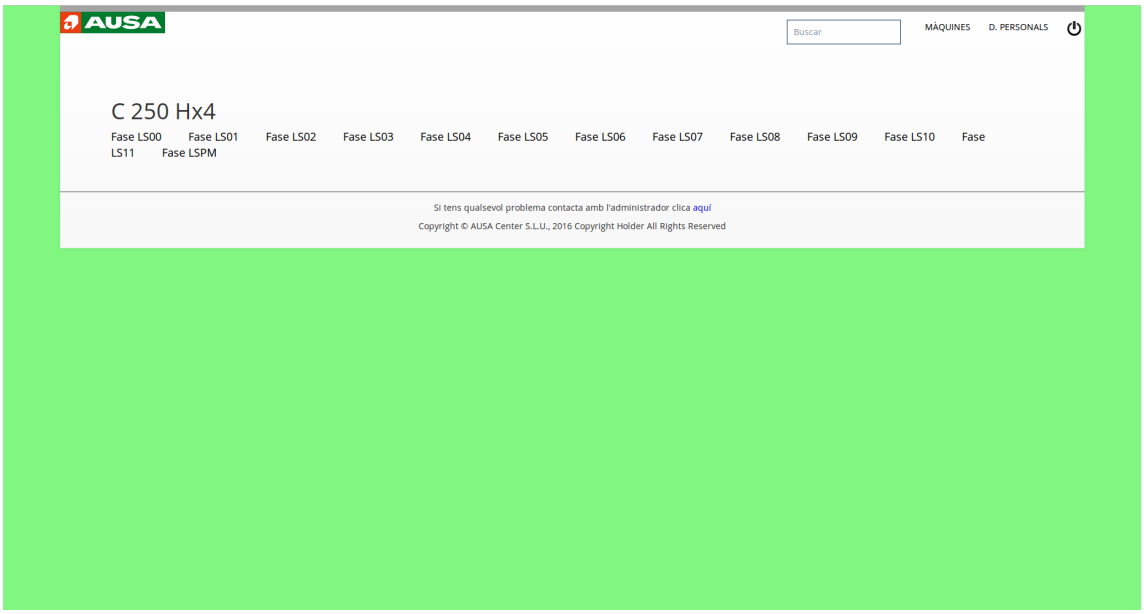

Figura 2.6.: Pàgina de les fases de la màquina

2.4.3.6. minivideos.html

El que fa el fitxer `minivideos.html` [8] és mostrar els mini-vídeos que formen la fase que s'ha escollit en la pàgina `fases_maquina.html`. Aquí només es mostraran els vídeos d'aquella fase i de la màquina escollida. Aquesta pàgina només s'hi pot accedir si s'està autentificat.
Les parts que formen el fitxer `minivideos.html` són les següents:

- {% `extends 'include/base.html'` %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definida.
- {% `load staticfiles` %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% `block section` %} {% `endblock` %}: És l'ordre de `Django` amb la que es diu que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.
- `gallery`: És la divisió on es mostren tots els mini-vídeos que formen part d'aquella fase i d'aquella màquina en concret.

La pàgina web que es pot observar és la següent:

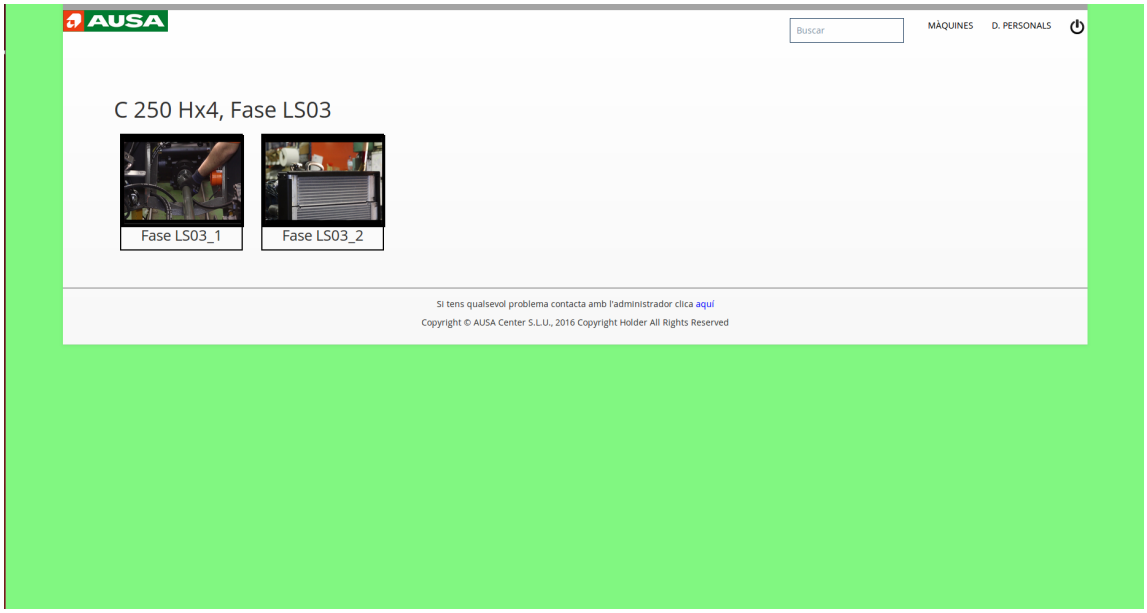

Figura 2.7.: Pàgina de les parts en que està dividida una fase

2.4.3.7. video.html

El que fa el fitxer `video.html` [9] és reproduir el vídeo que s'ha escollit en la pàgina `minivideos.html`. Al costat del vídeo que s'està reproduint, es poden veure els vídeos amb imatges petites que també formen part de la fase que està el vídeo en reproducció. Sota del vídeo que s'està reproduint, hi apareixen les instruccions o advertències que s'han de tenir en compte a l'hora de realitzar la tasca que es pot visualitzar en aquell vídeo. També es pot escollir si es vol veure a pantalla complerta i si vol fer la reproducció automàtica de tots els vídeos que formen part de la fase escollida anteriorment. Aquesta pàgina només s'hi pot accedir si s'està autentificat.
Les parts que formen el fitxer `video.html` són les següents:

- {% extends 'include/base.html' %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definit.
- {% load staticfiles %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de `Django` amb la que es diu que la secció que s'escriguin en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.
- `video`: És la divisió on es mostra el vídeo principal reproduint-se. També ens apareixen les opcions per el vídeo (pausar, pujar el volum i pantalla complerta).
- `minivideos`: És la divisió on es mostren les miniatures dels vídeos que formen part de la fase escollida, amb el que s'està reproduint inclòs, marcat en negre.

- `playlist`: És la divisió on he posat el *input* per escollir si es vol fer la reproducció automàtica de tots el vídeos que formen la fase o no.
- `script`: En els scripts, tenim les funcions en javascript per poder fer:
  1) Per fer la reproducció automàtica.
  2) Per saber si s'ha marcat el *input* de la reproducció automàtica per reproduir el següent vídeo automàticament.
  3) Perquè no es puguin descarregar els vídeos que s'estan visualitzant.

La pàgina web que es pot observar és la següent:

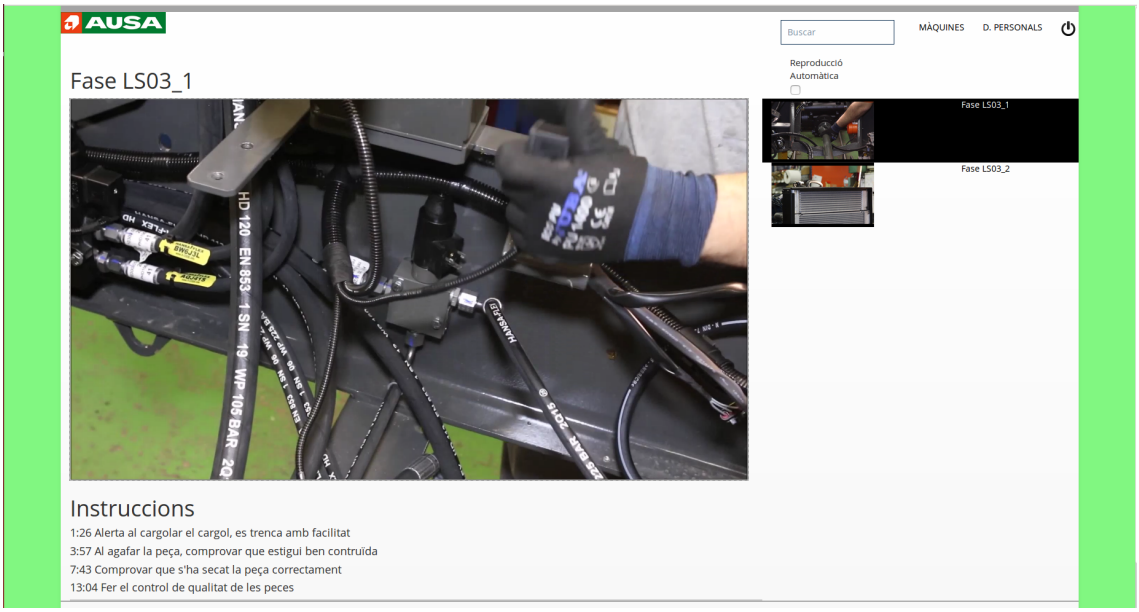

Figura 2.8.: Pàgina de reproducció del vídeo

2.4.3.8. link_fases.html

El que fa el fitxer `link_fases.html` [11] és mostrar totes les màquines, amb totes les fases que la formen. Aquesta pàgina només si pot accedir si s'està autentificat. Les parts que formen el fitxer `link_fases.html` són les següents:

- {% `extends 'include/base.html'` %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definit.
- {% `load staticfiles` %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% `block section` %} {% `endblock` %}: És l'ordre de `Django` amb la que es diu que la secció que s'escriguin en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.
- {% `if current_machines.count > 0` %} {% `for maquina in current_machines` %}: Si existeix alguna màquina, en el template es van mostrant totes les que hi ha creades.

- {% if current_fases.count > 0 %} {% for fase in current_fases %} {% if fase.name_machine_id == maquina.id %}: Condició que es mira si existeix alguna fase. Si aquesta condició es compleix, es recorren totes les fases que existeixen, i s'assignen a la màquina de la que formen part.

La pàgina web que pot observar és la següent:

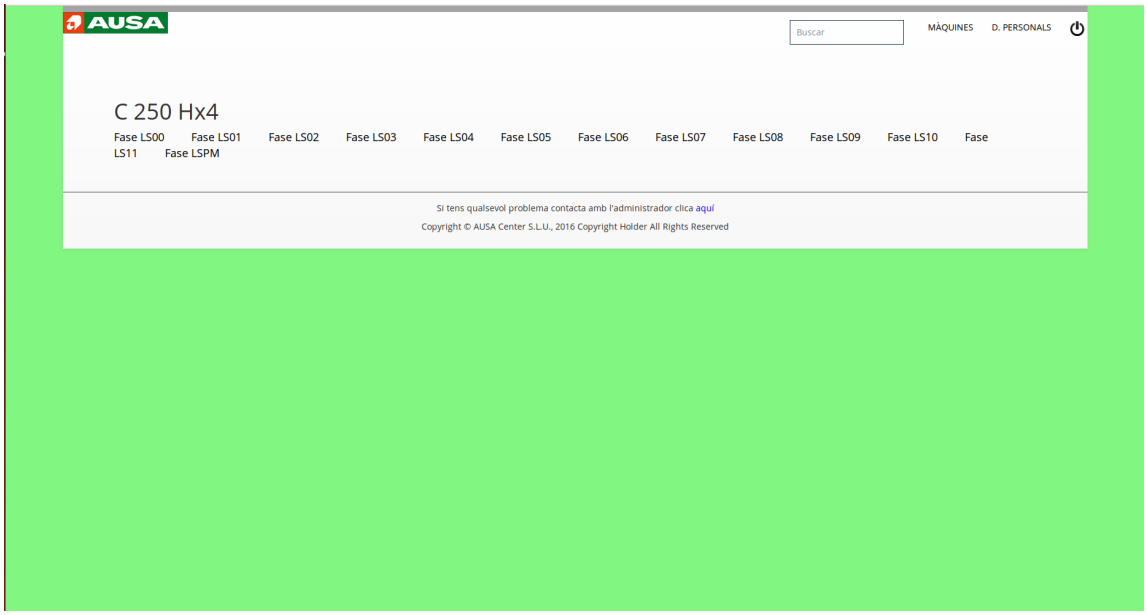

Figura 2.9.: Pàgina d'accés directe a les fases

2.4.3.9. link_videos.html

El que fa el fitxer `link_videos.html` [12] és mostrar totes les màquines amb les fases que formen a cadascuna, i dintre de cada fase, els vídeos que formen aquella fase. Les parts que formen el fitxer `link_videos.html` són les següents:

- {% extends 'include/base.html' %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definit.
- {% load staticfiles %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de `Django` amb la que es diu que la secció que s'escriguin en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.
- {% if current_machines.count > 0 %} {% for maquina in current_machines %}: Si existeix alguna màquina, en el template es van mostrant totes les que hi ha creades.
- {% if current_fases.count > 0 %} {% for fase in current_fases %} {% if fase.name_machine_id == maquina.id %}: Condició que es mira si existeix alguna fase. Si aquesta condició es compleix, es recorren totes les fases que existeixen, i s'assignen a la màquina de la que formen part.

- {% for muntatge in current_muntatges %} {% if muntatge.id_maquina_id == maquina.id %} {% if muntatge.id_fase_id == fase.id %} {% for video in current_videos %} {% if video.id == muntatge.id_video_id %}: Per cada vídeo afegit en una fase, s'assigna a la màquina i fase de la que forma part.

La pàgina web que pot observar és la següent:

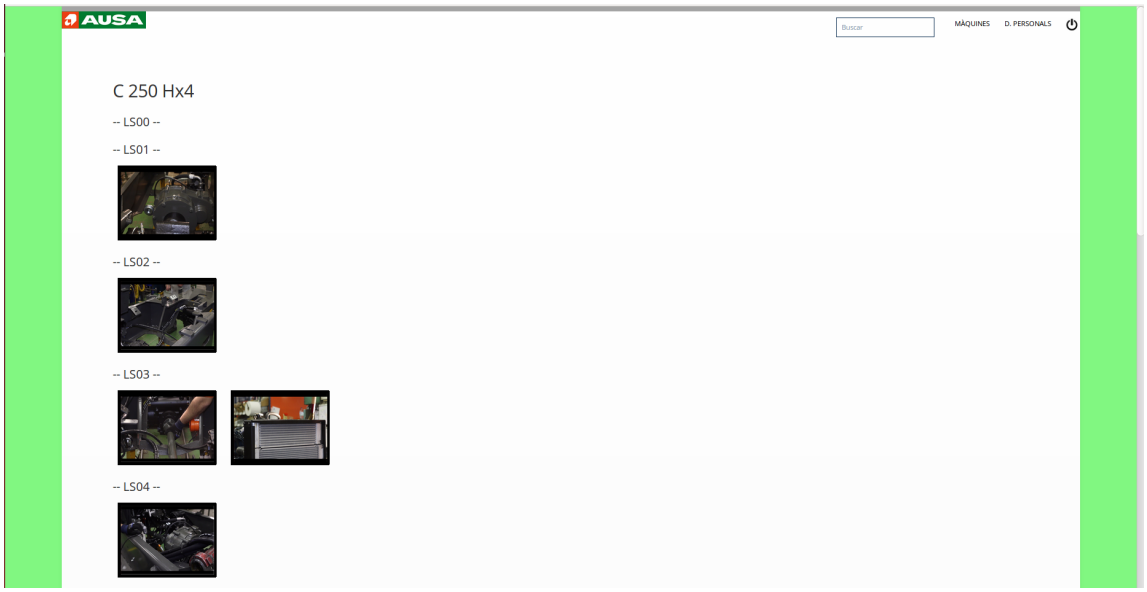

Figura 2.10.: Pàgina d'accés directe als vídeos

2.4.3.10. search.html

El que fa el fitxer `search.html` [13] és mostrar tots els resultats que coincideixen amb la cerca realitzada, ja siguin màquines, fases o vídeos. Si no hi ha cap resultat que coincideixi amb la cerca, es mostrarà el missatge de que no hi ha cap resultat per aquesta cerca. En el cercador es pot fer una cerca de una o dues paraules, separades amb una coma. Si es fa una cerca amb més paraules, en el `template` es mostra un missatge d'error informant a l'usuari que com a màxim es pot fer una cerca de dues paraules.
Aquesta cerca o cerques pot ser amb el nom complert de la màquina, fase o vídeo, o també amb una lletra o amb un tros del nom, i el cercador et mostrarà tots els resultats, siguin màquines, fases o vídeos que coincideixen amb la cerca realitzada per l'usuari.
Amb aquest fitxer no vaig poder fer servir el fitxer `base.html`, perquè al fer la cerca, no hem detectava que l'usuari estava autentificat, i per tant, no apareixia el menú dels usuaris. Per aquest motiu, vaig de tenir d'escriure des de zero aquest fitxer.
Les parts que formen el fitxer `search.html` són les següents:

- {% load staticfiles %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.

- {% if consultes == 1 %}: Variable que serveix per veure si l'usuari ha cercat més de dues paraules a la vegada. Si es així, es mostra el missatge d'error de que com a màxim

es poden cercar dues paraules a la vegada. Si es fa la cerca correctament, es mostra la cerca realitzada.

- `{% if m > 0 %}{% endif %}`: Condició que serveix per veure si la cerca realitzada ha obtingut algun resultat que coincideixi amb alguna màquina. Si és així, crea el títol de `Màquines`, i per cada màquina que coincideix, amb les comandes `{% for maquina in maquines %}{% endfor %}` creo una divisió on hi poso la màquina, amb el seu nom.
- `{% if f > 0 %}{% endif %}`: Condició que serveix per veure si la cerca realitzada ha obtingut algun resultat que coincideixi amb alguna fase. Si és així, crea el títol de `Fases`, i per cada fase que coincideix, amb les comandes `{% for element in ldf %}{% endfor %}` creo una divisió on hi poso la fase, amb el seu nom i la màquina a la que pertany.
- `{% if v > 0 %}{% endif %}`: Condició que serveix per veure si la cerca realitzada ha obtingut algun resultat que coincideixi amb algun vídeo. Si és així, crea el títol de `Vídeos`, i per cada vídeo que coincideix, amb les comandes `{% for element in ldv %}{% endfor %}` creo una divisió on hi poso el seu nom, la fase i la màquina a la que pertany.
  Si aquesta condició no es compleix, i les dues anteriors tampoc, es mostra en el `template` que no hi ha cap resultat per mostrar.

La pàgina web que podem observar és la següent:

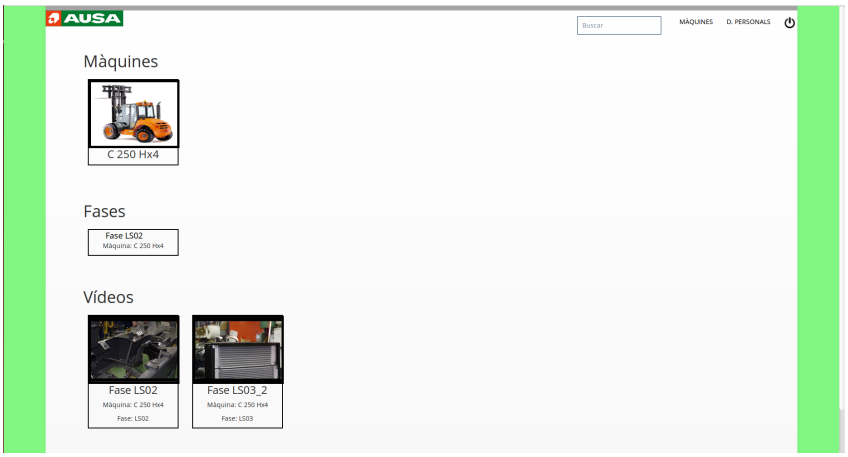

Figura 2.11.: Pàgina de cerca

2.4.3.11. contactar.html

El que fa el fitxer `contactar.html` [14] és oferir un formulari de contacte per els usuaris autentificats en la pàgina web, perquè puguin comunicar-li a l'administrador qualsevol problema, dubte o observació sobre la web.
Les parts que formen el fitxer `contactar.html` són:

- {% `extends 'include/base.html'` %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definida.
- {% `load staticfiles` %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% `block section` %} {% `endblock` %}: És l'ordre de Django amb la que es declara que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.
- {% `for error in errors` %}{% `endfor` %}: Si es produeix algun error al intentar enviar el email, aquest `for` el que fa es mostrar tots els errors que s'han produït en el `template`, perquè l'usuari sàpiga perquè no s'ha enviat.
- {% `if send` %}{% `endif` %}: Si el email s'ha enviat correctament, en el `template` apareix el missatge de que el correu s'ha enviat correctament.

La pàgina web que pot observar és la següent:

Figura 2.12.: Pàgina per contactar amb l'administrador autentificat

2.4.3.12. dades_personals.html

El que fa el fitxer `dades_personals.html` [15] és mostrar a l'usuari les seves dades personals:

- Nom
- Cognom
- email personal

Les parts que formen el fitxer `dades_personals.html` són:

- {% `extends 'include/base.html'` %}: És l'ordre de `Django` amb la que s'importa el fitxer `base.html`, per fer servir l'estructura que hi ha definida.
- {% `load staticfiles` %}: És una declaració que aporta `Django` que serveix per carregar tots els fitxers estàtics.
- {% `block section` %} {% `endblock` %}: És l'ordre de Django amb la que es declara que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer `base.html`.

