TREBALL FINAL

Fustot v3.0

Sergi Beltran Prat
Grau de Multimèdia
Àrea d'Enginyeria Web

Ignasi Lorente Puchades
Carlos Casado Martinez

Data de lliurament: 12/06/2017

FITXA DEL TREBALL FINAL

Títol del treball	Fustot v3.0
Nom de l'autor	Sergi Beltran Prat
Nom del consultor/a:	Ignasi Lorente Puchades
Nom del PRA	Carlos Casado Martínez
Data de lliurament	06/2017
Titulació o programa	Grau de Multimèdia
Àrea del Treball Final	Àrea d'Enginyeria Web
Idioma del treball	Català
Paraules clau	CMS, Foundation, Rails

Figura 1 - Fitxa del treball

Resum del treball

El treball final ha consistit en desenvolupar la pàgina web d'una empresa distribuïdora de fustes, anomenada Fustot.

La pàgina o aplicació web és capaç de gestionar el seu contingut, com per exemple, la creació o eliminació de productes, afegir imatges, vídeo, canviar descripcions, etcètera. Un cop realitzats aquests canvis s'han de poder veure a la web pública a la que usuaris accedeixen.

El disseny ha partit de la base de l'actual web www.fustot.cat, però tota la part de front s'ha desenvolupat de nou usant eines tan útils com SASS, un preprocessador molt popular de CSS. Aquesta eina té com a principal avantatge la possibilitat de convertir els CSS en un llenguatge més dinàmic. Permet treballar molt més ràpidament en la creació de codi amb la possibilitat de crear funcions que realitzin certes operacions matemàtiques i reutilitzar el codi gràcies als mixins, variables que ens permeten guardar valors.

Tant el disseny visual, el disseny de l'arquitectura del desenvolupament frontend com backend han començat des de zero, cal dir però, que, s'han usat principalment dos frameworks que aporten una base sòlida en la qual podrem començar: Foundation 6 (http://foundation.zurb.com/sites.html) per la part del frontend i Ruby on Rails 5 (http://rubyonrails.org) per la banda del backend.

També s'ha usat diverses llibreries que ens han ajudat en el procés de desenvolupament com per exemple "PaperClip" que facilita el procés de pujar imatges i assignar-la a un objecte Active Records de Ruby, o "Capybara" que ens ajuda a realitzar tests en un navegador virtual.

Concloent, en el treball s'ha intentat tocar una mica de tot el que s'ha après durant el grau, com pot ser dissenyar l'arquitectura de software, el disseny de la base de dades, el disseny de classes, el disseny de la interfície, desenvolupament front-end i back-end, la realització de tests i usar un gestor de versions com git.

L'experiència ha estat molt satisfactòria, el projecte entregat acaba aquí, però el seu desenvolupament seguirà el seu curs amb les noves funcionalitat que requereixin des dels administradors de Fustot.

Abstract

The final project has been develop a distributor website of wood, called Fustot.

The web page or application is able to manage its content, such as the creation or removal of products, add images, video, change descriptions, and so on. After making these changes must be seen in the public web users access.

The design has on the basis of the current web www.fustot.cat changing necessary elements such as "header" or "footer" and all the rest has been developed using new tools as useful as SASS, a popular CSS preprocessor.

Both the visual design, architectural design and frontend development backend started from scratch, but saying that, we have used two main frameworks that already provide a solid basis on which we started: Foundation 6 (http://foundation.zurb.com/sites.html) for the frontend Ruby on Rails and 5 (http://rubyonrails.org) from the side of the backend.

It has also been used several libraries that have helped in the development process such as "Paperclip" that facilitates the process of uploading images and assign it to an object in Active Records Ruby or "Capybara" helps us make test in a virtual browser.

Concluding, work has tried to play a little of everything that has been learned during the degree such as designing the software architecture, the design of the database design classes, designing interface development front-end and back-end, conducting tests and using git as a version control.

The experience has been very satisfactory, the project has delivered finished here, but once delivered will continue its development with new features that the company requires.

ÍNDEX GENERAL

ÍNDEX DE TAULES I FIGURES

1. INTRODUCCIÓ

1.1 Context i justificació del Treball

L'empresa Fustot requereix una millora en el seu web, el principal motiu és que no estan satisfets en com es gestiona el seu contingut. L'actual CMS (*Content Management System*) és Wordpress, un sistema que mai els ha acabat de fer el pes. Els inconvenients que hi veuen són varis:

La interfície és massa complexa, amb elements i funcionalitats que no han utilitzat mai i que no faciliten el manteniment del web. Cal dir que aquest manteniment és esporàdic, no accedeixen al Panell de Control de Wordpress diàriament i per això no arriben a memoritzar on trobar o com fer certes coses. Fustot.cat requereix un panell d'administració amb només aquells components útils per a ells, amb una interfície clara que els ajudi a ser més eficients quan calgui canviar el contingut.