La pàgina web que pot observar és la següent:

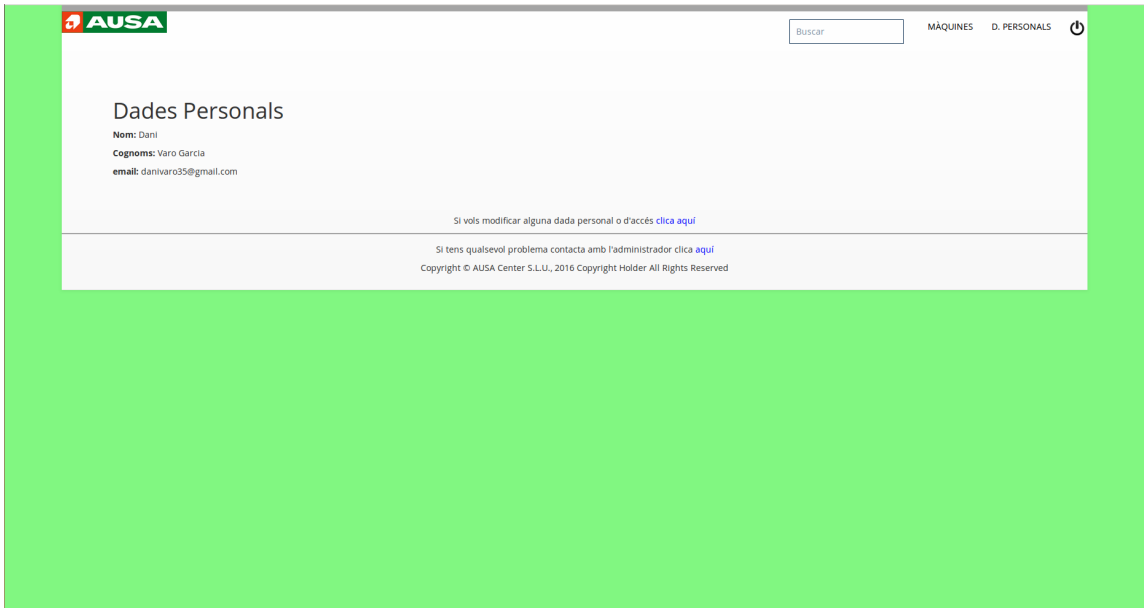

Figura 2.13.: Pàgina per visualitzar les dades personals de l'usuari

2.4.3.13. modify_data.html

El que fa el fitxer `modify_data.html` [16] és permetre a l'usuari modificar les seves dades personals, com són el nom, cognoms, email personal i la contrasenya per accedir a la pàgina web. Les parts que formen el fitxer `modify_data.html` són:

- {% extends 'include/base.html' %}: És l'ordre de Django amb la que s'importa el fitxer base.html, per fer servir l'estructura que hi ha definida.
- {% load staticfiles %}: És una declaració que aporta Django que serveix per carregar tots els fitxers estàtics.
- {% block section %} {% endblock %}: És l'ordre de Django amb la que es declara que la secció que s'escrigui en aquest fitxer es posi en el lloc que s'ha definit en el fitxer base.html.
- **{% if error %}{% endif %}**: Condició que serveix per si es produeix algun error durant la modificació d'alguna dada, s'avisa a l'usuari perquè el corregeixi.

La pàgina web que pot observar és la següent:

Figura 2.14.: Pàgina per modificar les dades personals de l'usuari

2.5. Servidor

Un cop acabat tant backend com el frontend i tenint acabat el projecte de Django, ja podia penjar el projecte en el servidor Apache. En l'empresa aquest pas de penjar el projecte al servidor no l'he fet, ja que se'n encarregava una altre persona. Però per demostrar com seria la configuració del servidor perquè el projecte funcionés correctament i fos accessible per la web, faré un servidor en el meu ordinador.
Primer de tot vaig tenir d'instal·lar Apache en l'ordinador amb la comanda:

```
$ apt-get install apache2
```

També vaig tenir d'instal·lar el mòdul wsgi, perquè Apache pugui executar codis de python. Aquest mòdul s'instal·la amb les comandes:

```
$ apt-get install python-setuptools
$ apt-get install libapache2-mod-wsgi
```

Un cop feta totes les instal·lacions, en el directori `/var/www/html/` copio el projecte de `Django`. Un cop està el projecte en aquest directori, s'ha d'anar al directori `/etc/apache2/sites-avaible/`, i crear el fitxer `ausassl.conf` [22]. Aquest fitxer serveix per fer la configuració del servidor utilitzant el servei `SSL`. El servei `SSL` serveix per proporcionar comunicacions segures per la xarxa.
Abans de crear el fitxer `ausassl.conf`, s'ha d'habilitar el suport `SSL` que incorpora `Apache`, executant la següent comanda:

```
a2enmod ssl
```

Un cop activat el servei `ssl`, s'ha de crear el directori `/etc/apache2/ssl`, que serà on guardaré la clau i el certificat que crearé. Per crear aquests dos fitxers, s'ha d'executar la comanda:

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout /etc/apache2/ssl/ausa.key -out
    /etc/apache2/ssl/ausa.crt.
```

Quan s'executa la comanda anterior, em demana que empleni les següents dades:

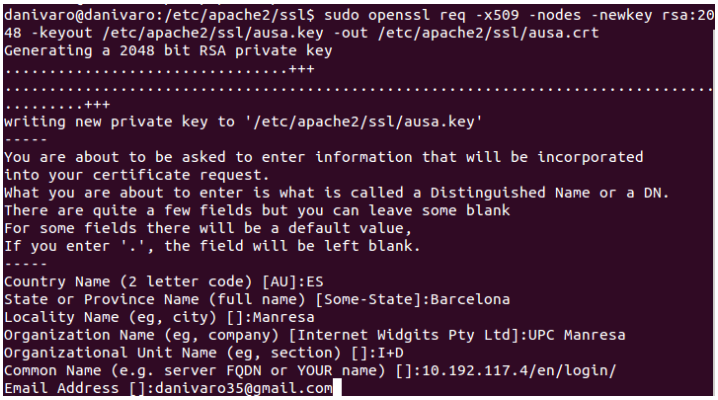

Figura 2.15.: Creació de la clau i el certificat

Un cop ja tinc disponibles la clau i el certificat, ja puc crear el fitxer `ausassl.conf`. Les parts més importants d'aquest fitxer són:

```
AliasMatch ^/([^/]*\.css) /var/www/html/tutorials/wsgi/static/css/$1
Alias /static/ /var/www/html/tutorials/wsgi/static/
Alias /media/ /var/www/html/tutorials/wsgi/media/
```

Aquestes comandes serveixen per informar a `Apache` on es troba el fitxer d'estàtics per els `templates`, els fitxers estàtics i els fitxers dinàmics respectivament.

```
WSGIScriptAlias / /var/www/html/tutorials/tutorials/wsgi.py
```

Aquesta comanda serveix per informar a `Apache` on està el script `wsgi` perquè es pugui executar el projecte de `Django`.

```
SSLEngine on
```

```
SSLProtocol all
SSLCertificateFile /etc/apache2/ssl/ausa.crt
SSLCertificateKeyFile /etc/apache2/ssl/ausa.key
```

Aquestes comandes serveixen perquè es faci servir el servei `SSL`, i també per informar a `Apache` on es troben la clau i el certificat per les connexions `SSL`. Un cop configurat el fitxer `ausassl.conf`, s'han d'executar les següents comandes:

```
a2ensite ausassl.conf
service apache2 restart
```

La primera comanda serveix per activar la configuració que hi ha feta en el fitxer, i la segona per reiniciar `Apache` i que la configuració que hi ha feta en el fitxer s'executi.
Després, s'ha d'obrir el fitxer `apache2.conf` i afegir al final de l'arxiu les següents línies:

```
WSGIPythonPath /var/www/html/tutorials
ServerName localhost
```

La primera línia serveix per dir-li a `Apache` on ha d'anar a buscar els fitxers `WSGI`, i la segona es perquè sàpiga que el nom del servidor és `localhost`.
Després d'afegir les comandes anteriors, s'han de donar permisos al projecte de Django perquè es pugui interactuar i modificar les dades des de `Apache`. Aquests permisos se li donen executant la comanda:

```
chown www-data:www-data -R /var/www/html/tutorials
```

Un cop fet això, ja es pot accedir a la pàgina web com qualsevol pàgina, però en aquest cas, com només és per demostrar com seria la configuració, només es pot veure la web si estem en la mateixa xarxa local.

El primer cop que algun usuari es connecti a la web veurà la següent pantalla:

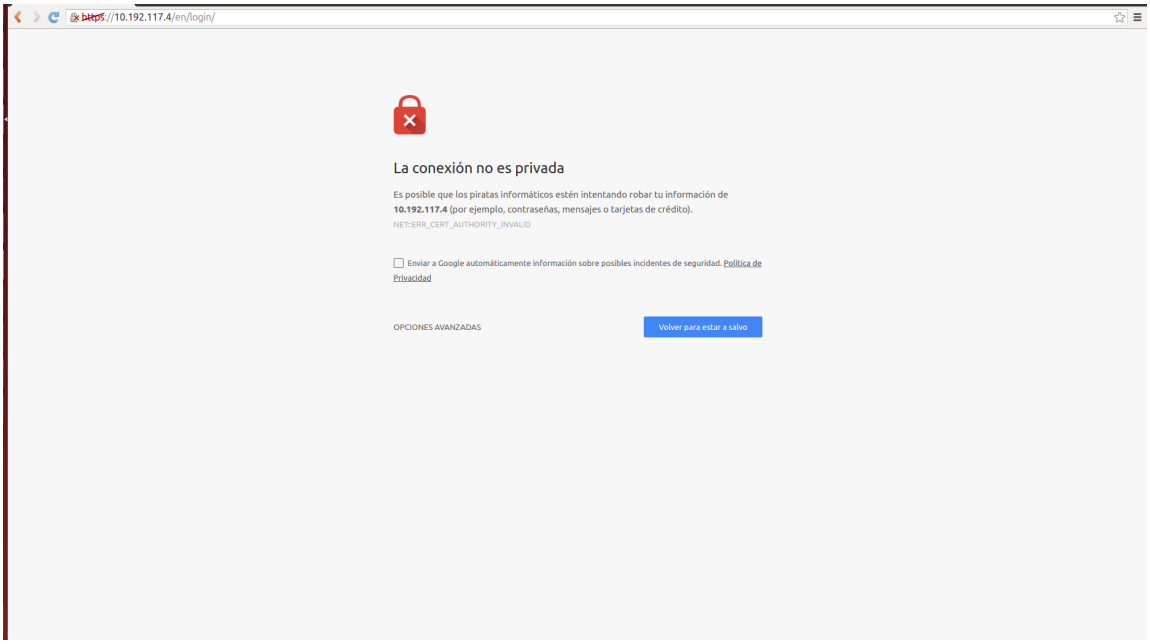

Figura 2.16.: Connexió no privada

Això es normal, ja que com el certificat l'he creat jo, i com no l'ha certificat cap autoritat certificadora, surt aquest avís. El que s'ha de fer Opciones Avanzadas i clicar Acceder a 10.192.117.4 (sitio no seguro). Al fer això, l'usuari ja pot accedir a la web fent servir el servei SSL sense cap problema.

3. Manuals de la Pàgina Web

Per interactuar correctament amb la pàgina web, he desenvolupat dos manuals d'usuari, un per la part d'**administració** i un altre per la part d'**usuari**. En cada un d'ells, explicaré pas a pas que és pot fer en la web.

3.1. Manual per la Part d'Administració

Per afegir, modificar o eliminar el contingut de la pàgina web dels usuaris, es té d'utilitzar la part d'administració. Des de aquí, podrem gestionar-ho tot, des de els usuaris i les seves dades, fins a les màquines, les fases de cada una, els vídeos que van dintre de cada fase, i l'ordre en que s'han de presentar tant les fases com els vídeos.
A continuació, explicaré tot el que es pot fer des de l'administració, explicant pas a pas com fer cada tasca.
Per dur a terme qualsevol de les tasques que explicaré, prèviament ens tenim d'autentificar en la següent pàgina, posant *usuari* i *contrasenya*:

Figura 3.1.: Autentificació de l'administració

S'ha de tenir en compte que l'usuari i contrasenya per accedir aquesta part de la web no és el mateix que per accedir a la part d'usuari, ja que l'usuari i contrasenya dels treballadors no servirà per entrar en la part d'administració.
És recomanable que el treballador o treballadors que s'encarreguin de la part d'administració, si és necessari, tingui dues comptes, una per la part d'administració, i l'altre per la part d'usuari.

3.1.1. Afegir Usuari

Per afegir un nou usuari, el que s'ha de fer és, quan ens trobem en la pàgina principal d'administració, clicar *Users*:

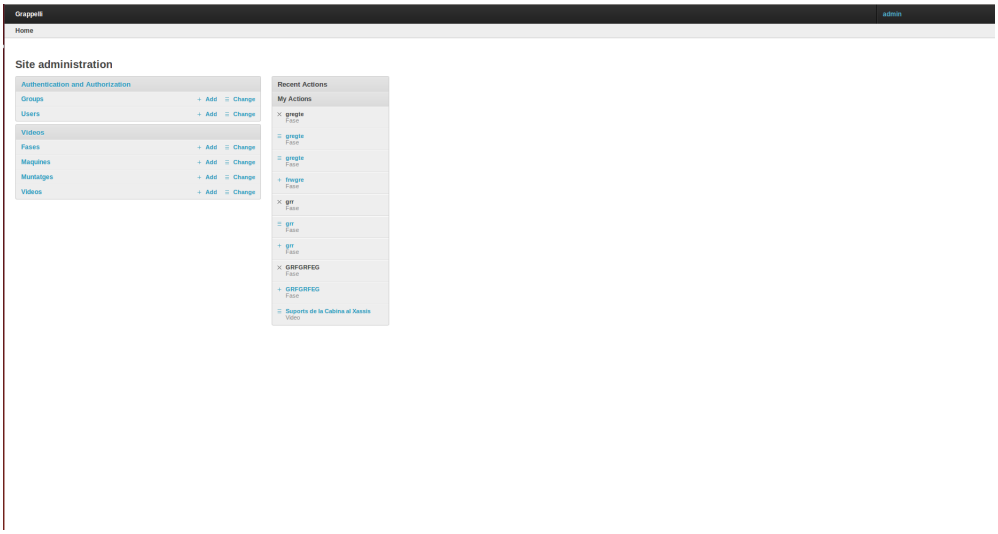

Figura 3.2.: Pàgina principal de l'administració

Quan s'hagi clicat en *Users*, se'ns redirigirà a la pàgina següent:

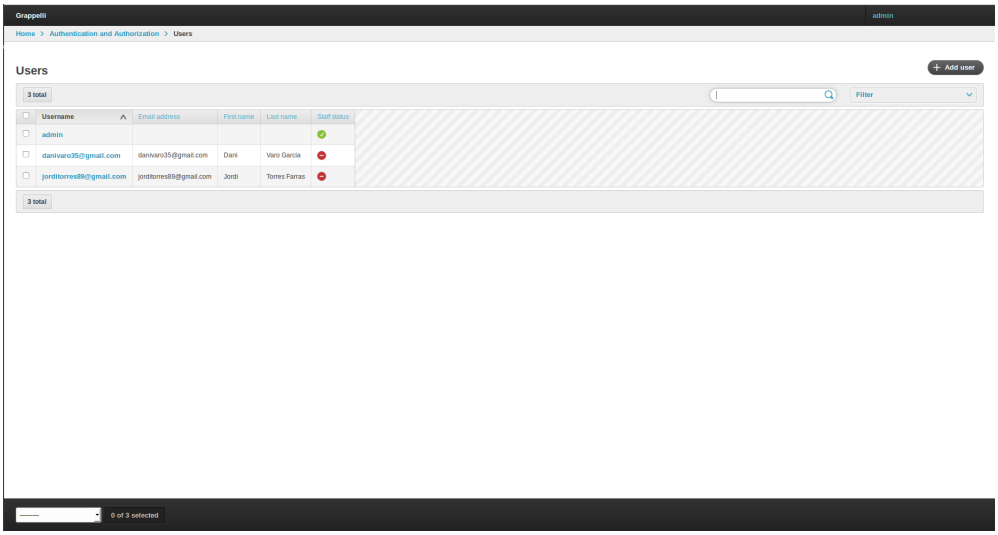

Figura 3.3.: Pàgina de gestió d'usuaris

En aquesta pàgina, podrem observar els usuaris que tenim creats fins aquest moment (si no s'ha creat encara cap, només veurem el compte de *admin*).
De tots els usuaris creats, se'ns mostraran les següents dades:

Username És el nom d'usuari que és tindrà de posar per poder accedir a la plataforma web. Menys la del administrador principal (admin) o els altres administradors, els altres noms d'usuaris han de ser emails.

Email address És el email personal del treballador. Aquest email pot coincidir amb el *Username*, però no tenen perquè ser el mateix. Si l'empresa proporciona un email, el username pot ser el de l'empresa, i en aquest apartat guardar l'altre email del que disposa l'usuari.

First name És on veiem el nom real de l'usuari que farà servir aquell compte.

Last name Són els cognoms de l'usuari.

Staff status És on podem veure de manera ràpida dels permisos que disposa cada usuari.

- Si és un icona verd, significa que l'usuari té permisos d'administrador, per tant, podrà accedir aquesta part de la web.
- Si és un icona vermell, significa que l'usuari no té permisos d'administrador, per tant, no podrà accedir aquesta part de la web.

En aquesta pàgina, també és pot buscar l'usuari escrivint seu nom d'usuari, email, nom o cognom, i se'ns mostrarà només aquell usuari, o els usuaris que coincideixin amb aquella cerca. També es pot filtrar el que volem buscar amb el *Filter*. Es pot filtrar pel *Staff status*. Se li pot dir que els mostri tots els status, o només els que ens interessen en aquell moment.
Per afegir un nou usuari, s'ha de clicar el botó *Add user*. Un cop clicat, ens trobarem en la següent pàgina:

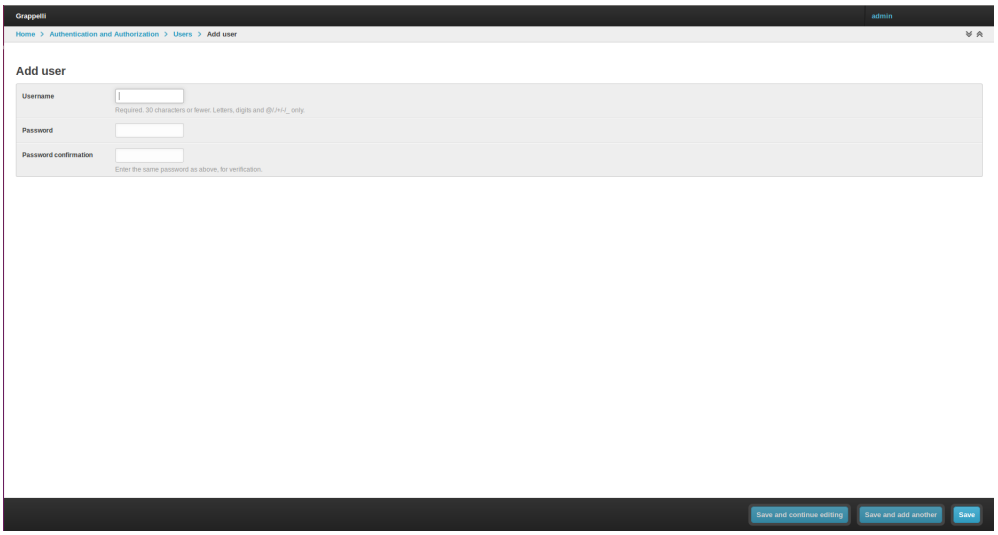

Figura 3.4.: Primera pàgina per afegir un usuari

En aquesta pàgina haurem d'afegir les següents dades de l'usuari:

Username El nom d'usuari que tindrà. Ha de ser un email.

Password Escriure la contrasenya que tindrà aquell usuari.

Password confirmation S'ha de tornar a escriure la contrasenya per confirmar que s'ha escrit correctament.

Un cop hem omplert tots els camps, hem de clicar el botó *Save*.
Quan cliquem el botó el *Save*, se'ns redirigirà cap a la pàgina següent:
En aquesta pàgina, s'ha d'omplir la secció *Personal info*. Haurem d'afegir les següents dades:

First name Escriurem el nom real de l'usuari.

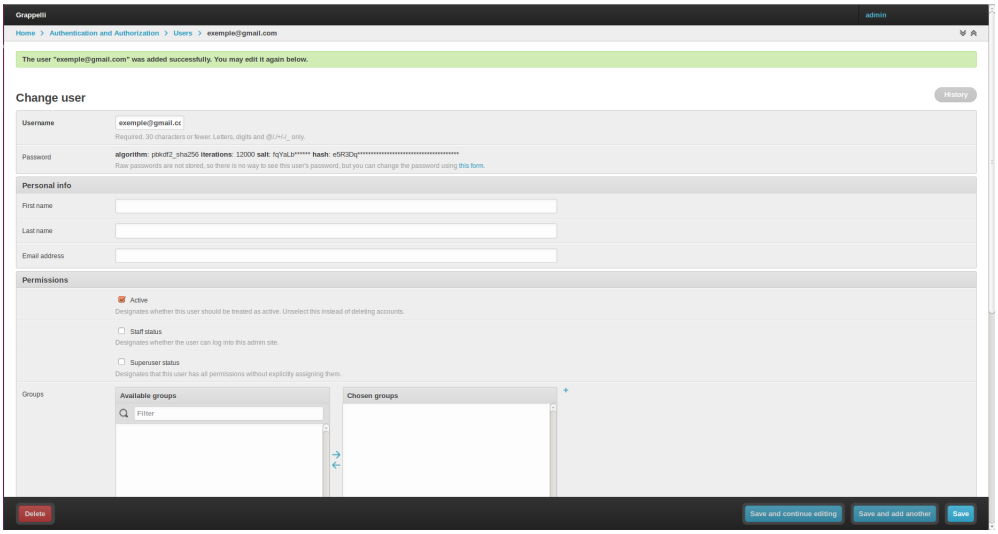

Figura 3.5.: Segona pàgina per afegir un usuari

Last name Escriurem els cognoms de l'usuari.

Email address Escriurem el email personal de l'usuari.

La secció *Permissions*, en aquest moment no toquem res. Per afegir permisos a l'usuari ho explicaré en el següent apartat.
Un cop s'hagi omplert la secció *Personal info*, cliquem el botó *Save*. Quan cliquem el botó, se'ns redirigirà cap a la pàgina on és veuen tots els usuaris, i l'usuari s'haurà creat correctament i ja el veurem en aquesta llista.

3.1.1.1. Afegir, Modificar o Treure Permisos a un Usuari

Quan creem un usuari seguint el procés que he descrit en l'apartat anterior, el creem amb els permisos bàsics. Per tant, aquest usuari només podrà accedir a la part d'usuari, però no podrà accedir a la part d'administració.
Aquests permisos són els correctes per la majoria d'usuaris, ja que només volem que puguin veure el que hi ha en la part d'usuari. Però si volem crear un usuari per poder accedir a la part d'administració, el que hem de fer és el següent.
Primer de tot, en la pàgina principal [3.2], s'ha de clicar el *Users*. Un cop s'hagi clicat *Users*, ens trobarem en la pàgina de gestió d'usuaris [3.3]. En aquesta pàgina, s'ha de clicar l'usuari que vulguem afegir, modificar o treure-li permisos.
Un cop s'hagi clicat l'usuari, ens trobarem en la pàgina següent:

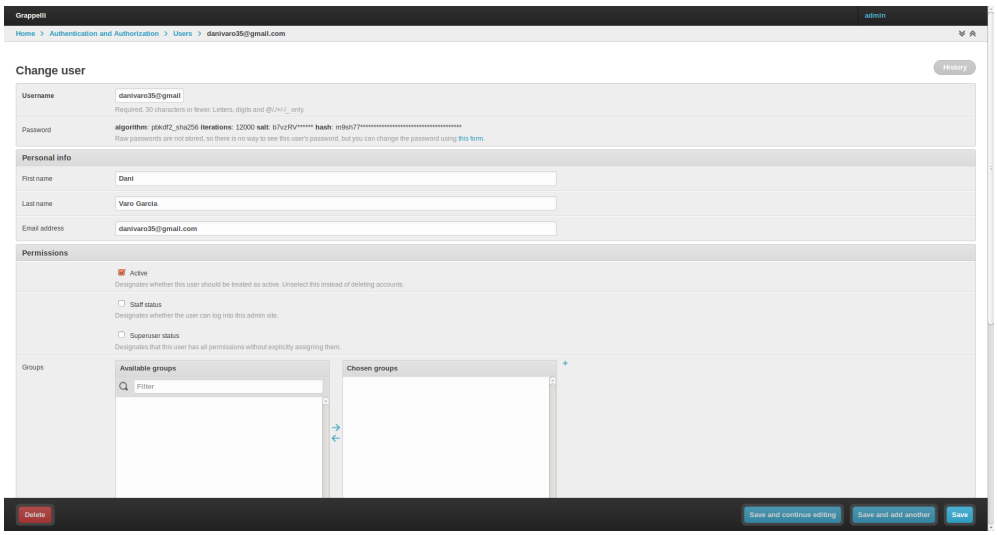

Figura 3.6.: Pàgina per afegir, modificar o treure permisos

En aquesta pàgina, veiem que l'Apartat de *Personal info* està omplerta. Aquesta part no l'hem de tocar.
El que s'ha de modificar és en l'apartat *Permissions*. D'aquest apartat, l'únic que hem de modificar, depenent dels permisos que li vulguem afegir a l'usuari és el següent:

Active Està marcat per defecte. Aquest permís és el que té un usuari estàndard per defecte. Per tant, aquest permís sempre ha d'estar marcat.

Staff status No està marcat per defecte. Marcarem aquesta opció si volem que aquell usuari tingui permís per accedir en aquesta part d'administració. Tot i així, encara hi ha algun permís que no tindrà, s'han de anar definint un a un.

Superuser status Amb aquest permís, l'usuari podrà fer tot del que disposa la part d'administració. Tindrà tots els permisos, i no caldrà anar-los definint un a un.

Per modificar el que es pot veure en la part d'usuari i poder interactuar sense cap problema en la part d'administració, recomano marcar com a mínim la primera i tercera opció.

3.1.2. Modificar Usuari

Per modificar les dades d'un usuari, en la pàgina principal d'administració [3.2], clicar *Users*. Un cop ens s'hagi clicat *Users*, ens trobarem la pàgina de gestió d'usuaris [3.3], on haurem de clicar l'usuari del que es vulgui modificar les seves dades.
Quan haguem clicat l'usuari del que es vulgui modificar les dades, ens trobarem en la pàgina per afegir, modificar o treure permisos [3.6]. Un cop ens trobem en aquesta pàgina, les dades que podem modificar de l'usuari són:

- Username
- First name
- Last name

- email address

Un cop haguem modificat les dades que ens interessen, cliquem el botó *Save*, i serem redirigits a la pàgina de gestió dels usuaris.

3.1.3. Eliminar Usuari

Per eliminar un usuari, en la pàgina principal d'administració [3.2], s'ha de clicar *Users*.
Un cop ens s'hagi clicat *Users*, ens trobarem la pàgina de gestió d'usuaris [3.3]. Quan ens trobem en aquesta pàgina, s'ha de seleccionar tots els usuaris que es vulguin eliminar, clicant el selector que hi ha a l'esquerra de l'username. Si els volem eliminar tots, es clica el primer selector, i seleccionem tots els usuaris.
Un cop s'hagin clicat els usuaris que es vulguin eliminar, anem a la part inferior esquerra de la pàgina. Cliquem la fletxa de la finestreta, i cliquem sobre *Delete selected users*.
Quan ho cliquem, serem redirigits automàticament a la següent pàgina, on es demanarà que confirmen l'acció que s'està a punt d'executar:

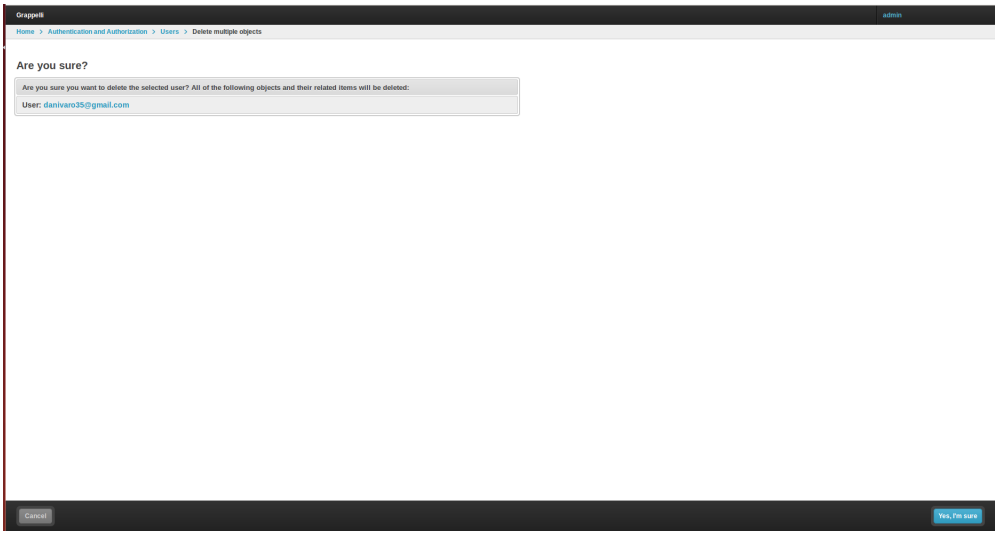

Figura 3.7.: Pàgina de confirmació d'eliminar un usuari

Cliquem el botó *Yes, I'm sure*, i l'usuari serà eliminat i se'ns redirigirà a la pàgina de gestió d'usuaris, amb un missatge en verd confirmant-nos que l'usuari s'ha esborrat correctament.

3.1.4. Afegir Màquina

Per afegir una nova màquina, el que s'ha de fer és, quan ens trobem en la pàgina principal d'administració [3.2], clicar *Maquines*.

Quan s'hagi clicat en *Maquines*, se'ns redirigirà a la pàgina següent:

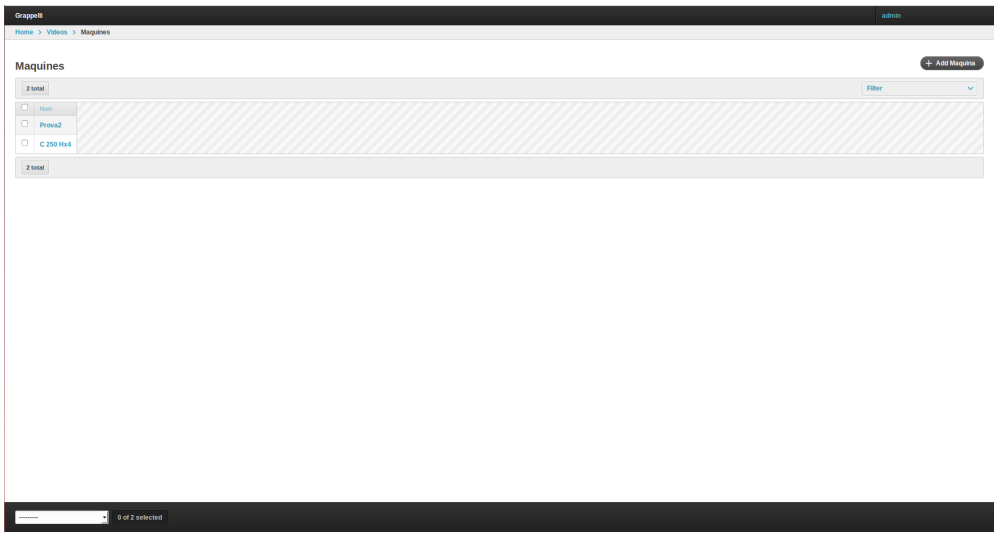

Figura 3.8.: Pàgina de gestió de les màquines

En aquesta pàgina, podrem observar les màquines que tenim creades fins aquest moment (si no s'ha creat encara cap, no se'ns mostrarà cap).
De totes les màquines creades, se'ns mostraran les següents dades:

Nom El nom que té cada màquina.

En aquesta pàgina, també podem buscar una màquina en concret, clicant en *Filter* i escollint el nom de la màquina que es vulgui visualitzar en aquell moment.
Per afegir una nova màquina, s'ha de clicar el botó *Add Maquina*. Un cop clicat, ens trobarem en la següent pàgina:

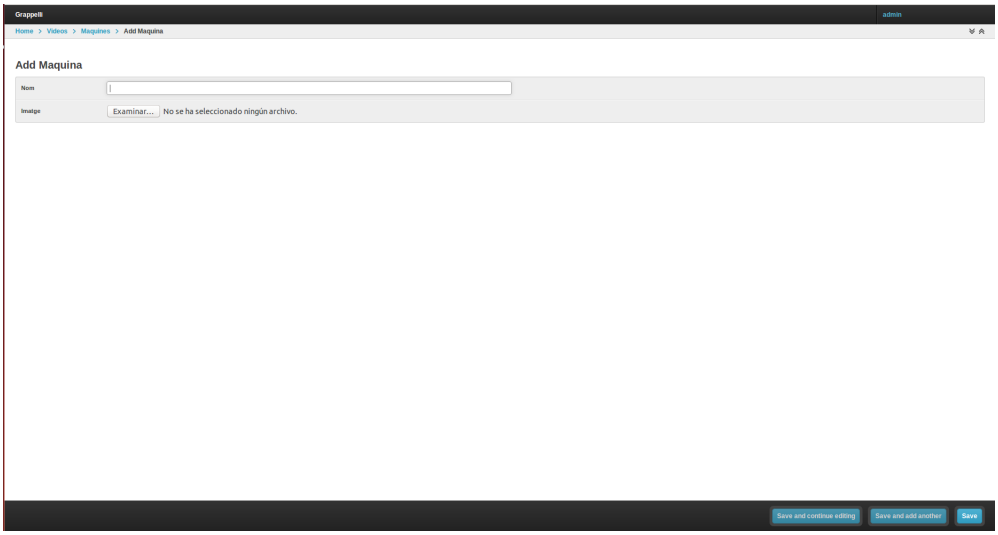

Figura 3.9.: Pàgina per afegir una màquina

En aquesta pàgina haurem d'afegir les següents dades de la màquina:

Nom El nom que tindrà la màquina.

Imatge Pujar una imatge de la màquina un cop finalitzada per mostrar-la en la part web de l'usuari. En aquest camp és important no posar en el nom de la imatge cap accent ni cap símbol semblant, sinó apareixerà un error.

Un cop s'han afegit totes les dades, cliquem el botó *Save*, i serem redirigits a la pàgina de gestió de les màquines, amb el missatge que la màquina s'ha afegit correctament. Aquesta ja apareixerà en la llista de màquines.

3.1.5. Modificar Màquina

Per modificar les dades d'una màquina, el que s'ha de fer és, des de la pàgina principal d'administració [3.2], clicar *Maquines*. Un cop s'hagi clicat en *Maquines*, se'ns redirigirà a la pàgina de gestió de les màquines [3.8].
Un cop en aquesta pàgina, hem de clicar la màquina de la que volem modificar les seves dades. Quan la cliquem se'ns redirigirà a la següent pàgina:

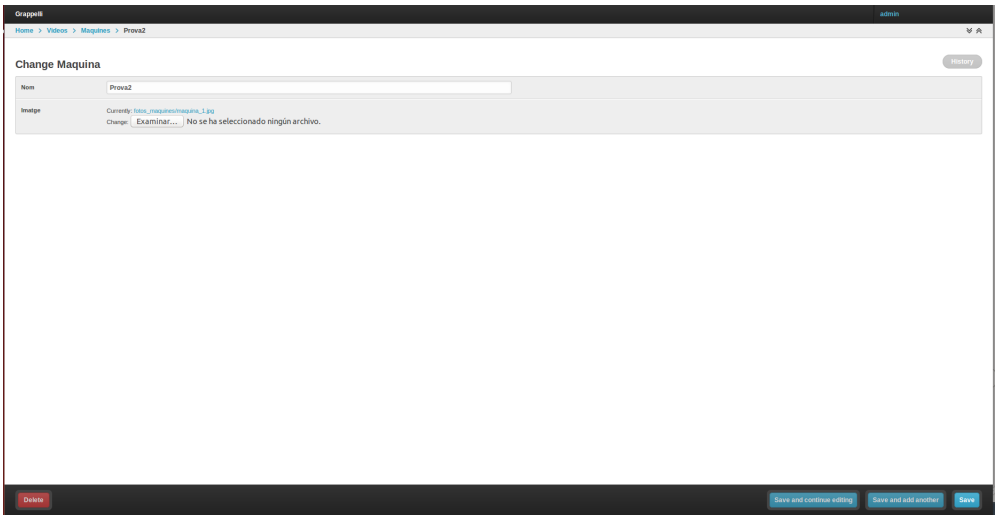

Figura 3.10.: Pàgina per modificar una màquina

Un cop s'hagin modificat les dades, cliquem el botó *Save*, i se'ns redirigirà automàticament a la pàgina de gestió de les màquines, on hi apareixerà un missatge conforme s'han modificat les dades de la màquina correctament.

3.1.6. Eliminar Màquina

Per eliminar una màquina, des de la pàgina principal d'administració [3.2], el que s'ha de fer és clicar *Maquines*. Al clicar *Màquines*, se'ns redirigirà a la pàgina de gestió de les màquines [3.8].
Un cop en aquesta pàgina, s'ha de seleccionar les màquines que es vulguin eliminar, clicant el selector que hi ha a l'esquerra del nom de la màquina. Si les volem seleccionar totes, cliquem el primer selector.
Un cop seleccionades les màquines que es volen eliminar, s'ha de dirigir a la part inferior

esquerra de la pàgina, i clicar la fletxa de la finestreta, i clicar *Delete selected Maquines*.
Quan s'hagi clicat, se'ns redirigirà automàticament a la següent pàgina, on es demanarà que es confirmi l'acció que s'està a punt d'executar:

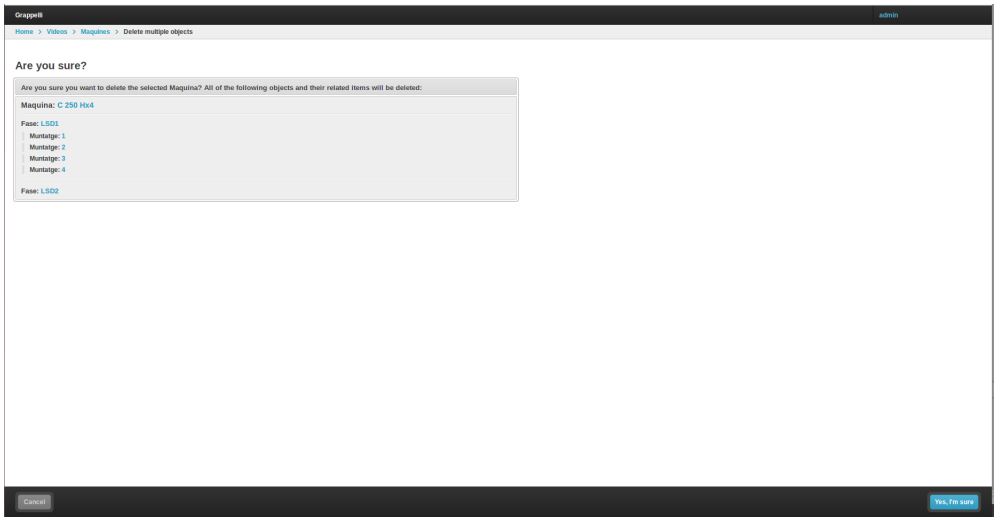

Figura 3.11.: Pàgina de confirmació d'eliminar una màquina

Cliquem el botó *Yes, I'm sure*, i la màquina o màquines seran eliminades. També s'eliminaran totes les fases i muntatges que estaven relacionats amb aquella màquina. Se'ns redirigirà a la pàgina de gestió de les màquines, amb un missatge en verd confirmant-nos que la màquina o màquines s'han eliminat correctament.

3.1.7. Afegir Fase

Per afegir una nova fase, el que s'ha de fer és, quan ens trobem en la pàgina principal d'administració [3.2], clicar *Fases*.
Quan s'hagi clicat en *Fases*, se'ns redirigirà a la pàgina següent:

En aquesta pàgina, podem observar les fases que hi ha creades fins aquell moment (si encara no s'ha creat cap, no se'ns mostrarà res). De totes les fases creades, se'ns mostraran les següents dades:

Nom de la fase El nom que s'ha definit per aquella fase.

Maquina on es troba Màquina la que pertany aquella fase.

Posicio de la fase dintre la maquina Nombre per definir l'ordre en que s'ha de presentar cada fase dintre de la màquina.

En aquesta pàgina també es pot fer servir el filtre per trobar les dades que ens interessin en aquell moment. Clicant el botó *Filter*, podem filtrar el que volem veure per els següents temes:

- Nom de la fase.
- Màquina on es troba la fase.
- Posició del la fase dintre de la màquina.

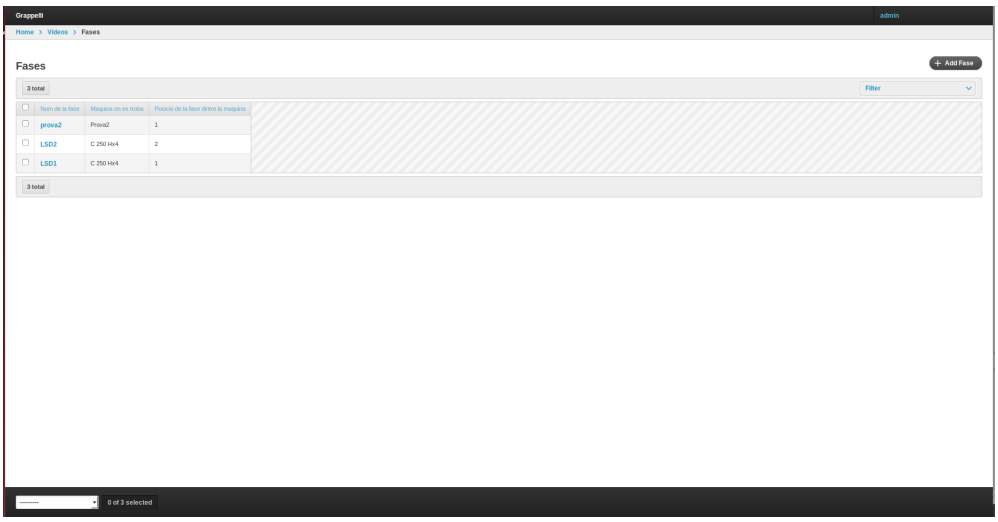

Figura 3.12.: Pàgina de gestió de les fases

Per afegir una nova fase, s'ha de clicar el botó *Add Fase*. Un cop clicat, ens trobarem en la següent pàgina:

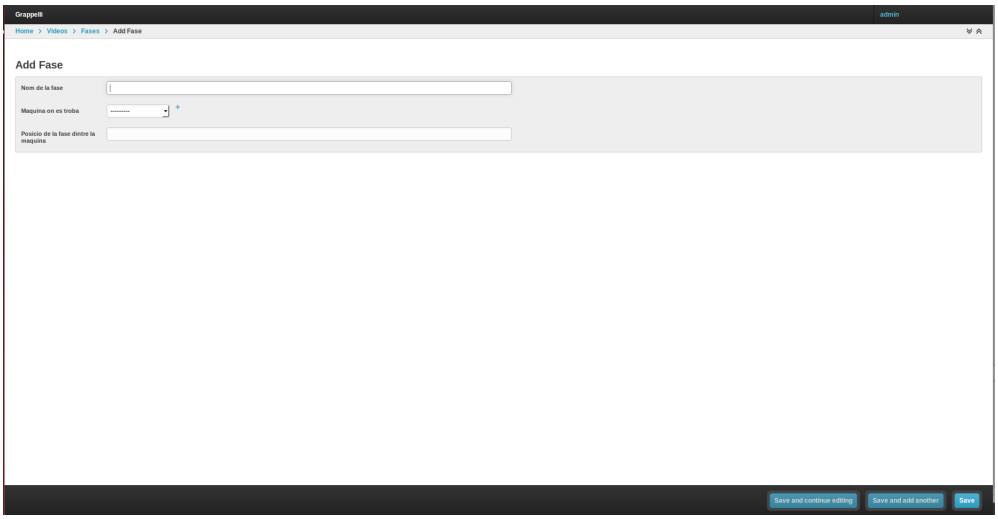

Figura 3.13.: Pàgina per afegir una fase

En aquesta pàgina s'hauran d'afegir les següents dades de la fase:

Nom de la fase El nom que s'ha definit per aquella fase.

Maquina on es troba Escollirem una de les màquines creades anteriorment perquè aquesta fase en formi part.

Posicio de la fase dintre la maquina Nombre per definir l'ordre en que s'ha de presentar cada

fase dintre de la màquina. S'ha d'anar en compte de no repetir el nombre en una mateixa màquina, ja que sinó les fases no s'ordenaran en l'ordre que nosaltres desitgem.

Un cop s'han afegit totes les dades, cliquem el botó *Save*, i serem redirigits a la pàgina de gestió de les fases, amb el missatge que la fase s'ha afegit correctament. Aquesta ja apareixerà en la llista de fases.

3.1.8. Modificar Fase

Per modificar les dades d'una fase, el que s'ha de fer és, des de la pàgina principal d'administració [3.2], clicar *Fases*. Un cop s'hagi clicat *Fases*, se'ns redirigirà a la pàgina de gestió de les fases [3.12].
Un cop en aquesta pàgina, s'ha de clicar la fase de la que volem modificar les seves dades. Quan la cliquem, se'ns redirigirà a la següent pàgina:

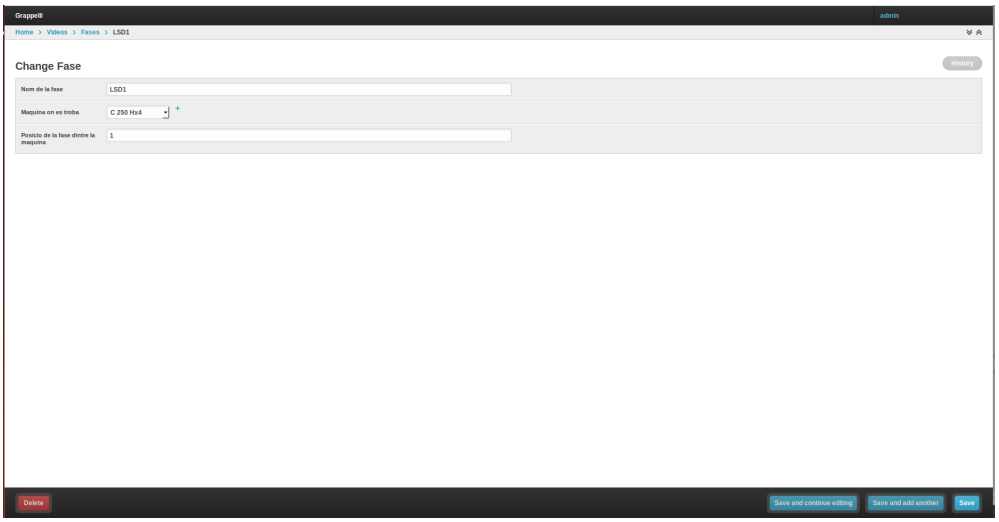

Figura 3.14.: Pàgina per modificar una fase

Un cop s'hagin modificat les dades, cliquem el botó *Save*, i se'ns redirigirà automàticament a la pàgina de gestió de les fases, on hi apareixerà un missatge conforme s'han modificat les dades de la fase correctament.

3.1.9. Eliminar Fase

Per eliminar una fase, des de la pàgina principal d'administració [3.2], el que s'ha de fer és clicar *Fases*. Al clicar *Fases*, se'ns redirigirà a la pàgina de gestió de les fases [3.12].
Un cop en aquesta pàgina, s'ha de seleccionar les fases que es vulguin eliminar, clicant el selector que hi ha a l'esquerra del nom de la fase. Si volem seleccionar totes les fases, cliquem el primer selector.
Un cop seleccionades la fase o fases que es volen eliminar, s'ha de dirigir a la part inferior esquerra de la pàgina, clicar la fletxa de la finestreta, i clicar *Delete selected Fases*.