A l'hora de gestionar alguns apartats, com per exemple, els preus dels productes, es troben que han de duplicar la seva feina de manera innecessària. Per posar un exemple, quan han de canviar preus han de fer-ho tantes vegades com idiomes hi ha en el web. Les taules de preus són continguts estàtics creats en el mateix Wordpress que es creen de manera manual per cada idioma.

Plugins, actualitzacions...Els gestors de fustot.cat no en volen saber res de tot això, per a ells, cada notificació d'actualització del mateix Wordpress o de plugins els suposa un mal de cap perquè no estan segurs de si és convenient actualitzar, ja que en episodis anteriors es va actualitzar Wordpress i van deixar de funcionar alguns dels plugins.

Amb aquest nou projecte es vol posar solució a totes les necessitats de fustot.cat amb una web totalment personalitzada, construïda d'una manera modular i escalable, preparada per afegir-hi funcionalitats de manera fàcil i que eviti trobar-nos en els problemes del present.

1.2 Objectius del projecte

- Millorar la productivitat i la facilitat a l'hora de gestionar els continguts del web.
- Millorar la interfície, que aquesta sigui clara i simple, que faciliti la seva navegació.
- Construir un lloc web modular, de fàcil manteniment, que es pugui anar ampliant funcionalitats de manera independent sense afectar a la resta de mòduls.
- Que els administradors puguin fer un CRUD dels productes.
- Possibilitat d'afegir vídeos als productes.
- Possibilitat d'afegir galeria d'imatges als productes.
- La web disponible en dos idiomes.

1.3 Enfocament i mètode seguit

Hi han diverses maneres d'afrontar aquest projecte, aquí en mencionarem 3 i argumentarem la solució triada.

1.3.1 Continuar amb Wordpress i creació de pluguin personalitzat per fustot.cat

Una opció era la de desinstal·lar tots els plugins de Wordpress actualment instal·lats a fustot.cat i crear-ne un personalitzat amb les necessitats esmentades anteriorment. Amb aquesta solució es resoldria en part la simplificació de la interfície i la facilitat d'ús en la gestió de continguts, però es mantindria en general el look&feel a l'hora de gestionar la web.

Personalment considero que basar tota la infraestructura d'una web a través d'un pluguin o varis no és la millor manera de desenvolupar un projecte personalitzat com el que es vol.

Tampoc em satisfà la manera com es desenvolupa Wordpress, penso que un model de desenvolupament MVC (Model-Vista-Controlador) ofereix la possibilitat de tenir una infraestructura més modular i fàcil de mantenir.

1.3.2 Usar una plataforma enfocada al comerç electrònic com Prestashop o Magento

Utilitzar una plataforma enfocada clarament al ecommerce ens facilitaria la feina en quant a la creació de productes de venda directa i la seva gestió, però fustot.cat no és un ecommerce a l'ús, amb això vull dir que la venda de productes directes és una minoria, la majoria d'aquests només es vol mostrar informació de preu i mides, perquè són productes de grans volums i més enfocats als industrials, i que, des de Fustot volen que el procés de venda d'aquest sigui més manual.

1.3.3 Creació de web des de scratch i CMS personalitzat

La millor manera de complir el requisit de tenir una àrea d'administració totalment personalitzable i amb una interfície només amb els components necessaris és tenir un CMS personalitzat i creat a mida pel lloc web. D'aquesta manera, jo, com a desenvolupador, podré tenir un major control de tot allò que es mostra en la interfície de l'administració i de la web, eliminant qualsevol possibilitat que els administradors puguin actualitzar extensions, com sí succeeix en les dues alternatives comentades anteriorment.

Centrant-nos ja en aquest mètode seguit, per desenvolupar aquesta web i gestor de continguts s'usaran tecnologies que ens ajudaran a tenir una bona base per construir el projecte. Aquestes tecnologies o frameworks seran: per la part del *backend* Ruby On Rails 5 i de cara al desenvolupament *frontend* s'usarà Foundation 6. Partint d'aquests frameworks es començaran a desenvolupar tots els requisits del projecte.

De cara a la visualització de la web que els usuaris veuen, no es requereixen grans canvis, sí algunes millores en la usabilitat.

Pel que fa a la fulla d'estils, no s'aprofitarà res del que hi ha desenvolupat perquè no es fa ús de cap processador de CSS, eina actualment imprescindible si es vol tenir un desenvolupament més ràpid i modular en la capa d'estils d'una aplicació.

De cara a poder gestionar les diferents versions del projecte es farà ús de GIT, el control de versions més utilitzat avui en dia; aquesta eina permet crear diferents branques per cada tasca que es realitzi, així, en un futur, podré localitzar en el log de git el desenvolupament fet en una tasca en concret, també em servirà per afegir tags de versió del desenvolupament del projecte.

1.4 Planificació

Tasca	Març	Abril	Maig	Juny
Disseny de bases de dades	08/03/2017 - 05/04/2017			
Diagrama UML de classes	08/03/2017 - 05/04/2017			
Wireframe de la interfície	08/03/2017 - 05/04/2017			
Memòria del TFG - PAC2	08/03/2017 - 05/04/2017			
Instal·lació de l'entorn		06/04/2017 - 12/05/2017		
Scaffold d'usuaris		06/04/2017 - 12/05/2017		
Implementació CRUD categories		06/04/2017 - 12/05/2017		
Implementació CRUD productes		06/04/2017 - 12/05/2017		
Implementació CRUD imatges		06/04/2017 - 12/05/2017		
Implementació *Frontend* Administració		06/04/2017 - 12/05/2017		
Implementació *Frontend* web pública		06/04/2017 - 12/05/2017		
Memòria del Treball PAC3		06/04/2017 - 12/05/2017		
Finalitzar proves en el desenvolupament			15/05/2017 - 12/06/2017	
Finalitzar memòria			15/05/2017 - 12/06/2017	
Elaborar presentació (escrita-visual)			15/05/2017 - 12/06/2017	
Elaborar presentació vídeo			15/05/2017 - 12/06/2017	
Elaborar autoinforme del TFG			15/05/2017 - 12/06/2017	

Figura 1 - Planificació

Veure Annex A

1.5 Fites significatives

Fita	Data
Tancament de la planificació	07/03/2017
Tancament dels dissenys	05/04/2017
Tancament del desenvolupament	06/06/2017
Finalització del Treball Final	12/06/2017

Figura 2 - Fites significatives

2. Disseny

El disseny proposat per a l'aplicació ha tingut com a objectiu mantenir una interfície el més neta possible, parlant en termes anglosajó s'ha volgut aproximar al "flat design".

El "flat design" o "disseny Pla" consisteix a eliminar o reduir tot tipus de decoració en un disseny d'interfície o web per simplificar el missatge i facilitar la funcionalitat. S'eliminen textures, degradats, bisellats, ombrejats... en definitiva, tot el que no aporti valor al missatge o informació que es vol transmetre a l'usuari que interactua amb la web.

Aquest tipus de disseny és vàlid per a la majoria de pàgines webs que trobem en l'actualitat, poden haver-hi altres contextos on sigui més adequat apostar per dissenys més elaborats o analògics, però si el que volem és facilitar la lectura de la informació, com és el cas que ens trobem en el present em sembla una decisió correcte.