Quan s'hagi clicat, se'ns redirigirà automàticament a la següent pàgina, on es demanarà que es confirmi l'acció que s'està a punt d'executar:

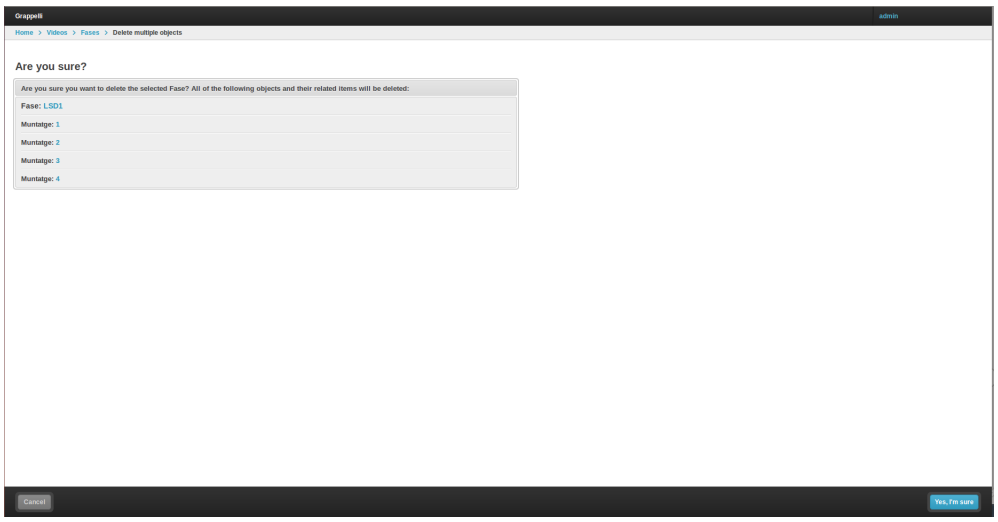

Figura 3.15.: Pàgina de confirmació d'eliminar una fase

Cliquem el botó *Yes, I'm sure*, i la fase o fases seran eliminades. També s'eliminaran tots els muntatges que estaven relacionats amb aquella fase. Explicaré que és un muntatge en l'apartat 3.1.13. Se'ns redirigirà a la pàgina de gestió de les fases, amb un missatge confirmant-nos que la fase o fases s'han eliminat correctament.

3.1.10. Afegir Vídeo

Per afegir un nou vídeo, el que s'ha de fer és, quan ens trobem en la pàgina principal d'administració [3.2], clicar *Videos*. Quan s'hagi clicat en *Videos*, se'ns redirigirà a la pàgina següent:

Figura 3.16.: Pàgina de gestió dels vídeos

En aquesta pàgina podem observar els vídeos que hi ha afegits fins aquell moment (si encara no s'ha afegit cap, no se'ns mostrarà cap). De tots els vídeos afegits, se'ns mostraran les següents dades:

Nom Nom que s'ha definit per aquell vídeo.

En aquesta pàgina, també podem buscar un vídeo en concret, clicant *Filter* i escollint el nom del vídeo que es vulgui visualitzar en aquell moment.
Per afegir un nou vídeo, s'ha de clicar el botó *Add Video*. Un cop clicat, ens trobarem en la següent pàgina:

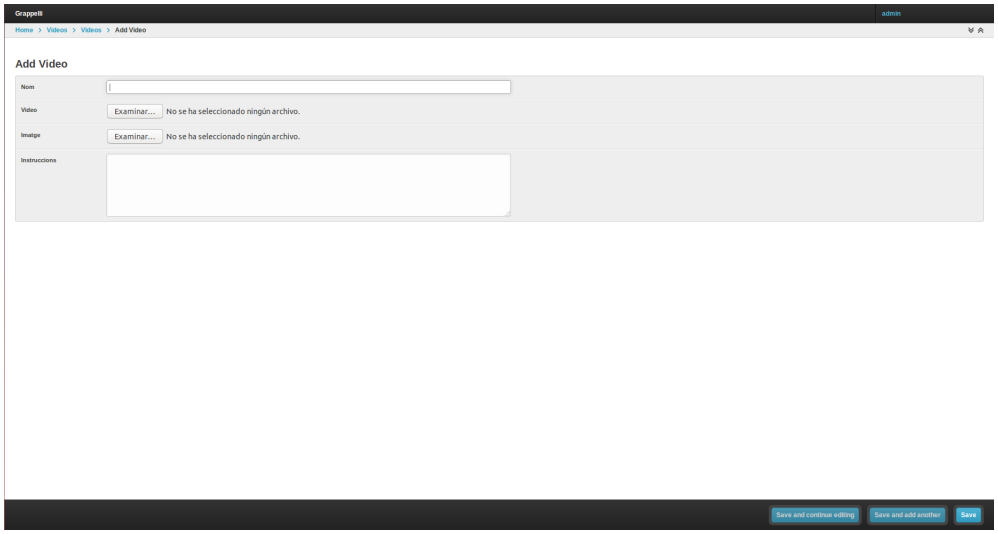

Figura 3.17.: Pàgina per afegir un vídeo

En aquesta pàgina haurem d'afegir les dades del vídeo:

Nom Nom que volem que tingui aquell vídeo.

Video Afegirem el fitxer de vídeo. Ha de ser en format `.mp4`. En aquest camp és important no posar en el nom del vídeo cap accent ni cap símbol semblant, sinó apareixerà un error.

Imatge Afegirem una imatge amb la que volem que es relacioni aquell vídeo. En aquest camp és important no posar en el nom de la imatge cap accent ni cap símbol semblant, sinó apareixerà un error.

Instruccions Afegirem totes les instruccions o alertes que es vulgui que apareguin per aquell vídeo.

Un cop s'ham afegit totes les dades, cliquem el botó *Save*, i serem redirigits a la pàgina de gestió dels vídeos, on se'ns mostrarà un missatge conforme el vídeo s'ha afegit correctament. Aquest ja apareixerà en la llista de vídeos.

3.1.11. Modificar Vídeo

Per modificar les dades d'un vídeo, el que s'ha de fer és, des de la pàgina principal d'administració [3.2], clicar *Videos*. Un cop s'hagi clicat *Videos*, se'ns redirigirà a la pàgina de gestió dels vídeos [3.16].
Un cop en aquesta pàgina, s'ha de clicar el vídeo del que volem modificar les dades. Quan el cliquem, se'ns redirigirà a la pàgina següent:

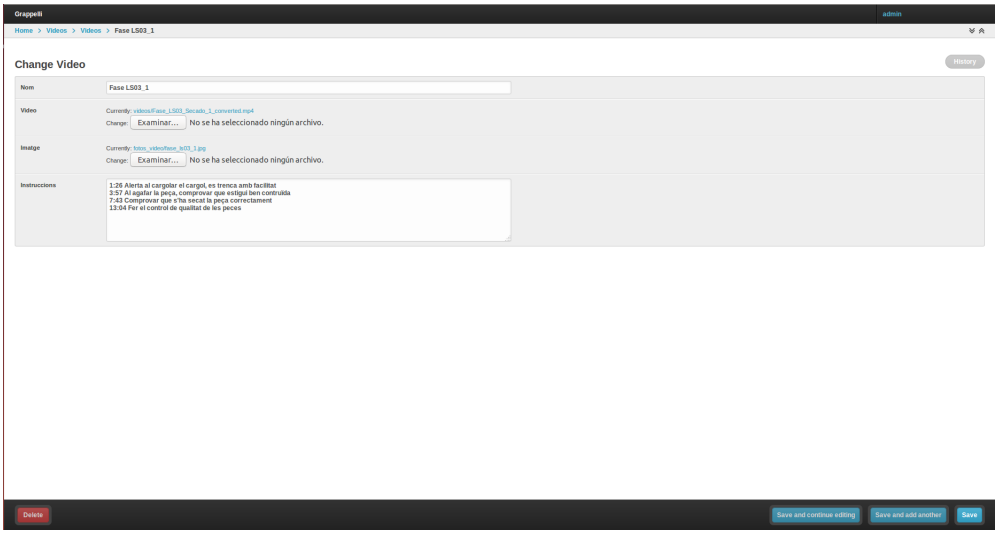

Figura 3.18.: Pàgina per modificar un vídeo

Un cop s'hagin modificat les dades, cliquem el botó *Save*, i se'ns redirigirà automàticament a la pàgina de gestió dels vídeos, on hi apareixerà un missatge conforme s'han modificat les dades del vídeo correctament.

3.1.12. Eliminar Vídeo

Per eliminar un vídeo, des de la pàgina principal d'administració [3.2], el que s'ha de fer és clicar *Videos*. Al clicar *Vídeos*, se'ns redirigirà a la pàgina de gestió dels vídeos [3.16].
Un cop en aquesta pàgina, s'ha de seleccionar els vídeos que es vulguin eliminar, clicant el selector que hi ha a l'esquerra del nom del vídeo. Si volem seleccionar tots els vídeos, cliquem el primer selector.
Un cop seleccionats el vídeo o vídeos que volem eliminar, s'ha de dirigir a la part inferior esquerra de la pàgina, clicar la fletxa de la finestreta, i clicar *Delete selected Videos*.
Quan s'hagi clicat, se'ns redirigirà automàticament a la pàgina següent, on es demanarà que es confirmi l'acció que s'està a punt d'executar:

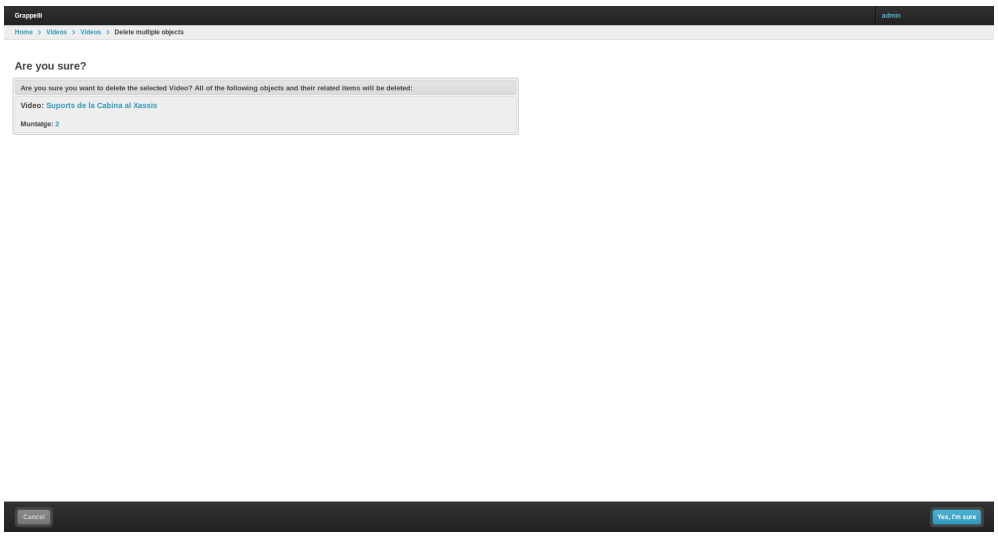

Figura 3.19.: Pàgina de confirmació d'eliminar un vídeo

Cliquem el botó *Yes, I'm sure*, i el vídeo o vídeos seran eliminats. També s'eliminaran tots els muntatges que estaven relacionats amb aquell vídeo. Explicaré que és un muntatge en l'apartat 3.1.13. Se'ns redirigirà a la pàgina de gestió dels vídeos, amb un missatge confirmant-nos que el vídeo o vídeos s'han eliminat correctament.

3.1.13. Ordenar els Vídeos dintre d'una Fase

Per tal de mostrar els vídeos dintre d'una fase en l'ordre que es desitgi, farem servir els muntatges. Sense els muntatges, els vídeos es mostrarien dintre d'una fase en l'ordre en que s'han afegit en l'administrador. I tampoc podríem posar un vídeo en més d'una fase. El muntatge és vital pel correcte funcionament de la web. Per tant, s'ha de fer sempre que vulguem afegir un nou vídeo en una fase.

3.1.13.1. Afegir Muntatge

Per afegir un nou muntatge, el que s'ha de fer és, quan ens trobem en la pàgina principal d'administració [3.2], clicant *Muntatges*. Quan s'hagi clicat en *Muntatges*, se'ns redirigirà a la pàgina següent:

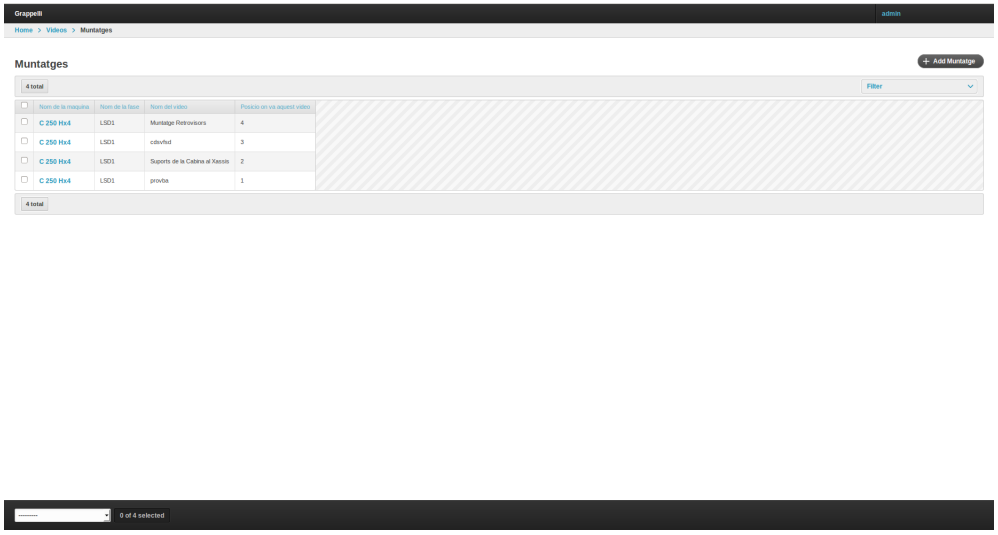

Figura 3.20.: Pàgina de gestió dels muntatges

En aquesta pàgina podem observar els muntatges que hi ha afegits fins aquell moment (si encara no s'ha afegit cap, no se'ns mostrarà cap). De tots els muntatges afegits, se'ns mostraran les següents dades:

Nom de la maquina Nom de la màquina on va aquest muntatge.

Nom de la fase Fase a la que pertany aquest muntatge.

Nom del video Vídeo pel que definim aquest muntatge

Posicio on va aquest video Posició on volem que vagi el vídeo dintre de la fase que hem definit anteriorment.

En aquesta pàgina també podem buscar un muntatge o muntatges concrets, clicant *Filter* i escollint la combinació entre les dades del muntatge que desitgem visualitzar en aquell moment. Per afegir un nou muntatge, s'ha de clicar el botó *Add Muntatge*. Un cop clicat, ens trobarem en la següent pàgina:

Figura 3.21.: Pàgina per afegir un muntatge

En aquesta pàgina haurem d'afegir les dades del muntatge:

Nom de la maquina Escollirem el nom de la màquina on va aquest muntatge.

Nom de la fase Escollirem el nom de la fase a la que pertany aquest muntatge.

Nom del video Escollirem el nom del vídeo pel que definim aquest muntatge

Posicio on va aquest video Escriurem la posició on volem que vagi el vídeo dintre de la fase que hem definit anteriorment. S'ha d'anar amb compte a no repetir el nombre per una mateixa fase, ja que d'aquesta manera no s'ordenaran correctament els vídeos.

Un cop s'han afegit totes les dades, cliquem el botó *Save*, i serem redirigits a la pàgina de gestió dels muntatges [3.20], on se'ns mostrarà un missatge conforme el muntatge s'ha afegit correctament. Aquest ja apareixerà en la llista de muntatges.

3.1.13.2. Modificar Muntatge

Per modificar les dades d'un muntatge, el que s'ha de fer és, des de la pàgina principal d'administració [3.2], clicar *Muntatges*. Un cop s'hagi clicat *Muntatges*, se'ns redirigirà a la pàgina de gestió dels muntatges [3.20].
Un cop en aquesta pàgina, s'ha de clicar el muntatge del que volem modificar les dades. Quan el cliquem, se'ns redirigirà a la següent pàgina:

Figura 3.22.: Pàgina per modificar un muntatge

Un cop s'hagin modificat les dades, cliquem el botó *Save*, i se'ns redirigirà automàticament a la pàgina de gestió dels muntatges, on hi apareixerà un missatge conforme s'han modificat les dades del muntatge correctament.

3.1.13.3. Eliminar Muntatge

Per eliminar un muntatge, des de la pàgina principal d'administració [3.2], el que s'ha de fer és clicar *Muntatges*. Al clicar *Muntatges*, se'ns redirigirà a la pàgina de gestió dels muntatges [3.20].
Un cop en aquesta pàgina, s'ha de seleccionar els muntatges que es vulguin eliminar, clicant el selector que hi ha a l'esquerra del nom del muntatge. Si volem seleccionar tots els muntatges, cliquem el primer selector.
Un cop seleccionats el muntatge o muntatges que volem eliminar, s'ha de dirigir a la part inferior esquerra de la pàgina, clicar la fletxa de la finestreta, i clicar *Delete selected Muntatges*.
Quan s'hagi clicat, se'ns redirigirà automàticament a la següent pàgina, on es demanarà que es confirmi l'acció que s'està a punt d'executar:

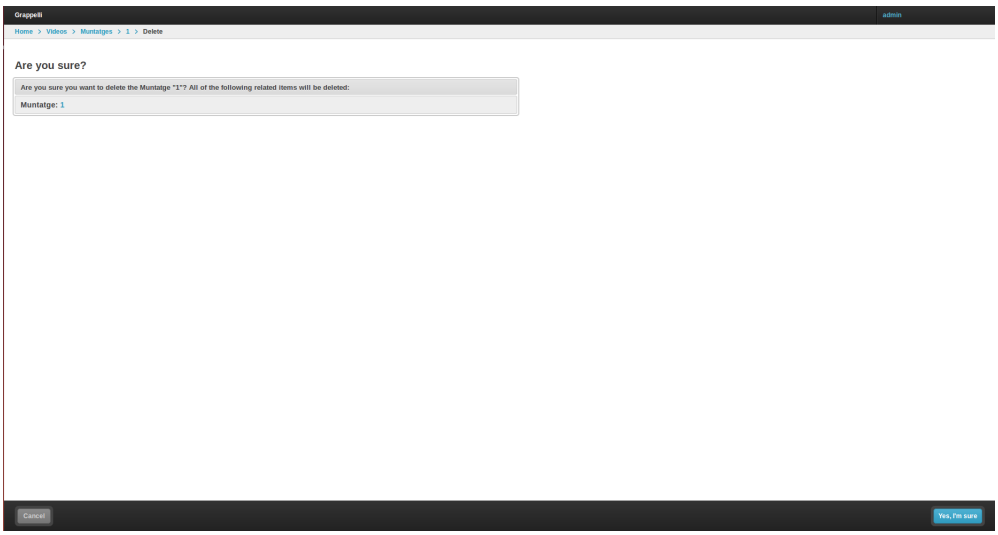

Figura 3.23.: Pàgina de confirmació d'eliminar un muntatge

Cliquem el botó *Yes, I'm sure*, i el muntatge o muntatges seran eliminats. Se'ns redirigirà a la pàgina de gestió dels muntatges, amb un missatge confirmant-nos que el muntatge o muntatges s'han eliminat correctament.

3.1.14. Tancar Sessió de la Pàgina d'Administració

Per tancar la sessió d'administració, des de qualsevol lloc on ens trobem, ens dirigim a la part superior dreta de l pàgina, i cliquem en *admin*. Se'ns mostrarà el següent:

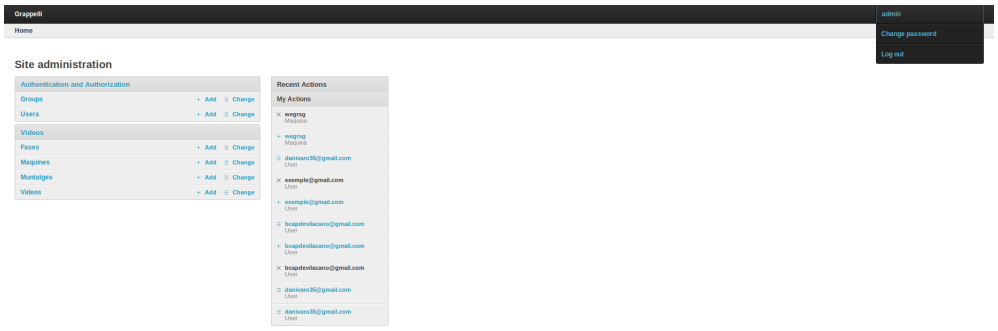

Figura 3.24.: Tancar sessió administració

Cliquem *Log out*, i tancarem la sessió, i serem redirigits a la pàgina d'autentificació de l'administració [3.1].

3.2. Manual per la Part d'Usuari

Un cop s'han fet totes les configuracions prèvies en la part d'administració, un usuari normal podrà accedir a la pàgina web, i executar una de les següents tasques:

- Reproduir un vídeo d'una màquina.
- Buscar una màquina, fase o vídeo.
- Enviar un correu perquè té problemes d'accés.
- Enviar un correu per explicar qualsevol observació, dubte o problema que tingui.
- Visualitzar les seves dades personals d'usuari.
- Modificar les seves dades d'usuari.

A continuació, explicaré que s'ha de fer per aconseguir realitzar qualsevol d'aquestes tasques.

3.2.1. Per reproduir un Vídeo d'una Màquina

Quan un usuari vol reproduir un video, el primer que ha de fer és autentificar-se. Sense autentificació no podrà accedir a cap màquina, per tant, tampoc podrà accedir a cap fase ni a cap vídeo.
Per autentificar-se, ha d'introduir el seu *usuari* i *contrasenya* en la següent pàgina:

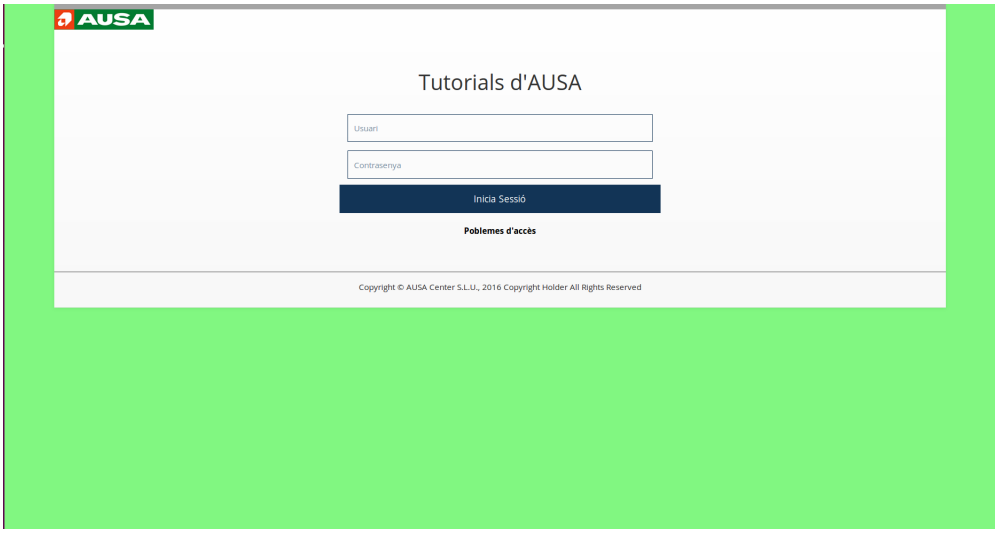

Figura 3.25.: Autentificació d'usuaris

Un cop l'usuari s'ha autentificat, serà redirigit a la següent pantalla, on se li mostraran totes les màquines de les que disposa per veure els seus tutorials, i també d'un navegador per poder accedir directament a qualsevol fase o vídeo:

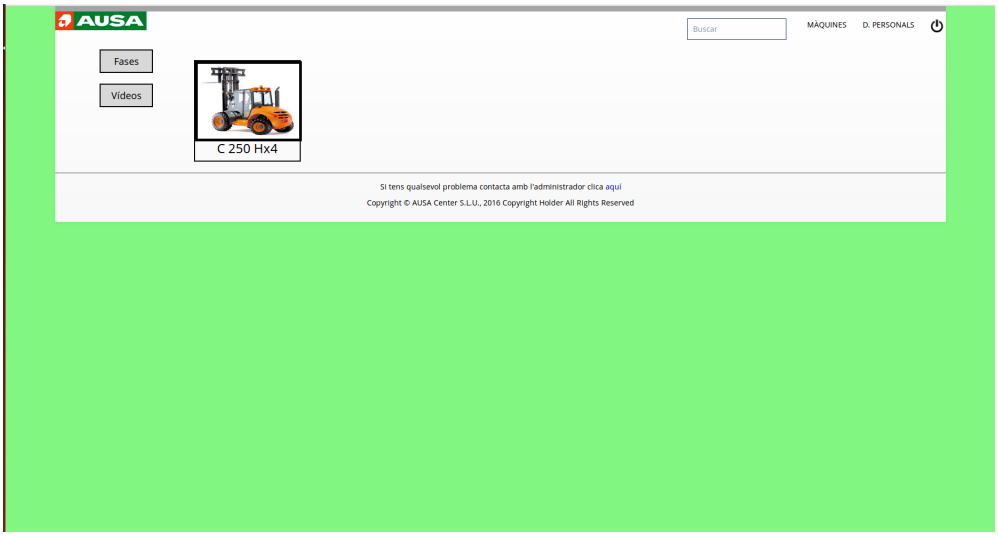

Figura 3.26.: Pàgina principal de la part d'usuari

Un cop s'ha clicat sobre la màquina de la que vol veure els vídeos, se li mostrarà la següent pantalla, on l'usuari ha d'escollir la fase de la màquina de la que li interessa veure els vídeos. Les fases es mostren en el ordre en que s'han de muntar:

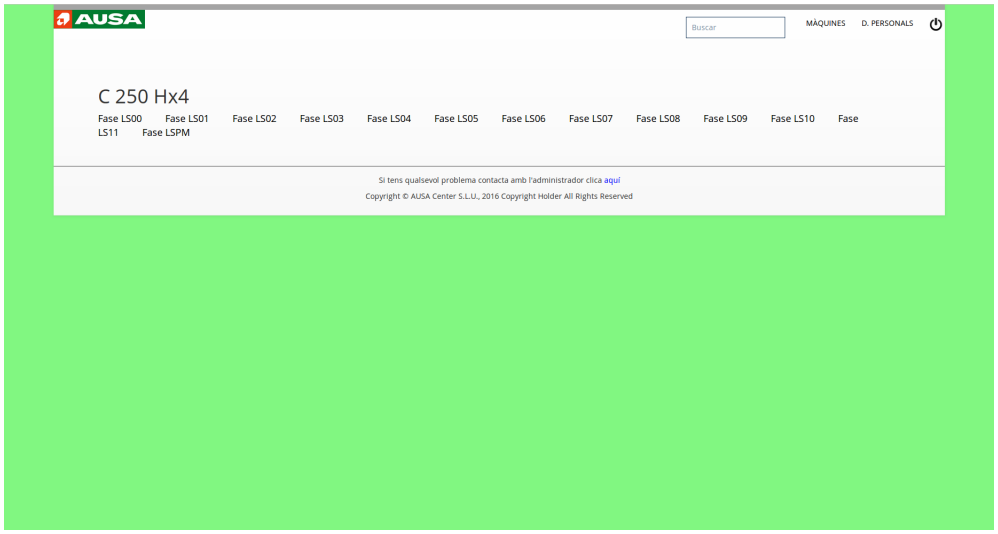

Figura 3.27.: Pàgina de les fases de la màquina

Després d'haver clicat sobre la fase que li interessa veure, se li mostraran tots els vídeos de la màquina que ha escollit abans que hi ha en aquella fase. Els vídeos estan en l'ordre en que s'han d'executar per muntar correctament aquella fase.
Aquí l'usuari ha de tornar a clicar sobre el vídeo que vol visualitzar:

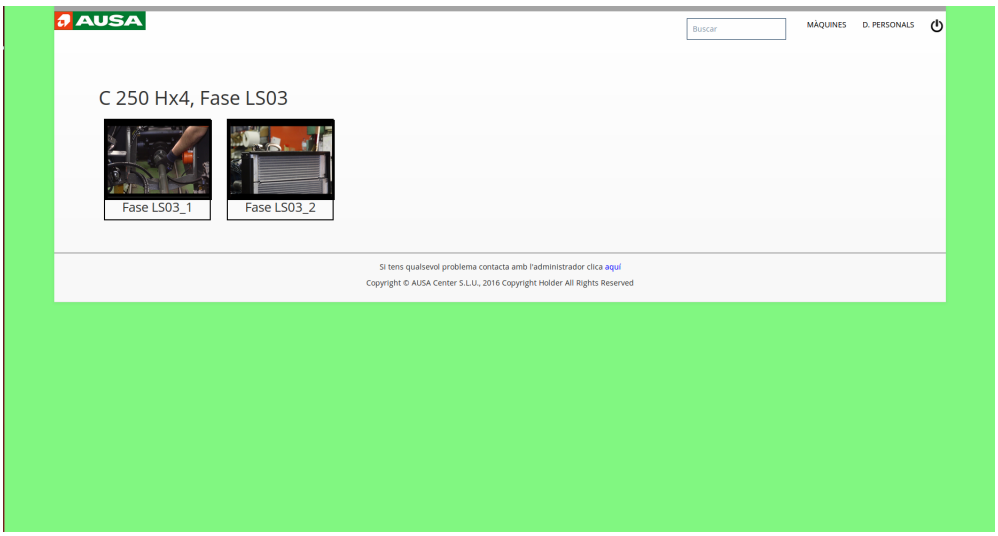

Figura 3.28.: Pàgina de les parts en que està dividida una fase

Per últim, un cop ha clicat sobre el vídeo, aquest es començarà a reproduir en la següent pantalla:

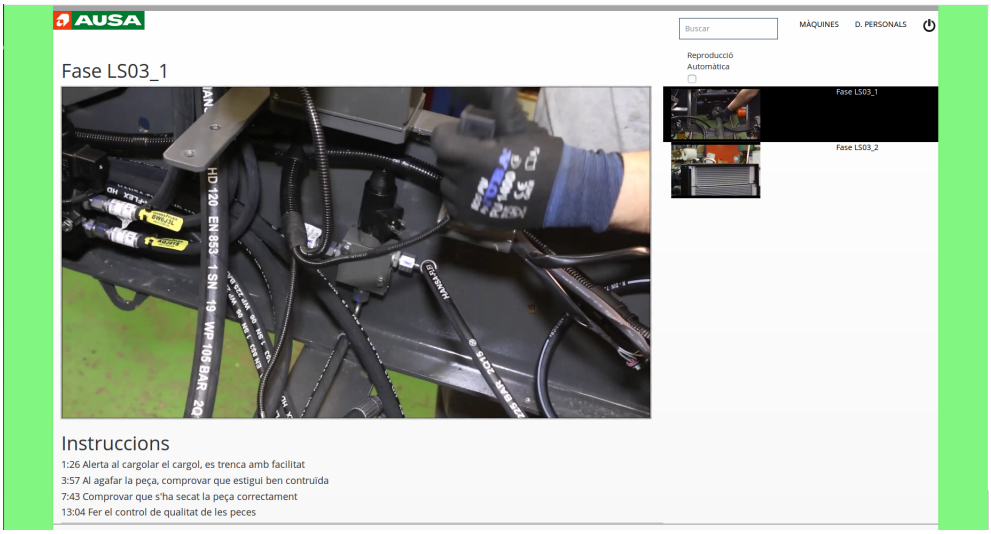

Figura 3.29.: Pàgina de reproducció del vídeo

Veiem que en aquesta pàgina, en gran se'ns està reproduint el vídeo que hem escollit anteriorment. A la dreta del vídeo en reproducció, veiem que es troben tots els vídeos d'aquella fase, marcat en negre el que s'està reproduint en aquest moment.
També podem veure que, a la dreta, a sobre de les miniatures dels vídeos de la fase, hi ha l'opció de *Reproducció Automàtica*. Clicant aquesta opció, els vídeos s'aniran reproduint un darrere l'altre, des de el que hem escollit fins al final.
Si no cliquem aquesta opció, quan acabi el vídeo que hem escollit, no es reproduirà res més. Llavors, si volem reproduir algun altre vídeo d'aquesta fase, el podem clicar en la llista de la dreta.

També podem observar que sota del vídeo apareix l'apartat *Instruccions*. Les *Instruccions* són indicacions d'on es troba una part important del vídeo o advertències que s'han de tenir en compte a l'hora de realitzar la tasca que es visualitza en aquell vídeo.
Si volem canviar de fase, hem de clicar el botó d'endarrere que ens proporciona el navegador. Si volem canviar de màquina, podem clicar el botó de dalt a la dreta *MÀQUINES*, i ens portarà on se'ns mostraven totes les màquines. Llavors hem de repetir el procés descrit abans per reproduir un altre vídeo.

3.2.1.1. Accés Directe a les Fases

Si l'usuari clica l'accés directe *Fases* de la pàgina principal, aquest serà redirigit a la següent pàgina, on se li mostraran el nom de totes les màquines de les que es disposa tutorial, amb links a totes les fases:

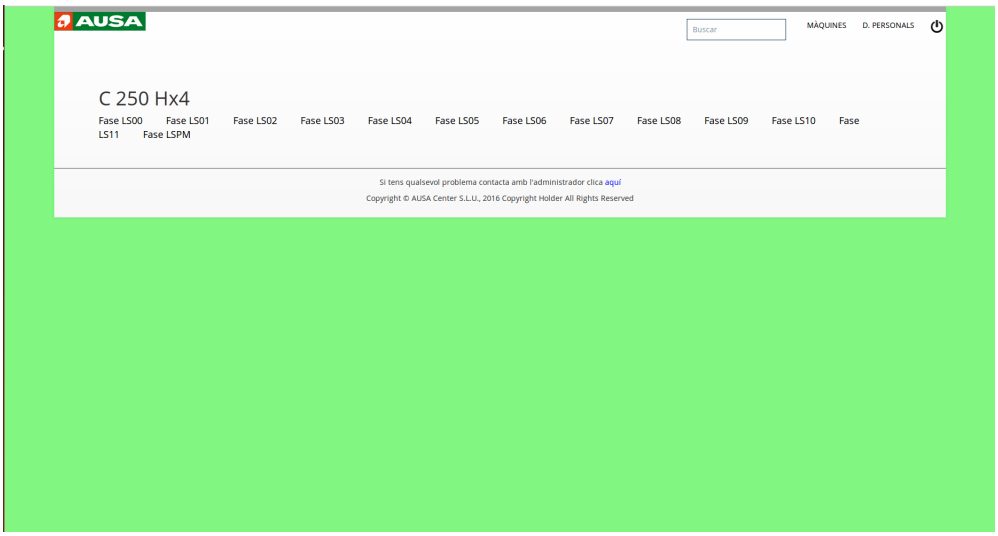

Figura 3.30.: Accés directe a les fases

En aquesta pàgina, ha d'escollir quina fase vol visualitzar els vídeos. Un cop escull una fase, se li mostraran els vídeos que formen aquella fase [3.28]. En aquesta pàgina ha d'escollir quin vídeo vol visualitzar, i quan el clica, se'l redirigeix a la pàgina de reproducció de vídeo [3.29].

3.2.1.2. Accés Directe als Vídeos

Si l'usuari clica l'accés directe *Vídeos* de la pàgina principal, aquest serà redirigit a la següent pàgina, on se li mostraran el nom de totes les màquines de les que es disposa tutorial, dividit per les fases que la formen. Dintre de cada fase, hi haurà els vídeos que la formen:

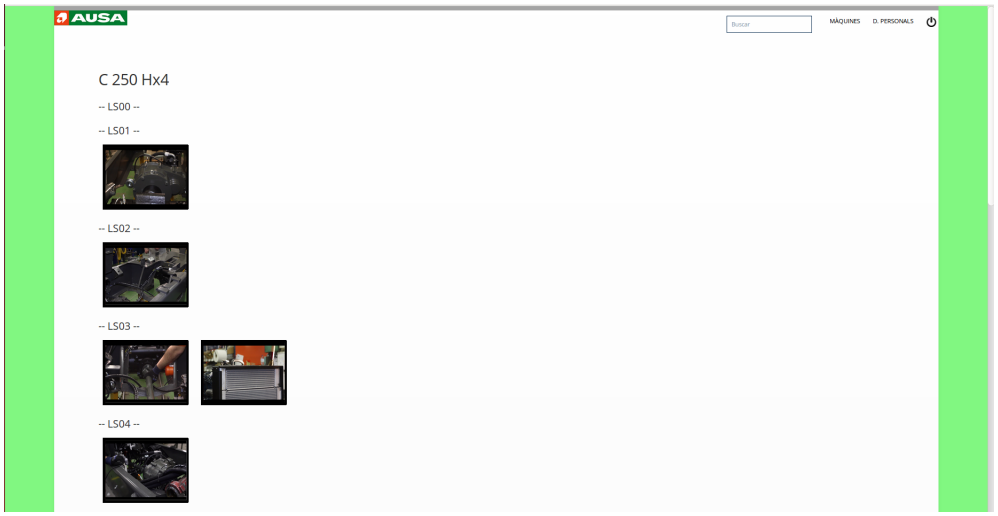

Figura 3.31.: Accés directe als vídeos

En aquesta pàgina, ha d'escollir quin vídeo vol visualitzar. Un cop escollit, serà redirigit a la pàgina de reproducció de vídeo [3.29].

3.2.2. Per Buscar una Màquina, Fase o Vídeo

Per poder buscar una màquina, una fase o un vídeo, primer l'usuari s'ha d'autentificar introduint el seu *usuari* i *contrasenya* en la pantalla d'autentificació d'usuaris [3.25], ja que l'opció de buscar només la tenen els usuaris autentificats.
Un cop l'usuari s'ha autentificat, es trobarà en la pantalla principal, on es mostren totes les màquines de les que disposa de tutorials [3.26].
Per realitzar una cerca, l'usuari ha d'escriure en el requadre de dalt a la dreta que posa *Buscar* el que desitgi buscar, sigui una màquina, una fase o un vídeo. Es poden cercar una o dues màquines a la vegada, separant cada paraula que es vol cercar amb una coma. Si s'escriuen tres o més paraules per cercar-les a la vegada, es mostra la següent pàgina d'error:

Figura 3.32.: Cerca amb excés de paraules per cercar

Un cop ha escrit el que vol cercar i ha clicat `enter`, si la seva cerca troba un o més resultats se li mostrarà la següent pantalla:

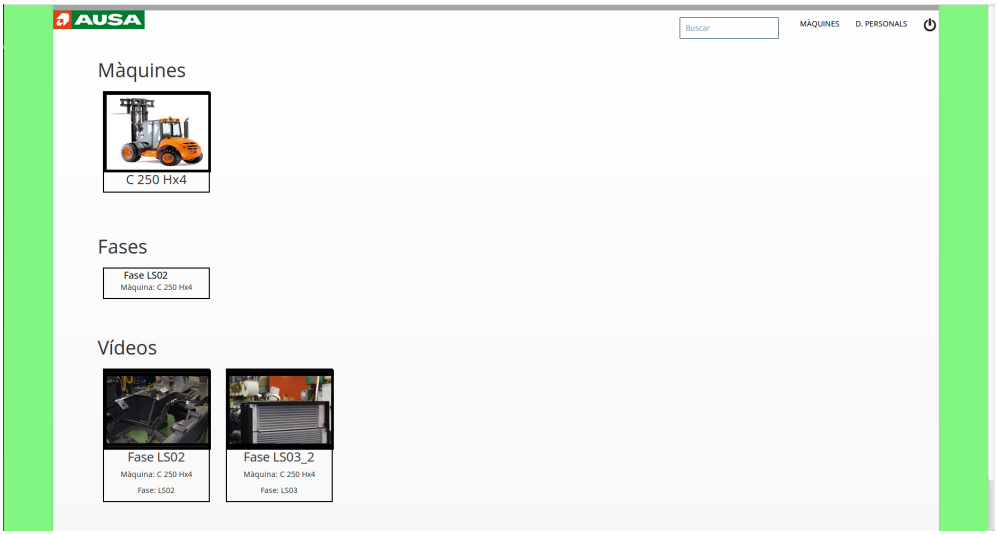

Figura 3.33.: Pàgina de cerca correcte

Si la cerca es correcte, llavors l'usuari ha de clicar l'opció que desitgi i seguir els passos descrits en l'apartat anterior, des de el punt on es trobi. Si al contrari, la seva cerca no troba cap resultat, se li mostrarà la següent pantalla:

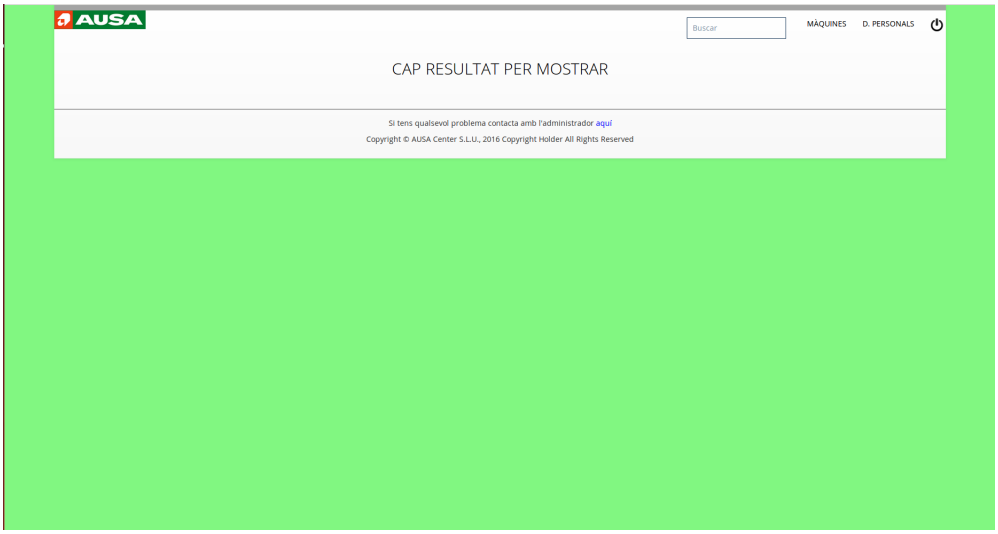

Figura 3.34.: Cerca no satisfactòria

En aquest punt, l'usuari pot tornar a realitzar una altre cerca de la manera descrita anteriorment, o clicar el botó *MÀQUINES* i seguir el procés descrit en l'apartat anterior.

3.2.3. Enviar Correu per Observacions, Dubtes o Problemes

Si un usuari, prèviament actualitzat, vol enviar un correu al l'administrador per comunicar-li alguna observació, dubte o problema que li hagi sorgit, només ha d'anar al peu de pàgina de qualsevol lloc on es trobi i clicar *aquí*. Un cop clicat, se li mostrarà una pantalla on ja surten el seu nom, cognoms i email on vol rebre les notificacions de l'empresa. L'usuari només ha d'escriure un missatge que s'enviarà a l'administrador, on hi escriurà el que ell vulgui:

Figura 3.35.: Contactar autentificat

Si es vol canviar el nom, cognoms o email on l'usuari vol rebre els correus, ho ha de fer des de la pàgina de modificar les dades personals, com explicaré en l'apartat [3.2.5].

Un cop escrit, només a de clicar el botó *Envia* i el seu correu s'enviarà a l'administrador de la web.
Un cop enviat el correu, l'usuari ja pot seguir fent servir la pàgina web per seguir veient tutorials.

3.2.4. Visualitzar les Dades d'Usuari

Si un usuari vol visualitzar les seves dades d'usuari que té per aquest lloc web, ha d'estar prèviament autentificat. Un cop autentificat, en qualsevol pàgina on es trobi, ha de dirigir-se a la part superior dreta de la pantalla i clicar en *D. PERSONALS*. Un cop clicat, se li mostrarà la següent pantalla:

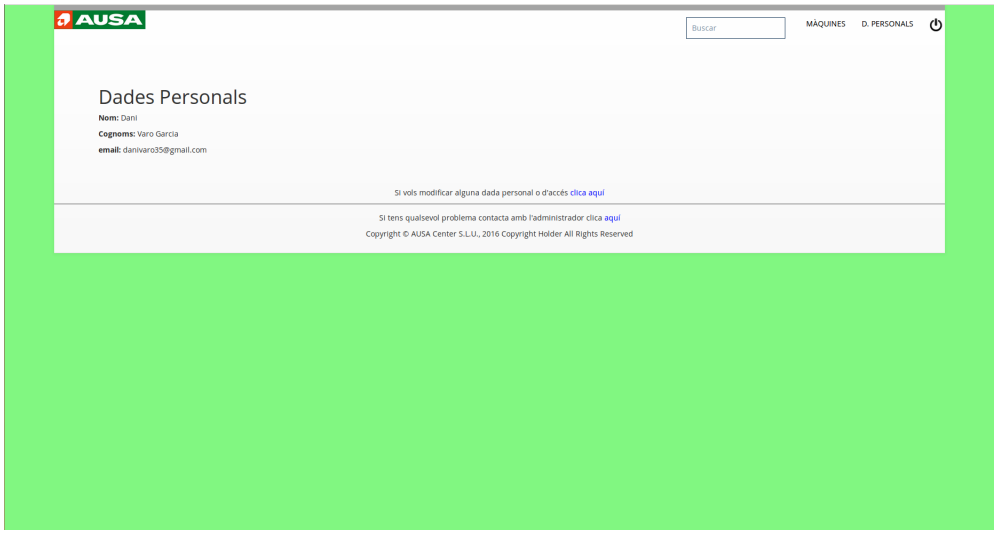

Figura 3.36.: Dades personals de l'usuari

Com podem observar, en aquesta pàgina es mostren el nom, cognoms i email personal de l'usuari. Per modificar alguna d'aquestes dades, s'ha de clicar en *clica aquí*, i serem redirigits a la pàgina de modificar les dades de l'usuari.

3.2.5. Modificar les Dades d'Usuari

Si un usuari vol modificar les seves dades, un cop autentificat, ha de clicar en *D. PERSONALS*, i després clicar *clica aquí*. Un cop fets aquests dos passos, ens trobarem en la següent pàgina:

Figura 3.37.: Pàgina per modificar les dades de l'usuari

Un cop ens trobem aquí, l'usuari pot modificar el seu nom, cognoms, email personal i contrasenya per accedir al lloc web. El que no pot modificar és el correu electrònic que es fa servir per username del lloc web. Això només ho pot fer l'administrador.

3.2.6. Tancar Sessió d'Usuari

Quan un usuari està prèviament autentificat, per tancar la sessió només ha de dirigir-se a la part dreta de dalt de tot de la pàgina web i clicar la icona de tancar sessió. La sessió es tancarà i redirigirà a l'usuari a la pantalla d'autentificació d'usuaris [3.25].

3.2.7. Enviar Correu per Problemes D'Accés

Quan un usuari de la pàgina web no pot accedir a veure els tutorials, en la pàgina d'autentificació, ha de clicar en *Problemes d'accés*.
Un cop ha clicat l'enllaç, li apareixerà una pàgina web com la següent:

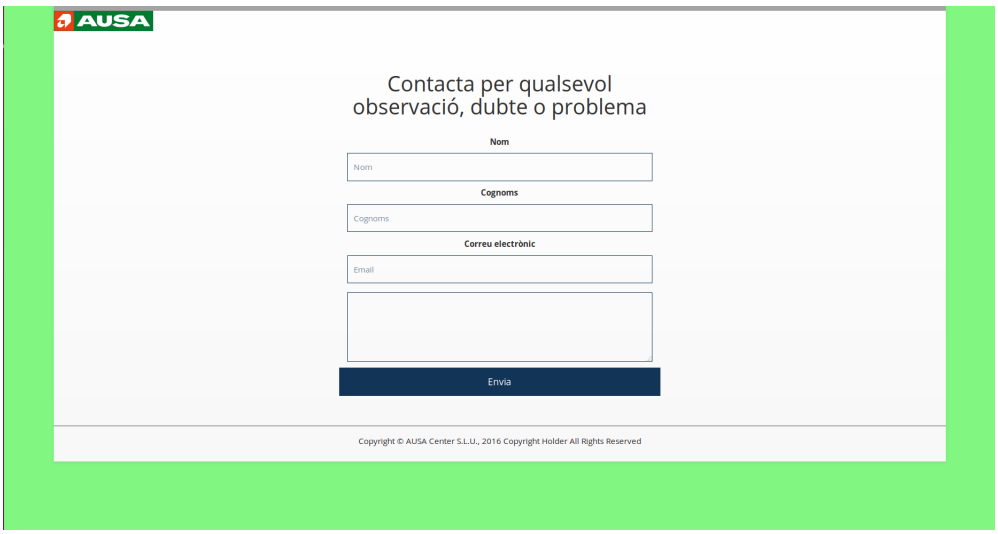

Figura 3.38.: Contactar no autentificat

L'usuari haurà d'introduir el seu nom, cognoms, email on vol rebre el correu i el problema que estigui tenint, és a dir, que és el que fa que no pugui accedir a la web.
Aquest correu se li enviarà a l'administrador, que se'n encarregarà d'arreglar el problema i comunicar-li a l'usuari que aquell problema està solucionat enviant-li un correu a l'email que l'usuari a proporcionat.

4. Problemes a l'Hora de Desenvolupar el Projecte

Alhora de desenvolupar el projecte m'he anat trobant amb una sèrie de problemes, els quals exposo a continuació:

1) Vídeos en fullscreen: Quan vaig intentar implementar que, si s'està reproduint un vídeo en fullscreen i està la opció de *Reproducció Automàtica* activada, el següent vídeo també es reproduís en fullscreen. Al intentar fer aquesta tasca, hem vaig trobar que en el navegador m'apareixia l'error `Failed to execute 'requestfullscreen' on 'element' api can only be initiated by a user gesture`. Vaig investigar aquest error, i vaig trobar que en la web de `stackoverflow` [Sta16] deien que no es permet a les webs que utilitzin aquesta eina. És a dir, per fer el vídeo en fullscreen, és necessària la interacció de l'usuari de la web. El motiu és que no volen que algú crei una pàgina web que pugui molestar la interacció amb el navegador de l'usuari.

2) Anotacions en els vídeos: Vaig intentar fer implementar un sistema per posar anotacions en els vídeos que fos dinàmic. El que vaig intentar fer es que des de una pàgina web, poguessis anar posant anotacions en els vídeos, com ho fa la plataforma `YouTube`. Però hem vaig trobar que aquest aspecte de les webs, en aquests moments no està molt desenvolupat. No vaig trobar casi res d'informació sobre aquest tema. I els `frameworks` que vaig trobar que feien aquesta feina, estaven en desenvolupament i no tenien documentació escrita, i no sabia com utilitzar-los ni si funcionarien. Per tant, vaig decidir no implementar les anotacions.

3) La pàgina search.html: Quan vaig implementar la funció de cerca en la web, vaig veure que al utilitzar el mètode `GET`, la pàgina `search.html` no detectava que l'usuari estava autentificat. Vaig buscar com solucionar aquest problema, però no vaig trobar la solució. Per tant, el que vaig fer és, no utilitzar el fitxer `base.html` per aquesta pàgina, i crear-la des de zero. Així, tot i que no detecti que l'usuari està autentificat, hi apareix tot el que ha d'aparèixer quan un usuari està autentificat.

4) Per enviar correus des de el projecte Django: Per enviar correus des de el projecte `Django`, si es volen enviar des de `gmail`, com faig jo en aquest projecte, per poder enviar els correus de l'aplicació, s'ha d'anar a la configuració de gmail, entrar en *mi cuenta*, clicar *Aplicaciones menos seguras* i posar *Activar*. Si no fem aquest canvi, des de el projecte ens dirà que el correu no s'ha pogut enviar.

5) Certificat Auto-certificat: Al utilitzar un certificat creat i certificat per mi mateix, i no per una autoritat certificadora, els vídeos en els smartphones i tablets no es reprodueixen, només es reprodueix en els ordinadors. Si el certificat el certifica una autoritat certificadora, aquest projecte ja està configurat per reproduir els vídeos en qualsevol plataforma. Vaig trobar aquesta raó de que no es reproduïssin els vídeos en els smartphones i tablets

en la web de StackOverflow [Dev16]. He pogut comprovar que els vídeos es reprodueixen en qualsevol plataforma, perquè per provar-ho, vaig fer la configuració de `Apache` amb `HTTP` (sense certificat), i els vídeos es reprodueixen perfectament en qualsevol plataforma.

6) Autoplay: Al fer la prova de reproduir els vídeos en els smartphones i tablets, vaig veure que aquests no es començaven a reproduir automàticament en tots els navegadors, sinó que en alguns si que ho feia i en alguns no. Vaig investigar, i en la pàgina de `JWPlayer` [JWP16] deien els navegadors que tenien implementat el tag `autoplay` de `HTML`. Dels navegadors de smartphones i tablets, només Internet Explorer de Windows Phone i Firefox d'Android el tenen implementat. Per tant, en tots els altres navegadors, l'usuari haurà de clicar el play per reproduir el vídeo. Aquest problema no crec que triguin molt a solucionar-lo, ja que quan ho implementin en els navegadors, els vídeos ja es posaran a reproduir automàticament.

5. Requeriments

En aquest moments, la plataforma web està preparada per funcionar correctament en qualsevol dispositiu, ja sigui ordinador, smartphone o tablet. L'únic que faltaria seria comprar un certificat autentificat per una autoritat certificadora. No he pogut comprovar el funcionament del certificat, perquè l'empresa no volia comprar-ne cap en aquests moments, però puc assegurar que comprant un certificat la plataforma web és completament multi-plataforma.

6. Possibles Millores

Algunes de les millores que es podrien afegir al projecte són les següents:

1) Utilitzar el sistema multi-llenguatge: El sistema multi-llenguatge per el projecte està preparat per fer servir, però no l'utilitzo. Si en un futur es vol tenir aquesta web en més d'un idioma, només s'ha d'utilitzar aquest sistema. En aquests moments la web només està en català.

2) Anotacions: Implementar quan sigui possible un sistema d'anotacions per poder posar anotacions en els vídeos des de la plataforma web.

7. Conclusions

Després d'haver realitzat aquest projecte, puc dir que s'han complert els objectius que es van plantejar a l'hora de començar el desenvolupament del projecte. Aquests objectius són:

- Desenvolupar una plataforma web per que els treballadors de l'empresa AUSA Center S.L.U. puguin accedir i visualitzar vídeos de com es realitza el muntatge de les diferents màquines de l'empresa.
- La plataforma web havia de ser un sistema multi-plataforma.
- Aquesta plataforma web havia de permetre afegir i extreure dades (usuaris, màquines, fases i vídeos) d'una manera ràpida i senzilla.
- Havia de presentar dues parts:
  - Una part d'administració, des d'on poder afegir, modificar o eliminar usuaris, màquines, fases i vídeos.
  - I una altre part pels treballadors, en la qual poden accedir per visualitzar els tutorials, veure les seves dades personals i modificar-les si cal, i enviar correus a l'administrador per si tenen algun dubte, problema o observació per compartir amb l'administrador.
- Proporcionar als treballadors una eina molt potent per poder aprendre d'una forma més ràpida i visual com es realitzen els muntatges de les diferents màquines.

Les principals dificultats que m'he trobat a l'hora de desenvolupar aquest projecte han sigut:

- Passar d'un vídeo que s'està reproduint en fullscreen, utilitzant la reproducció automàtica, que el següent vídeo també es reprodueixi en fullscreen. Aquesta tasca no la vaig poder arribar a aconseguir per que els navegadors no deixen posar cap element d'una web en fullscreen sense la interacció de l'usuari.
- Vaig intentar implementar un sistema per posar anotacions en els vídeos. Però hem vaig trobar que tots els frameworks que es dediquen a fer aquesta tasca estan en desenvolupament i no tenien documentació escrita. Davant d'aquesta situació, vaig decidir no implementar el sistema d'anotacions.
- Quan vaig penjar el projecte en el servidor activant el servei SSL, vaig trobar-me en que des de els smartphones i tablets no es reproduïen els vídeos, però des de els ordinadors si. Vaig investigar-ho, i vaig trobar que era culpa del certificat que faig servir, ja que aquest no es d'una autoritat certificadora. Al utilitzar un certificat d'una autoritat certificadora, els vídeos ja es reproduiran en totes les plataformes.
- Al reproduir els vídeos des de un smartphone o tablet, depenent del navegador que s'utilitza, aquests no es començaven reproduïen automàticament. Vaig investigar-ho i

això es degut a que en aquestes plataformes encara no tenen implementada aquesta funció. Però un cop l'implementin, els vídeos ja es reproduiran automàticament sense problema.

Després d'haver realitzat aquest, arribo a la conclusió de que per mi ha sigut productiu la seva realització ja que m'ha permès aprendre a utilitzar eines noves que no havia après en el grau, com per exemple, Django. També m'ha permès aprendre més com es treballa amb les plataformes webs.
A més a més, també he aprés a resoldre els problemes que m'anaven apareixent per mi sol, sense suport de cap professor, a afrontar-los, analitzar-los i solucionar-los si es podia. També m'ha aportat un aprenentatge més profund a l'hora d'utilitzar la documentació de les eines que he utilitzat per desenvolupar el projecte.

Bibliografia

[Ang16] Curso de AngularJS y REST. *15.1 Estructura de ficheros*. 2016. URL: http://cursoangularjs.es/doku.php?id=unidades:15_avanzado:01_ficheros.

[Cul16] Culturacion. *¿Qué es Apache?* 2016. URL: http://culturacion.com/que-es-apache/.

[dbE16] db-Engines. *System Properties Comparison MySQL vs. SQLite*. 2016. URL: http://db-engines.com/en/system/MySQL%3BSQLite.

[Dev16] Android Developers. *Autoplay, Loop and Muted*. 2016. URL: http://developer.android.com/training/articles/security-ssl.html.

[Doc16a] Django Documentation. *Explicació del middleware Django*. 2016. URL: https://docs.djangoproject.com/en/1.7/ref/middleware/.

[Doc16b] Django Documentation. *Gestió dels fitxers estàtics en Django*. 2016. URL: https://docs.djangoproject.com/es/1.9/ref/settings/.

[Doc16c] Django Documentation. *Model de la taula Users de Django*. 2016. URL: https://docs.djangoproject.com/es/1.9/ref/contrib/auth/.

[Git16] GitBook. *Estructura de un proyecto en Laravel*. 2016. URL: https://richos.gitbooks.io/laravel-5/content/capitulos/chapter4.html.

[HTM16] HTML5fácil. *Las principales características de Angularjs*. 2016. URL: http://html5facil.com/tutoriales/las-principales-caracteristicas-de-angularjs/.

[JWP16] JWPlayer. *Autoplay, Loop and Muted*. 2016. URL: https://www.jwplayer.com/html5/autoloop/.

[K W12] J. F. Kurose i K. W. Ross. *Computer Networking: a top-down approach*. 6th revised. New Jersey, USA: Pearson, 2012.

[LIG16] LIGHTTPD. *LIGHTTPD*. 2016. URL: https://www.lighttpd.net/.

[Lin16] Desde Linux. *LIGHTTPD – un servidor web muy ágil y liviano*. 2016. URL: http://blog.desdelinux.net/lighttpd-un-servidor-web-muy-agil-y-liviano/.

[Pla16] Platzi. *Laravel, el mejor framework en PHP*. 2016. URL: https://platzi.com/blog/laravel-framework-php/.

[Sli16] SlideShare. *Pylons*. 2016. URL: http://www.slideshare.net/DulceRomeroGarcia/pylons-41734126.

[Sta16] StackOverflow. *Run a website in fullscreen mode*. 2016. URL: http://stackoverflow.com/questions/29281986/run-a-website-in-fullscreen-mode.

[vsC16a] vsChart. *MongoDB vs. Redis vs. CouchDB*. 2016. URL: http://vschart.com/compare/mongodb/vs/redis-database/vs/couchdb.

[vsC16b] vsChart. *Oracle vs. PostgreSQL*. 2016. URL: http://vschart.com/compare/oracle/vs/postgresql.

[vsC16c] vsChart. *Pyramid vs Django*. 2016. URL: http://vschart.com/compare/pyramid-web-framework/vs/django-framework.

[vsC16d] vsChart. *Pyramid vs Django*. 2016. URL: http://vschart.com/compare/angularjs/vs/laravel.

[Wik16a] Wikipedia. *AngularJS*. 2016. URL: https://es.wikipedia.org/wiki/AngularJS.

[Wik16b] Wikipedia. *Base de datos*. 2016. URL: https://es.wikipedia.org/wiki/Base_de_datos.

[Wik16c] Wikipedia. *Base de datos relacional*. 2016. URL: https://es.wikipedia.org/wiki/Base_de_datos_relacional.

[Wik16d] Wikipedia. *CouchDB*. 2016. URL: https://es.wikipedia.org/wiki/CouchDB#Caracter.C3.ADsticas_principales.

[Wik16e] Wikipedia. *Django (framework)*. 2016. URL: https://es.wikipedia.org/wiki/Django_%28framework%29.

[Wik16f] Wikipedia. *Document-oriented database*. 2016. URL: https://en.wikipedia.org/wiki/Document-oriented_database.

[Wik16g] Wikipedia. *Front-end y Back-end*. 2016. URL: https://es.wikipedia.org/wiki/Front-end_y_back-end#Inform.C3.A1tica.

[Wik16h] Wikipedia. *Hoja de estilos en cascada*. 2016. URL: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada.

[Wik16i] Wikipedia. *HTML*. 2016. URL: https://es.wikipedia.org/wiki/HTML.

[Wik16j] Wikipedia. *JavaScript*. 2016. URL: https://es.wikipedia.org/wiki/JavaScript.

[Wik16k] Wikipedia. *jQuery*. 2016. URL: https://es.wikipedia.org/wiki/JQuery.

[Wik16l] Wikipedia. *Key-value database*. 2016. URL: https://en.wikipedia.org/wiki/Key-value_database.

[Wik16m] Wikipedia. *Laravel*. 2016. URL: https://es.wikipedia.org/wiki/Laravel.

[Wik16n] Wikipedia. *MongoDB*. 2016. URL: https://es.wikipedia.org/wiki/MongoDB.

[Wik16o] Wikipedia. *MySQL*. 2016. URL: https://es.wikipedia.org/wiki/MySQL.

[Wik16p] Wikipedia. *Nginx*. 2016. URL: https://es.wikipedia.org/wiki/Nginx.

[Wik16q] Wikipedia. *NoSQL*. 2016. URL: https://es.wikipedia.org/wiki/NoSQL.

[Wik16r] Wikipedia. *Oracle Database*. 2016. URL: https://es.wikipedia.org/wiki/Oracle_Database#Caracter.C3.ADsticas.

[Wik16s] Wikipedia. *PostgreSQL*. 2016. URL: https://es.wikipedia.org/wiki/PostgreSQL.

[Wik16t] Wikipedia. *Pylons project*. 2016. URL: https://en.wikipedia.org/wiki/Pylons_project.

[Wik16u] Wikipedia. *Redis*. 2016. URL: https://en.wikipedia.org/wiki/Redis.

[Wik16v] Wikipedia. *Servidor*. 2016. URL: https://es.wikipedia.org/wiki/Servidor.

[Wik16w] Wikipedia. *Servidor HTTP Apache*. 2016. URL: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache.

[Wik16x] Wikipedia. *SQLite*. 2016. URL: https://es.wikipedia.org/wiki/SQLite.

[Wik16y] Wikipedia. *Twitter Bootstrap*. 2016. URL: `https://es.wikipedia.org/wiki/Twitter_Bootstrap`.

Part II.

Annexes

1. settings.py

```
"""
Django settings for tutorials project.

For more information on this file, see
https://docs.djangoproject.com/en/1.7/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.7/ref/settings/
"""

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
from django.conf.global_settings import TEMPLATE_CONTEXT_PROCESSORS as TCP
import os
import socket

BASE_DIR = os.path.dirname(os.path.dirname(__file__))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.7/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'y-1os0cw^$j-x_q%3d(qcj)f)a08a7*a44fj65@j#6iy*r)3a2'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

TEMPLATE_DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = (
    #admin stuff
    'grappelli',
    'grappelli_modeltranslation',

    #'filebrowser',
    'djrill',

    #plugins
    'modeltranslation',
    'rosetta',

    #admin default
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    #own applications
    #'treballadors'
    'videos',
)

MIDDLEWARE_CLASSES = (
    'django.middleware.gzip.GZipMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
)

ROOT_URLCONF = 'tutorials.urls'

WSGI_APPLICATION = 'tutorials.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.7/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Internationalization
# https://docs.djangoproject.com/en/1.7/topics/i18n/

LANGUAGE_CODE = 'en'

DEFAULT_CHARSET = 'utf-8'
FILE_CHARSET = 'utf-8'

TIME_ZONE = 'Europe/Madrid'

USE_I18N = True
USE_L10N = True
USE_TZ = True

MODELTRANSLATION_DEFAULT_LANGUAGE = 'en'
LANGUAGES = (
    ('en', u'English'),
```

```
    ('ca', u'Catalan'),
)

LOCALE_PATHS = (
    os.path.join(BASE_DIR, 'locale'),
)


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.7/howto/static-files/

STATIC_ROOT = os.path.join(BASE_DIR, 'wsgi', 'static', 'root')
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'wsgi/static'),
)

STATIC_URL = '/static/'

STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)


MEDIA_ROOT = os.path.join(BASE_DIR, 'wsgi', 'media')

MEDIA_URL = '/media/'

VIDEO_TYPES = 'mp4'

IMAGE_TYPES = 'jpg'

TEMPLATE_DIRS = (
    os.path.join(BASE_DIR, 'templates'),
)

TEMPLATE_CONTEXT_PROCESSORS = TCP + (
    'django.core.context_processors.i18n',
    'django.core.context_processors.request',
    'django.contrib.auth.context_processors.auth',
    'django.core.context_processors.media',
)

MANAGERS = ('Daniel Varo Garcia', 'danivaro35@gmail.com')

#for authentication
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)


#Email del administrador on s'enviaran els correus de contacte de la plataforma
EMAIL = 'danivaro35@gmail.com'
```

```
PASSWORD = '78150087'

LOGIN_URL = "/en/login"
LOGIN_REDIRECT_URL = "/en/login"
```

2. models.py

```
#-*- encoding: utf-8 -*-
from django.db import models
from django.contrib.auth.models import User
#Per poder posar imatges en els models
from PIL import Image
# Rebem els senyals pre_delete i delete per esborrar el fitxer associat a la
    instancia del model
from django.db.models.signals import pre_delete, pre_save
from django.dispatch.dispatcher import receiver
# Per sobreescriure una imatge
from django.core.files.storage import FileSystemStorage
from django.conf import settings
import os
#Per restringuir l'acces de fitxers en el filefield
from django.conf import settings
from django.core.files.storage import default_storage
from django.core.files.base import ContentFile
from django.core.exceptions import ValidationError

#funcio perque en el filefield nomes s'acceptin fitxers mp4
def validate_file_type(upload):
    #Fem que el fitxer de pujada sigui accessible per analitzar-lo per guardar-lo
        com a arxiu temporal
    file_name = str(upload)
    #Agafem la extensio del fitxer per veure si es el que ens interessa
    extension = file_name.split('.', 1)[1]
    #Fer sortir un error de validacio si el fitxer que pujem no es del tipus que
        volem acceptar
    if extension not in settings.VIDEO_TYPES:
        raise ValidationError('File type not supported. MP4 recommended')

#Funcio perque el imagefield nomes accepti fitxers jpg
def validate_image_type(upload):
    name = str(upload)
    extension = name.split('.', 1)[1]
    if extension not in settings.IMAGE_TYPES:
        raise ValidationError('File type not supported. JPG recommended')


#Classe per sobreescriure una imatge amb el mateix nom
class OverwriteStorage(FileSystemStorage):
    def get_available_name(self, name):
        # Si el nom del fitxer existeix, borra el que hi havia i guarda la nova imatge
        if self.exists(name):
            os.remove(os.path.join(settings.MEDIA_ROOT, name))
        return name


#Models de la base de dades
class Maquina(models.Model):
```

```
    name = models.CharField(verbose_name=u"Nom", max_length=128)
    image = models.ImageField(storage=OverwriteStorage(),
        upload_to='fotos_maquines', validators=[validate_image_type],
        verbose_name=u"Imatge")

    def __str__(self):
        return "%s"%self.name

    class Meta:
        verbose_name = u"Maquina"
        verbose_name_plural = u"Maquines"

class Fase(models.Model):
    name_phase = models.CharField(verbose_name=u"Nom de la fase", max_length=128)
    name_machine = models.ForeignKey(Maquina, verbose_name=u"Maquina on es troba",
        related_name='phase_machine')
    num_ordre = models.CharField(verbose_name=u"Posicio de la fase dintre la
        maquina", max_length=128)

    def __str__(self):
        return "%s"%self.name_phase

    class Meta:
        verbose_name = u"Fase"
        verbose_name_plural = u"Fases"

class Video(models.Model):
    name = models.CharField(verbose_name=u"Nom", max_length=128)
    video = models.FileField(storage=OverwriteStorage(), upload_to='videos',
        validators=[validate_file_type],verbose_name=u"Video")
    photo = models.ImageField(storage=OverwriteStorage(), upload_to='fotos_video',
        validators=[validate_image_type], verbose_name=u"Imatge")
    instruccions = models.TextField(verbose_name=u"Instruccions")

    def __str__(self):
        return "%s"%self.name

    class Meta:
        verbose_name = u"Video"
        verbose_name_plural = u"Videos"


class Muntatge(models.Model):
    id_maquina = models.ForeignKey(Maquina, verbose_name=u'Nom de la maquina',
        related_name='id_machine')
    id_fase = models.ForeignKey(Fase, verbose_name=u'Nom de la fase',
        related_name='id_phase')
    id_video = models.ForeignKey(Video, verbose_name=u'Nom del video',
        related_name='id_video')
    num_ordre = models.CharField(verbose_name=u"Posicio on va aquest video",
        max_length=128) #POSAR QUE NO ES PUGUI REPETIR EN UNA MATEIXA FASE

    def __str__(self):
```

```
        return "%s"%self.num_ordre

    class Meta:
        verbose_name = u"Muntatge"
        verbose_name_plural = u"Muntatges"


#Per esborrar una imatge quan esborrem la maquina
@receiver(pre_delete, sender=Maquina)
def maquina_delete(sender, instance, **kwargs):
    # Passem False perque el FileField no guarda el model.
    instance.image.delete(False)

#Per esborrar la imatge i el video quan esborrem el video
@receiver(pre_delete, sender=Video)
def video_delete(sender, instance, **kwargs):
    # Passem False perque el FileField no guarda el model.
    instance.video.delete(False)
    instance.photo.delete(False)
```

3. admin.html

```
from django.contrib import admin
from videos.models import *
# Register your models here.

@admin.register(Maquina)
class MaquinaAdmin(admin.ModelAdmin):
    list_display = ('name',)
    list_filter = ('name',)

@admin.register(Fase)
class FaseAdmin(admin.ModelAdmin):
    list_display = ('name_phase', 'name_machine', 'num_ordre',)
    list_filter = ('name_phase', 'name_machine', 'num_ordre',)

@admin.register(Video)
class VideoAdmin(admin.ModelAdmin):
    list_display = ('name',)
    list_filter = ('name',)

@admin.register(Muntatge)
class MuntatgeAdmin(admin.ModelAdmin):
    list_display = ('id_maquina', 'id_fase', 'id_video', 'num_ordre',)
    list_filter = ('id_maquina', 'id_fase', 'id_video', 'num_ordre',)
```

4. base.html

```
{% load staticfiles %}

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <!-- Per posar la icona en el title -->
        <link rel="shortcut icon" href="{% static 'img/icon_ausa.png' %}">
        <title>AUSA Tutorials</title>
        <!-- Perque es vegi en dispositius mobils -->
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
            href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
        <link rel="stylesheet"
            href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">
        <!-- Per carregar els nostres estils -->
        <link rel="stylesheet" href="{% static 'css/style.css' %}">
        <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
            rel='stylesheet' type='text/css'>
        <script
            src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
        <script
            src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
    </head>
    <body>
      <div>
        <div class="navbar navbar-default navbar-static-top" id="wrapper">
          <div class="main">
            <div class="navbar-header">
              {% if user.is_authenticated %}
                <button type="button" data-target="#navbarCollapse"
                    data-toggle="collapse" class="navbar-toggle">
                  <span class="sr-only">Toggle navigation</span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                </button>
                {% endif %}
                <img class="logo" src="{% static 'img/ausa.jpg' %}" alt="ausa">
              </div>
       <!-- Coleccio de links que aniran dintre de la icone -->
              <div id="navbarCollapse" class="collapse navbar-collapse navbar-right">
                <ul class="nav navbar-nav ">
                  {% if user.is_authenticated %}
                    <form class="navbar-form navbar-left" role="search" action="{% url
                        'searchResults' %}">
                      <div class="form-group">
                        <input type="text" class="buscador" placeholder="Buscar"
                            name="mrname">
```

```
                  </div>
                </form>
                <li><a href="{% url 'machines_list' %}"><font
                    color="black">MAQUINES</font></a></li>
                <li><a href="{% url 'personal_data' %}"><font color="black">D.
                    PERSONALS</font></a></li>
                <li><a href="{% url 'logout' %}"><img id="logouti" src="{% static
                    'img/logout_icon.png' %}" style="width:24px; height:auto;
                    margin-right:5px;"></a></li>
                {% endif %}
              </ul>
            </div>
          </div>
      <div class="container-fluid">
        <div class="row">
              {% block section %} {% endblock %}
        </div>
      </div>
          <footer>
            {% if user.is_authenticated %}
                <p>Si tens qualsevol problema contacta amb el administrador clica <a
                    href="{% url 'contactar' %}"><font color="blue">aqui</font></a>
            {% endif %}
            <p>Copyright &copy; AUSA Center S.L.U., 2016 Copyright Holder All
                Rights Reserved</p>
          </footer>
        </div>
       </div>
    </body>
</html>
```

5. index.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
        <div id="main">
            <div id="formulari">
                <h1 class="title">Tutorials AUSA</h1>
                <br>
                {% if error %}
                  <small> {{error}} </small>
                {% endif %}
                <form method="post" action="">{% csrf_token %}
                    <input type="text" name="email" placeholder="Usuari" required="">
                    <input type="password" name="password" placeholder="Contrasenya"
                        required="">
                    <input type="hidden" name="next" value="{{ request.GET.next }}">
                    <input type="submit" value="Inicia Sessio">
                </form>
                <br>
                <p><a href="{% url 'contactar2' %}">Poblemes d'acces</a></p>
            </div>
        </div>
    </section>
{% endblock %}
```

6. maquines.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
<section>
  <div id="main2">
    <div class="panel-group" id="panel">
      <h4>
        <a href="{% url 'fases_list' %}"><font color="black">Fases</a>
      </h4>
    </div>
    <div class="panel-group" id="panel">
      <h4>
        <a href="{% url 'videos_list' %}"><font color="black">Videos</a>
      </h4>
    </div>
  </div>
  <div id="main1">
    {% if current_machines.count > 0 %}
      {% for maquina in current_machines %}
        <div class="maquina">
          <a href="{% url 'phases' maquina.pk %}" title="{{maquina.name}}">
            <img src={{maquina.image.url}} alt="{{maquina.name}}" class="image"></img>
          </a>
          <h3>{{maquina.name}}</h3>
        </div>
      {% endfor %}
    {% else %}
      <h2 style="text-align:center;">NO HI HA CAP MAQUINA PER MOSTRAR</h2>
    {% endif %}
  </div>
</section>
{% endblock %}
```

7. fases_maquina.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
      <div id="main">
        {% if fases.count == 0 %}
          {% for nom_maquina in maquina_actual %}
            <h1 class="title">{{nom_maquina.name}}</h1>
          {% endfor %}
          <h2 style="text-align:center;">AQUESTA MAQUINA NO TE CAP FASE</h2>
        {% else %}
          {% for nom_maquina in maquina_actual %}
            <h1 class="title">{{nom_maquina.name}}</h1>
              {% for fase in fases %}
                <a class="fase" href="{% url 'minivideos' fase.pk %}"><font size="4"
                    color="black">Fase {{fase.name_phase}}</font></a>
              {% endfor %}
          {% endfor %}
        {% endif %}
      </div>
    </section>
{% endblock %}
```

8. minivideos.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
<section>
  <div id="main">
    <h1 class="title">{{nom_maquina_actual}}, Fase {{nom_fase_actual}}</h1>
    {% if vid == 1 %}
      {% for element in videos_mostrar2 %}
        <div class="maquina">
          <a href="{% url 'video' element.id_muntatge %}"
              title="{{element.video.name}}">
            <img src={{element.video.photo.url}} alt="{{element.video.name}}"
                class="image"></img>
          </a>
          <h3>{{element.video.name}}</h3>
        </div>
      {% endfor %}
    {% else %}
      <h2 style="text-align:center;">NO HI HA CAP VIDEO EN AQUESTA FASE</h2>
    {% endif %}
  </div>
</section>
{% endblock %}
```

9. video.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
  <section>
    <!-- Part principal on van el video i els minivideos -->
      <div class="row" id="videos">
          <!-- Part on va el video -->
        <div class="col-md-6" id="video">
            <video controls autoplay muted class="caixa_video" id="my-video"
                width="98%" height="auto">
              {% for element in videos %}
                {% if element.id_muntatge == pagina_actual %}
                  <source src={{element.video.video.url}} type="video/mp4">
                {% endif %}
              {% endfor %}
            </video>
        </div>
        <!-- Part on van els minivideos -->
        <div class="col-md-6" id="minivideos">
          <div class="row" id="options">
            <div class="col-md-10 col-md-offset-2" id="playlist">Reproduccio
                Automatica <input id="reprauto" type="checkbox" value="false"></div>
          </div>
            <div class="row">
              {% for element in videos %}
                {% if element.id_muntatge == pagina_actual %}
                <div class="row" style="background-color:black;">
                  <div id="miniature" class="col-md-7">
                    <a href="./" class="currentvid">
                      <img  src={{element.video.photo.url}}
                          alt="{{element.video.name}}" class="img-responsive">
                    </a>
                  </div>
                  <div id="textminiature" class="col-md-5">
                    <a href="./" class="currentvid"><font
                        color="white">{{element.video.name}}</font></a>
                  </div>
                </div>
                {% else %}
                <div class="row">
                  <div id="miniature" class="col-md-7">
                    <a href="{% url 'video' element.id_muntatge %}" class="currentvid">
                      <img  src={{element.video.photo.url}}
                          alt="{{element.video.name}}" class="img-responsive">
                    </a>
                  </div>
                  <div id="textminiature" class="col-md-5">
                    <a href="{% url 'video' element.id_muntatge %}"
                        class="currentvid"><font
                        color="black">{{element.video.name}}</font></a>
```

```
                </div>
              </div>
              {% endif %}
            {% endfor %}
        </div>
      </div>
    </div>
  </section>
  <!-- script per fer la reproduccio automatica -->
      <script type="text/javascript">
        var videos = {{num_videos}};
        var vid = document.getElementById("my-video"); //Variable del video
        var repr = document.getElementById("reprauto"); //Variable de la opcio
            reproduccio automatica
        vid.onended = function() { //Funcio per saber quan acaba el video i passar al
            altre si la opcio de reproduccio automatica esta triada
          if (repr.checked){
            if ({{pagina_seguent}} != {{pagina_actual}}){
            localStorage.setItem("reprauto",$("#reprauto").is(":checked"));
            window.location.href="{% url 'video' pagina_seguent %}"
            }
          }
        };
        </script>
        <!-- Funcio per saber el estat del checkbox quan envia la pagina-->
        <script>
          $("#reprauto").change(function (event) {
            localStorage.setItem("reprauto",$("#reprauto").is(":checked"));
          });
          </script>
          <!-- script per saber el estat del checkbox quan la rep de una altre
              pagina-->
          <script>
          $(function() {
            if (localStorage.getItem("reprauto") == "true")
            $("#reprauto").prop("checked","true")
          });
          </script>
            <!-- Per fer que no et puguis descarregar els videos amb el boto dret -->
           <script type="text/javascript">
              document.oncontextmenu = new Function("return false");
           </script>
{% endblock %}
```

10. contactar2.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
        <div id="main">
          <div id="formulari">
              <h1 class="title">Contacta per qualsevol observacio, dubte o
                  problema</h1>
              {% for error in errors %}
              <small style="color: red"> {{error}}</small>
              {% endfor %}
              {% if send %}
              <small>El missatge sha enviat correctament</small>
              {% endif %}
              <br>
              <form method="post" action=""> {% csrf_token %}
                  <label>Nom</label>
                  <input type="text" name="name" placeholder="Nom" required="">
                  <label>Cognoms</label>
                  <input type="text" name="surname" placeholder="Cognoms" required="">
                  <label>Correu electronic</label>
                  <input type="text" name="email" placeholder="Email" required="">
                  <textarea type="text" name="text" rows="4" cols="67"></textarea>
                  <br>
                  <input type="submit" value="Envia">
              </form>
          </div>
        </div>
    </section>
{% endblock %}
```

11. link_fases.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
      <div id="main">
        {% if current_machines.count > 0 %}
          {% for maquina in current_machines %}
            <h1 class="title">{{maquina.name}}</h1>
            {% if current_fases.count > 0 %}
              {% for fase in current_fases %}
                {% if fase.name_machine_id == maquina.id %}
                  <a class="fase" href="{% url 'minivideos' fase.pk %}"><font size="4"
                      color="black">Fase {{fase.name_phase}}</font></a>
                {% endif %}
              {% endfor %}
            {% else %}
              <h4> No hi ha cap fase</h4>
            {% endif %}
          {% endfor %}
        {% else %}
          <h4> No hi ha cap maquina</h4>
        {% endif %}
      </div>
    </section>
{% endblock %}
```

12. link_videos.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
<section>
  <div id="main">
    {% if current_machines.count > 0 %}
      {% for maquina in current_machines %}
        <h1 class="title">{{maquina.name}}</h1>
        {% if current_fases.count > 0 %}
          {% for fase in current_fases %}
            <div class="row">
              <div id="links" class="col-md-12">
                {% if fase.name_machine_id == maquina.id %}
                  <h3>-- {{fase.name_phase}} --</h3>
                {% endif %}
                {% for muntatge in current_muntatges %}
                  {% if muntatge.id_maquina_id == maquina.id %}
                    {% if muntatge.id_fase_id == fase.id %}
                      {% for video in current_videos %}
                        {% if video.id == muntatge.id_video_id%}
                          <div class="maquina">
                            <a href="{% url 'video' muntatge.id %}"
                                title="{{video.name}}">
                              <img src={{video.photo.url}} alt="{{video.name}}"
                                  class="image"></img>
                            </a>
                            <h3>{{element.video.name}}</h3>
                          </div>
                        {% endif %}
                      {% endfor %}
                    {% endif %}
                  {% endif %}
                {% endfor %}
              </div>
            </div>
          {% endfor %}
        {% else %}
          <h4> No hi ha cap fase</h4>
        {% endif %}
      {% endfor %}
    {% else %}
      <h4> No hi ha cap maquina</h4>
    {% endif %}
  </div>
</section>
{% endblock %}
```

13. search.html

```
{% load staticfiles %}

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <!-- Per posar la icona en el title -->
    <link rel="shortcut icon" href="{% static 'img/icon_ausa.png' %}">
    <title>AUSA Tutorials</title>
    <!-- Perque es vegi en dispositius mobils -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">
    <!-- Per carregar els nostres estils -->
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
    <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
        rel='stylesheet' type='text/css'>
    <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script
        src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
  </head>
  <body>
    <div class="navbar navbar-default navbar-static-top" id="wrapper">
      <div class="main">
        <div class="navbar-header">
          <button type="button" data-target="#navbarCollapse" data-toggle="collapse"
              class="navbar-toggle">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <img src="{% static 'img/ausa.jpg' %}" alt="ausa" style="height: 35px;
              weight: auto;">
        </div>
        <!-- Coleccio de links que aniran dintre de la icone -->
        <div id="navbarCollapse" class="collapse navbar-collapse navbar-right">
          <ul class="nav navbar-nav ">
            <form class="navbar-form navbar-left" role="search" action="{% url
                'searchResults' %}">
              <div class="form-group">
                <input type="text" class="buscador" placeholder="Buscar"
                    name="mrname">
              </div>
            </form>
            <li><a href="{% url 'machines_list' %}"><font
```

```
            color="black">MAQUINES</font></a></li>
        <li><a href="{% url 'personal_data' %}"><font color="black">D.
            PERSONALS</font></a></li>
        <li><a href="{% url 'logout' %}"><img id="logouti" src="{% static
            'img/logout_icon.png' %}" style="width:24px; height:auto;
            margin-right:5px;"></a></li>
      </ul>
    </div>
  </div>
  {% if consultes == 1 %}
    <div id="main">
      <h2 style="text-align:center;">ERROR: CERCAR COM A MAXIM DUES PARAULES</h2>
    </div>
  {% else %}
    {% if m > 0 %}
      <div id="main">
        <h1>Maquines</h1>
        {% for maquina in maquines %}
          <div class="maquina">
            <a href="{% url 'phases' maquina.pk %}" title="{{maquina.name}}">
              <img src={{maquina.image.url}} alt="{{maquina.name}}"
                  class="image"></img>
            </a>
            <h3>{{maquina.name}}</h3>
          </div>
        {% endfor %}
      </div>
    {% endif %}
    {% if f > 0 %}
      <div id="main">
        <h1>Fases</h1>
        {% for element in ldf %}
          <div id="fases">
            <a class="fase" href="{% url 'minivideos' element.id_fase %}"><font
                size="4" color="black">Fase {{element.nom_fase}}</font></a>
            <p>Maquina: {{element.maquina}}</p>
          </div>
        {% endfor %}
      </div>
    {% endif %}
    {% if v > 0 %}
      <div id="main">
        <h1>Videos</h1>
        {% for element in ldv %}
          <div class="maquina">
            <a href="{% url 'video' element.id_muntatge %}"
                title="{{element.video.name}}">
              <img src={{element.video.photo.url}} alt="{{element.video.name}}"
                  class="image"></img>
            </a>
            <h3>{{element.video.name}}</h3>
            <p>Maquina: {{element.maquina}}</p>
            <p>Fase: {{element.fase}}</p>
```

```
            </div>
          {% endfor %}
        </div>
      {% else %}
        {% if f == 0 %}
          {% if m == 0 %}
            <div id="main">
              <h2 style="text-align:center;">CAP RESULTAT PER MOSTRAR</h2>
            </div>
          {% endif %}
        {% endif %}
      {% endif %}
      <footer>
        {% if user.is_authenticated %}
          <p>Si tens qualsevol problema contacta amb el administrador <a href="{% url
              'contactar' %}"><font color="blue">aqui</font></a>
        {% endif %}
        <p>Copyright &copy; AUSA Center S.L.U., 2016 Copyright Holder All Rights
            Reserved</p>
      </footer>
    </div>
  </body>
</html>
```

14. contactar.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
        <div id="main">
          <div id="formulari">
              <h1 class="title">Contacta per qualsevol observacio, dubte o
                  problema</h1>
              {% for error in errors %}
              <small style="color: red"> {{error}}</small>
              {% endfor %}
              {% if send %}
              <small>El missatge sha enviat correctament</small>
              {% endif %}
              <br>
              <form method="post" action=""> {% csrf_token %}
                  <label>Nom</label>
                  <input style="background-color: #F0F0F0" type="text" name="name"
                      placeholder="" required="" value="{{nom_treballador}}"
                      readonly="readonly">
                  <label>Cognoms</label>
                  <input style="background-color: #F0F0F0" type="text" name="surname"
                      placeholder="" required="" value="{{cognom_treballador}}"
                      readonly="readonly">
                  <label>Correu electronic</label>
                  <input style="background-color: #F0F0F0" type="email" name="email"
                      placeholder="" required="" value="{{user.email}}"
                      readonly="readonly">
                  <textarea type="text" name="text" rows="4" cols="67"></textarea>
                  <input type="submit" value="Envia">
              </form>
          </div>
        </div>
    </section>
{% endblock %}
```

15. dades_personals.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
        <div id="main">
            <div id="dades">
                <h1 class="title">Dades Personals </h1>
                <div>
                    <p><strong>Nom: </strong>{{user.first_name}}</p>
                    <p><strong>Cognoms: </strong> {{user.last_name}}</p>
                    <p><strong>email: </strong> {{ user.email }}</p>
                </div>
            </div>
        </div>
        <p style="text-align:center">Si vols modificar alguna dada personal o de
            acces <a href="{% url 'modify_data' %}"><font color="blue">clica
            aqui</font></a></p>
    </section>
{% endblock %}
```

16. modify_data.html

```
{% extends 'include/base.html' %}
{% load staticfiles %}
{% block section %}
    <section>
        <div id="main">
            <div id="formulari">
                <h1 class="title">Modifica les dades</h1>
                {% if error %}
                <small style="color: red"> {{ error }} </small>
                {% endif %}
                <br>
                <form method="post" action="">{% csrf_token %}
                    <h2>Dades Personals</h2>
                    <br>
                    <label>Nom</label>
                    <input type="text" name="name" placeholder="Nom" required=""
                        value="{{user.first_name}}">
                    <label>Cognoms</label>
                    <input type="text" name="surname" placeholder="Cognom" required=""
                        value="{{user.last_name}}">
                    <label>EMAIL</label>
                    <input type="text" name="email" placeholder="email" required=""
                        value="{{user.email}}">
                    <h2>Compte de usuari</h2>
                    <label>Correu electronic</label>
                    <input style="background-color: #F0F0F0" type="email"
                        name="username" placeholder="Correu electronic" required=""
                        value="{{user.username}}" readonly="readonly">
                    <!--<input type="password" placeholder="Contrasenya actual"
                        required="" value="abcdefg">-->
                    <label>Contrasenya</label>
                    <input type="password" name="new_password" placeholder="Nova
                        contrasenya" value="">
                    <label>Repeteix Contrasenya</label>
                    <input type="password" name="repeat_new_password"
                        placeholder="Repeteix la nova contrasenya" value="">
                    <br>
                    <input type="submit">
                </form>
            </div>
        </div>
    </section>
{% endblock %}
```

17. views.py de tutorials

```
# -*- coding: utf-8 -*-
from django.views.generic import TemplateView
from django.conf import settings
from django.core.mail import send_mail, BadHeaderError, EmailMessage
from django.core.urlresolvers import reverse
from django.http import HttpResponse, HttpResponseRedirect
from django.core.mail import EmailMultiAlternatives
import smtplib
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render
from forms import *
from django.conf import settings
#Per fer que les pagines estiguin restringides
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator


class HomeView(TemplateView):
    template_name = 'index.html'

    def get(self, request, *args, **kwargs):
        return render(request, 'index.html')

    def post(self, request, *args, **kwargs):
        email = request.POST['email']
        password = request.POST['password']
        user = authenticate(username=email, password=password)

        # Si l'usuari existeix
        if user is not None:
            # Si el usuari esta actiu
            if user.is_active:
                login(request, user)
                redirection = reverse('machines_list')
                try:
                    if request.GET["next"]:
                        redirection = request.GET["next"]
                except:
                    pass
                return HttpResponseRedirect(redirection)
            else:
                return render(request, 'index.html', {'error': 'Compte no actiu:
                    Contactar amb el gestor de la web'})
        else:
            return render(request, 'index.html', {'error': 'Usuari i/o contrasenya
                incorrectes'})

class Logout(TemplateView):
    # Logout and redirect
    def get(self, request, *args, **kwargs):
```

```
        logout(request)
        return HttpResponseRedirect('/en/login/')



#Per problemes quan estas autentificat
class ContactView(TemplateView):
    template_name = 'contactar.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        user=request.user
        nom_treballador = user.first_name
        cognom_treballador = user.last_name
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'contactar.html', {'nom_treballador': nom_treballador,
            'cognom_treballador':cognom_treballador,})

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data["name"]
            surname = form.cleaned_data["surname"]
            email = form.cleaned_data["email"]
            text = form.cleaned_data["text"]

            mes = "Nou missatge"
            mes += "\n Nom: %s" % name
            mes += "\n Cognoms: %s" % surname
            mes += "\n Email: %s" % email
            mes += "\n Text: %s" % text

            server = smtplib.SMTP('smtp.gmail.com',587)
            server.starttls()
            server.login(settings.EMAIL, settings.PASSWORD)
            server.sendmail(email, settings.EMAIL, mes)
            server.quit()
            return render(request, 'contactar.html', {'send':True,'nom_treballador':
                name, 'cognom_treballador':surname,})
        else:
            errors = []
            for field in form:
                for error in field.errors:
                    errors += error
            return render(request, 'contactar.html', {'errors': errors,})


#Per problemes de acces quan no estas autentificat
class Contact2View(TemplateView):
    template_name = 'contactar2.html'

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
```

```
if form.is_valid():
    name = form.cleaned_data["name"]
    surname = form.cleaned_data["surname"]
    email = form.cleaned_data["email"]
    text = form.cleaned_data["text"]

    mes = "Nou missatge"
    mes += "\n Nom: %s" % name
    mes += "\n Cognoms: %s" % surname
    mes += "\n Email: %s" % email
    mes += "\n Text: %s" % text

    server = smtplib.SMTP('smtp.gmail.com',587)
    server.starttls()
    server.login(settings.EMAIL, settings.PASSWORD)
    server.sendmail(email, settings.EMAIL, mes)
    server.quit()
    return render(request, 'contactar2.html', {'send':True,})
```

18. forms.py de tutorials

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(max_length=128, required=True)
    surname = forms.CharField(max_length=128, required=True)
    email = forms.EmailField(required=True)
    text = forms.CharField(max_length=512, required=True)
```

19. urls.py de tutorials

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
from django.conf.urls.i18n import i18n_patterns
from django.views.generic import TemplateView
from django.conf.urls.static import static
# importem totes les views de totes les aplicacions
from tutorials.views import *
from videos.views import *


urlpatterns = patterns(
    '',
    url(r'^i18n/', include('django.conf.urls.i18n')),
)


urlpatterns += i18n_patterns('',
    # Examples:
    # url(r'^$', 'gaitanalysis.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    #(r'^admin/filebrowser/', include(site.urls)),

    (r'^grappelli/', include('grappelli.urls')),
    url(r'^admin/', include(admin.site.urls)),

    #pagina principal
    url(r'^login/', HomeView.as_view(), name='home'),
    url(r'^tutorials/', include('videos.urls')),

    url(r'^contactar/', ContactView.as_view(), name='contactar'),
    url(r'^contactar2/', Contact2View.as_view(), name='contactar2'),
    url(r'^logout/', Logout.as_view(), name='logout'),
) + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

20. views.py de videos

```
# -*- coding: utf-8 -*-
import json
#per fer el buscador
import operator
from django.shortcuts import render

from django.core import serializers
from django.views.generic import DetailView, ListView, TemplateView
from django.shortcuts import render_to_response
from django.template import RequestContext
from django.core.urlresolvers import reverse
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse

from datetime import datetime

from django.contrib.auth import authenticate, login
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect
from django.shortcuts import render
#per fer el buscador
from django.db.models import Q
#Per fer que les pagines estiguin restringides
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator


#from treballadors.models import *
from videos.models import *
from tutorials.urls import *

from django.views.decorators.csrf import csrf_exempt

#Pagina principal
class MachinesDataView(TemplateView):
    template_name = 'maquines.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        #Fer una llista que contingui la llista de videos i muntatge per recorre-la a
            la vegada
        return render(request, 'maquines.html', {'current_machines':
            current_machines})

#Pagina per quan es clica el link de Fases en la pagina principal. Mostra totes les
    maquines i les fases que te cada una
class PhasesLinkView(TemplateView):
    template_name = 'link_fases.html'

    @method_decorator(login_required)
```

```
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        current_fases = Fase.objects.all().order_by('num_ordre')
        return render(request, 'link_fases.html',
            {'current_machines':current_machines, 'current_fases':current_fases})

#Pagina per quan es el link de Videos en la pagina principal. Mostra tots els videos
    que formen cada fase
class VideosLinkView(TemplateView):
    template_name = 'link_videos.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_machines = Maquina.objects.all()
        current_fases = Fase.objects.all().order_by('num_ordre')
        current_videos = Video.objects.all()
        current_muntatges = Muntatge.objects.all().order_by('num_ordre')
        return render(request, 'link_videos.html',
            {'current_machines':current_machines, 'current_fases':current_fases,
            'current_videos': current_videos, 'current_muntatges': current_muntatges})

#Pagina quan es clica una maquina, on es mostren les fases que la formen
class PhasesDataView(TemplateView):
    template_name = 'fases_maquina.html'
    context_object_name = 'maquina_actual'
    model = Maquina

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        pk_actual = int(self.kwargs['pk'])
        maquina_actual = Maquina.objects.filter(id=pk_actual)
        #Per obtenir els noms de les fases
        fases = Fase.objects.filter(name_machine_id=pk_actual).order_by('num_ordre')
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'fases_maquina.html', {'fases': fases,
            'maquina_actual':maquina_actual})

#Pagina on es mostren els videos que formen cada fase
class MiniVideosDataView(TemplateView):
    template_name = 'minivideos.html'
    context_object_name = 'minivideos'
    model = Video

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        pk_actual = int(self.kwargs['pk'])
        nom_fase_actual =
            str(Fase.objects.filter(id=pk_actual).values('name_phase')[0]['name_phase'])
        id_fase = str(Fase.objects.filter(id=pk_actual).values('id')[0]['id'])
        id_maquina =
            str(Fase.objects.filter(id=pk_actual).values('name_machine_id')[0]['name_machine_id'])
        nom_maquina_actual =
            Maquina.objects.filter(id=id_maquina).values('name')[0]['name']
```

```
        #Obtenim els id dels videos que hem de mostrar, pero en un diccionari
        id_videos = Muntatge.objects.filter(id_maquina=id_maquina,
            id_fase=id_fase).values('id_video').order_by('num_ordre')
        #Obtenim dels id dels muntatges en que es troben els videos que hem de
            mostrar per poder passar a la pagina de video correcte
        id_muntatges = Muntatge.objects.filter(id_maquina=id_maquina,
            id_fase=id_fase).values('id').order_by('num_ordre')
        #Agafem els id dels videos que hem de mostar i els posem en una llista
        x = 0
        id_videos2 = []
        for video in id_videos:
            id_videos2.insert(x, id_videos[x]['id_video'])
            x += 1
        #Agafem els id dels muntatges que hem de mostrar i els posem en una llista
        z = 0
        id_muntatges2 = []
        for muntatge in id_muntatges:
            id_muntatges2.insert(z, id_muntatges[z]['id'])
            z += 1
        # La variable vid serveix al template per saber si hi ha videos per mostrar o
            no
        y = 0
        vid = 0
        videos_mostrar = []
        while (y < len(id_videos2)):
            videos_mostrar.insert(y, Video.objects.filter(id=id_videos2[y])[0])
            vid = 1
            y += 1
        #Llista de diccionaris on passem el id del muntatge i el video que s'ha de
            mostrar per cada link
        videos_mostrar2 = []
        z = 0
        for x,y in zip(id_muntatges2, videos_mostrar):
            d = {}
            d['id_muntatge'] = x
            d['video'] = y
            videos_mostrar2.insert(z, d)
            z += 1
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'minivideos.html', {'vid':vid,
            'videos_mostrar2':videos_mostrar2, 'nom_maquina_actual':
            nom_maquina_actual, 'nom_fase_actual':nom_fase_actual})

#Aqui posar el template de reproduccio de video
class VideoDataView(TemplateView):
    template_name = 'video.html'


    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        current_pk = int(self.kwargs['pk'])
        #Per saber en el muntatge que volem reproduir el video, i saber la maquina i
            la fase on es troba
```

```
muntatge = Muntatge.objects.filter(id=current_pk)
#Obtenim la fase del video que hem de reproduir
fase =
    str(Muntatge.objects.filter(id=current_pk).values('id_fase_id')[0]['id_fase_id'])
#Obtenim la maquina del video que hem dereproduir
maquina =
    str(Muntatge.objects.filter(id=current_pk).values('id_maquina_id')[0]['id_maquina_id'])
#Per obtenir els id dels videos que hem de mostrar, pero en diccionari
id_videos = Muntatge.objects.filter(id_maquina=maquina,
    id_fase=fase).values('id_video').order_by('num_ordre')
#Agafem els id dels videos que hem de mostar i els posem en una llista
x = 0
id_videos2 = []
for video in id_videos:
    id_videos2.insert(x, id_videos[x]['id_video'])
    x += 1
#Agafem els id dels muntatges que s'han de mostrar i els posem en una llista
z = 0
id_muntatges2 = []
for i in id_videos2:
    m = Muntatge.objects.filter(id_video=i,
        id_fase_id=fase).values('id')[0]['id']
    id_muntatges2.insert(z, m)
    z += 1
#Agafem els videos que van en aquella fase en el ordre correcte
y = 0
videos_mostrar = []
while (y < len(id_videos2)):
    videos_mostrar.insert(y, Video.objects.filter(id=id_videos2[y])[0])
    vid = 1
    y += 1
#Numero que serveix per fer la reproduccio automatica
num_videos = len(id_muntatges2)
current_position = id_muntatges2.index(current_pk)
#Per agafar la posicio de la llista de pk al qual tenim d'anar a continuacio
position_seguent_pk = id_muntatges2.index(current_pk) + 1
#Si el seguent pk no existeix
if (current_position >= (len(id_muntatges2))-1):
    next_pk = current_pk
#Si existeix seguent pk
else:
    next_pk = id_muntatges2[position_seguent_pk]
#Adjunto les llistes id_muntatges2 i videos_mostrar per poder posar a
    reproduir el video que toca
i2 = 0
videos = []
for x,y in zip(id_muntatges2, videos_mostrar):
    d = {}
    d['id_muntatge'] = x
    d['video'] = y
    videos.insert(i2, d)
    i2 += 1
#Agafo el video que hem de reproduir, i agafo les seves instruccions per
```

```
            mostrar-les en el template
        i = videos_mostrar[current_position].instruccions
        instruccions = i.split('\r\n')
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'video.html', {'videos':videos, 'num_videos':
            num_videos, 'pagina_actual': current_pk, 'pagina_seguent': next_pk,
            'current_position': current_position, 'instruccions': instruccions})


#Funcio per fer el buscador, busca maquines, fases i videos
class SearchResultsView(TemplateView):
    template_name = 'search.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        mrname = request.GET.get('mrname')
        #Llista on guardo totes les paraules que ha introduit l'usuari per buscar-les
        mrname_list = mrname.split(',')
        #Llista on guardo les maquines per mostrar al template
        maquines = []
        #Per saber si hi ha mes d'una maquina en el template
        m = 0
        #Llista on guardo les fases per mostrar al template
        fases = []
        #Per saber si hi ha mes d'una fase en el template
        f = 0
        ldf = []
        #Llista on guardo els videos per mostrar al template
        videos = []
        v = 0
        ldv = []
        #Variable per comprovar que no es passen mes de dues consultes a la vegada.
            Si passa aixo, en el template surtira el misstage d'error de masses
            variables
        consultes = 0
        variables_buscades = len(mrname_list)
        if variables_buscades > 2:
            consultes = 1
        #Si la variable consultes es 1, surt error perque nomes es podran cercar com
            a maxim dues paraules a la vegada. S
        if consultes == 1:
            return render(request, 'search.html', {'consultes':consultes})
        else:
            if variables_buscades == 2:
                if mrname_list[0] == mrname_list[1]:
                    del(mrname_list [0])
        for mrname in mrname_list:
            #Consulta per mostar les maquines que coincideixen amb la cerca
            maquines += Maquina.objects.filter(name__contains=mrname)
            #Consulta per buscar les fases que coincideixen amb la cerca
            fases += Fase.objects.filter(name_phase__contains=mrname)
            fases_final = Fase.objects.filter(name_phase__contains=mrname)
            #Fa el mateix que la consulta de dalt, pero ens servira per trobar el nom
```

```
    que correspon a cada maquina, no per buscar les fases
fases2 = Fase.objects.filter(name_phase__contains=mrname).values()
#Llista del id de les fases que ens coincideixen amb la cerca, en format
    diccionari
id_fases_d = Fase.objects.filter(name_phase__contains=mrname).values('id')
s = 0
#Llista on tenim nomes els id
id_fases = []
for e in id_fases_d:
    id_fases.insert(s, id_fases_d[s]['id'])
    s += 1
#Llista de diccionaris que ens servira per passar junts la fase i el nom
    de la maquina on es troba
ld = [{'x':1,'y':2}]
id_maquines = []
nom_maquines = []
x = 0
if len(fases2) > 0:
    for fase2 in fases2:
        id_maquines.insert(x, fases2[x]['name_machine_id'])
        x += 1
    if len(id_maquines) > 0:
        y = 0
        for id_maquina in id_maquines:
            nom =
                str(Maquina.objects.filter(id=id_maquina).values('name')[0]['name'])
            nom_maquines.insert(y, nom)
            y += 1
#Llista de diccionaris on relacionarem la fase amb el nom de la maquina
    on es troba
index_ldf = 0
for x,y,w in zip(id_fases, fases_final, nom_maquines):
    d={}
    d['nom_fase'] = y
    d['id_fase'] = x
    d['maquina'] = w
    ldf.insert(index_ldf, d)
    index_ldf += 1
#Consulta per mostar els videos que coincideixen amb la cerca
videos += Video.objects.filter(name__contains=mrname)
#Ens quedem amb els id dels videos per buscar les fases i maquines a les
    que pertany. En format diccionari
videos2 = Video.objects.filter(name__contains=mrname).values('id')
#Llista on hi ha guardat els numeros dels id dels videos en el format
    correcte
videos_id = []
#Per trobar tots els muntatges que coincideixen amb la cerca
muntatgesv = []
#Llista on hi guardo els id dels muntatges dels videos que coincideixen
    amb la cerca, en diccionari
muntatgesv_id = []
#Llista on hi guardo els id dels muntatges dels videos que coincideixen
    amb la cerca, en format correcte
```

```
muntatgesv_id2 = []
muntatgesf = []
muntatgesm = []
#Llista amb tots els id dels videos que es troben en els muntatges
videos_id2 = []
videos_name = []
#Llista on hi ha guardat els numeros de id de cada maquina a la que
    pertany cada video
maquines_id = []
maquines_name = []
#Llista on hi ha guardat els numeros de id de cada fase a la que pertany
    cada video
fases_id = []
fases_name = []
#Llista de diccionaris on relacionem els videos amb les fases i les
    maquines on es troben
x = 0
if len(videos2) > 0:
    for video2 in videos2:
        videos_id.insert(x, videos2[x]['id'])
        x += 1
    #Per saber els id dels videos, fases i maquines que tenen d'apareixer
        en la cerca, en format diccionari
    for i in videos_id:
        muntatgesv +=
            Muntatge.objects.filter(id_video_id=i).values('id_video_id')
        muntatgesv_id +=
            Muntatge.objects.filter(id_video_id=i).values('id')
        muntatgesf +=
            Muntatge.objects.filter(id_video_id=i).values('id_fase_id')
        muntatgesm +=
            Muntatge.objects.filter(id_video_id=i).values('id_maquina_id')
    #Per obtenir els id dels muntatges en una llista
    i = 0
    for e in muntatgesv_id:
        muntatgesv_id2.insert(i, muntatgesv_id[i]['id'])
        i += 1
    #Per obtenir els id dels videos en una llista
    i2 = 0
    for e in muntatgesv:
        videos_id2.insert(i2, muntatgesv[i2]['id_video_id'])
        i2 += 1
    #Per obtenir els id de les fases en una llista
    i3 = 0
    for e in muntatgesf:
        fases_id.insert(i3, muntatgesf[i3]['id_fase_id'])
        i3 += 1
    #Per obtenir els id de les maquines en una llista
    i4 = 0
    for e in muntatgesm:
        maquines_id.insert(i4, muntatgesm[i4]['id_maquina_id'])
        i4 += 1
    #Recorro les llistes anteriors per aconsegui el nom de cada id de
```

```
                maquina,de fase i de video i fer tres llistes noves
            y = 0
            for index in maquines_id:
                maquina =
                    str(Maquina.objects.filter(id=maquines_id[y]).values('name')[0]['name'])
                maquines_name.insert(y, maquina)
                y += 1
            y = 0
            for index2 in fases_id:
                fase =
                    str(Fase.objects.filter(id=fases_id[y]).values('name_phase')[0]['name_phase'])
                fases_name.insert(y, fase)
                y += 1
            y = 0
            for index3 in videos_id2:
                video = Video.objects.filter(id=videos_id2[y])[0]
                videos_name.insert(y, video)
                y += 1
            #Llista de diccionaris on relacionem els videos amb les fases i les
                maquines on es troben
            z = 0
            for v,w,x,y in zip(muntatgesv_id2, videos_name, maquines_name,
                fases_name):
                d={}
                d['id_muntatge'] = v
                d['video'] = w
                d['maquina'] = x
                d['fase'] = y
                ldv.insert(z,d)
                z += 1
        if len(maquines) > 0:
            m = 1
        if len(fases) > 0:
            f = 1
        if len(videos) > 0:
            v = 1
        #Li passo els objectes al html per poder mostrar-los
        return render(request, 'search.html', {'maquines':maquines, 'm':m,
            'fases':fases, 'f':f, 'ldf':ldf, 'videos':videos, 'ldv':ldv, 'v':v})

#Per mostrar les dades personals del treballador autentificat
class PersonalDataView(TemplateView):
    template_name = 'dades_personals.html'

    @method_decorator(login_required)
    def get(self, request, *args, **kwargs):
        return render(request, 'dades_personals.html', {})


#Perque l'usuari pugui modificar el seu email personal i la seva contrasenya
class ModifyDataView(TemplateView):
    template_name = 'modify_data.html'
```

```
@method_decorator(login_required)
def get(self, request, *args, **kwargs):
    return render(request, 'modify_data.html', {})

def post(self, request, *args, **kwargs):
    user = self.request.user
    name = request.POST["name"]
    surname = request.POST["surname"]
    email = request.POST["email"]
    new_password = request.POST["new_password"]
    repeat_new_password = request.POST["repeat_new_password"]
    if name!="" and surname!="" and email!="":
        user.first_name = name
        user.last_name = surname
        user.email = email
        user.save()
        redirection = reverse('personal_data')
        if new_password!="":
            if repeat_new_password!="":
                if repeat_new_password==new_password:
                    user.set_password(new_password)
                    user.save()
                    user = authenticate(username=user.username,
                        password=new_password)
                    login(request, user)
                else:
                    error = "Les contrasenyes no coincideixen"
                    return render(request, 'modify_data.html', {'error':error,})
            else:
                error = "Tornar a repetir contrasenya"
                return render(request, 'modify_data.html', {'error':error,})

        return HttpResponseRedirect(redirection)

    else:
        error = "Omplir tots els camps obligatoris"
        return render(request, 'modify_data.html', {'error':error,})
```

21. urls.py de videos

```
# -*- encoding: utf-8 -*-

from django.conf.urls import patterns, include, url
from videos.views import *
from . import views

urlpatterns = patterns(
    '',
    #Llistat on hi ha totes les maquines que podem veure el tutorial
    url(r'^machines/$', MachinesDataView.as_view(), name='machines_list'),
    #Patrons per les links de la pagina principal
    url(r'^phases/$', PhasesLinkView.as_view(), name='fases_list'),
    url(r'^videos/$', VideosLinkView.as_view(), name='videos_list'),
    #Patrons per mostrar el videos relacionats amb la maquina
    url(r'^machine-phases/(?P<pk>[-_\w]+)/$', PhasesDataView.as_view(),
        name='phases'),
    url(r'^phases-parts/(?P<pk>[-_\w]+)/$', MiniVideosDataView.as_view(),
        name='minivideos'),
    url(r'^video/(?P<pk>[-_\w]+)/$', VideoDataView.as_view(), name='video'),
    #Patro per quan es fa una cerca
    url(r'^search/$', SearchResultsView.as_view(), name='searchResults'),
    #Patrons per les dades personals dels usuaris
    url(r'^modify-data/$', ModifyDataView.as_view(), name='modify_data'),
    url(r'^personal-data/$', PersonalDataView.as_view(), name='personal_data'),
)
```

22. ausassl.conf

```
<IfModule mod_ssl.c>
    <VirtualHost *:443> #Canviar la ip per la del servername
            ServerName 10.192.35.245
            ServerAdmin danivaro35@gmail.com

            DocumentRoot /var/www/html/tutorials

             ErrorLog ${APACHE_LOG_DIR}/error.log
           LogLevel warn
           CustomLog ${APACHE_LOG_DIR}/custom.log combined

           AliasMatch ^/([^/]*\.css) /var/www/html/tutorials/wsgi/static/css/$1
           Alias /static/ /var/www/html/tutorials/wsgi/static/
           Alias /media/ /var/www/html/tutorials/wsgi/media/

           <Directory /var/www/html/tutorials>
                        Order allow,deny
                        Allow from all
              </Directory>

           <Directory /var/www/html/tutorials/wsgi/media/videos>
              Order allow,deny
              Allow from all
              </Directory>


           WSGIScriptAlias / /var/www/html/tutorials/tutorials/wsgi.py

           SSLEngine on
           SSLProtocol all
           SSLCertificateFile /etc/apache2/ssl/ausa.crt
           SSLCertificateKeyFile /etc/apache2/ssl/ausa.key

           <FilesMatch "\.(cgi|shtml|phtml|php)$">
               SSLOptions +StdEnvVars
           </FilesMatch>


           <Directory /usr/lib/cgi-bin>
              SSLOptions +StdEnvVars
           </Directory>

           BrowserMatch "MSIE [2-6]" \
           nokeepalive ssl-unclean-shutdown \
           downgrade-1.0 force-response-1.0
           BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
         </VirtualHost>
</IfModule>
```