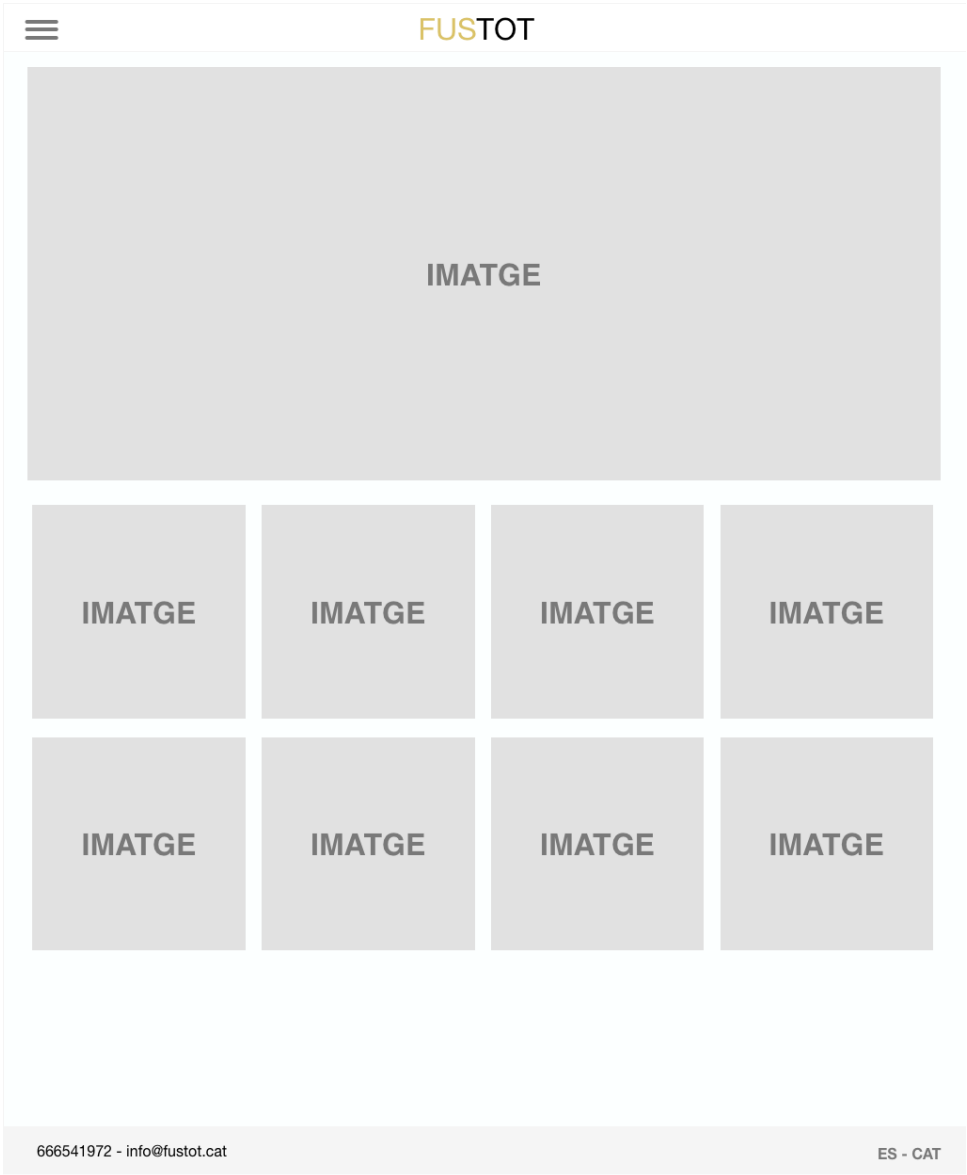
En el moment de decidir un framework frontend també va influenciar molt l'aspecte que tenien els components i pluguins que incorporaven, en el cas de Foundation 6 els components són absolutament flat design, això i altres aspectes com la possibilitat de crear el sistema de grid a través de Flexbox em va fer decantar en contra de Bootstrap 3.

Pel que fa els wiraframes de les pàgines següents, l'objectiu d'aquests ha sigut fer un esquema de les pàgines que componen el projecte, la seva ordenació, elements de la interfície i sistemes de navegació. Els wireframes són esquemàtics i no són d'alta fidelitat respecte al disseny implementat.

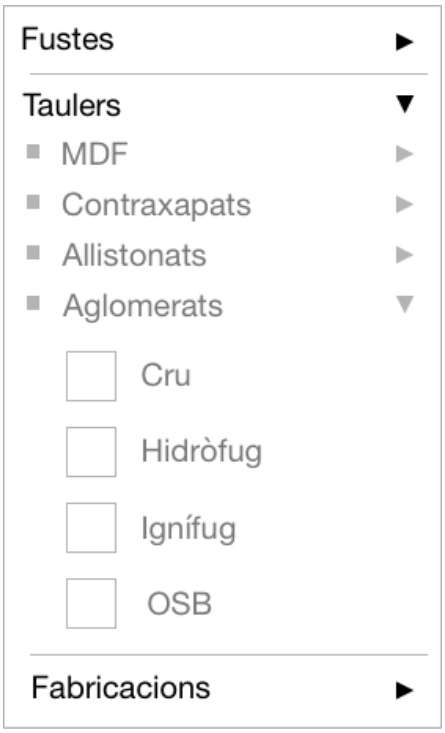
2.1 Home

El que s'ha volgut aconseguir en la portada o home és que tingui un gran impacte visual a través d'imatges grans dels productes.

Bàsicament, el contingut el forma una imatge principal que ocupa el 100% de l'amplada del contenidor, i tot seguit un llistat d'imatges de productes en format quadrat de diferents productes del web, cadascun dels elements d'aquest llistat enllaça a la pàgina de producte.



2.2 Navegació de productes



La navegació de productes estèticament es veurà igual tant en l'administració com a la pàgina que tenen accés els usuaris. La idea d'aquesta navegació és que puguin navegar pels productes sense perdre la pàgina la que estan, això es farà gràcies al desplegament i plegament dels diferents llistats dels productes quan es faci clic a les fletxes, en canvi, si fem clic al nom, anirem a la pàgina enllaçada.

Per millorar la localització dels productes i fer més atractiu l'aspecte visual de la navegació es visualitzaran imatges en miniatura.

2.3 Pàgina de producte

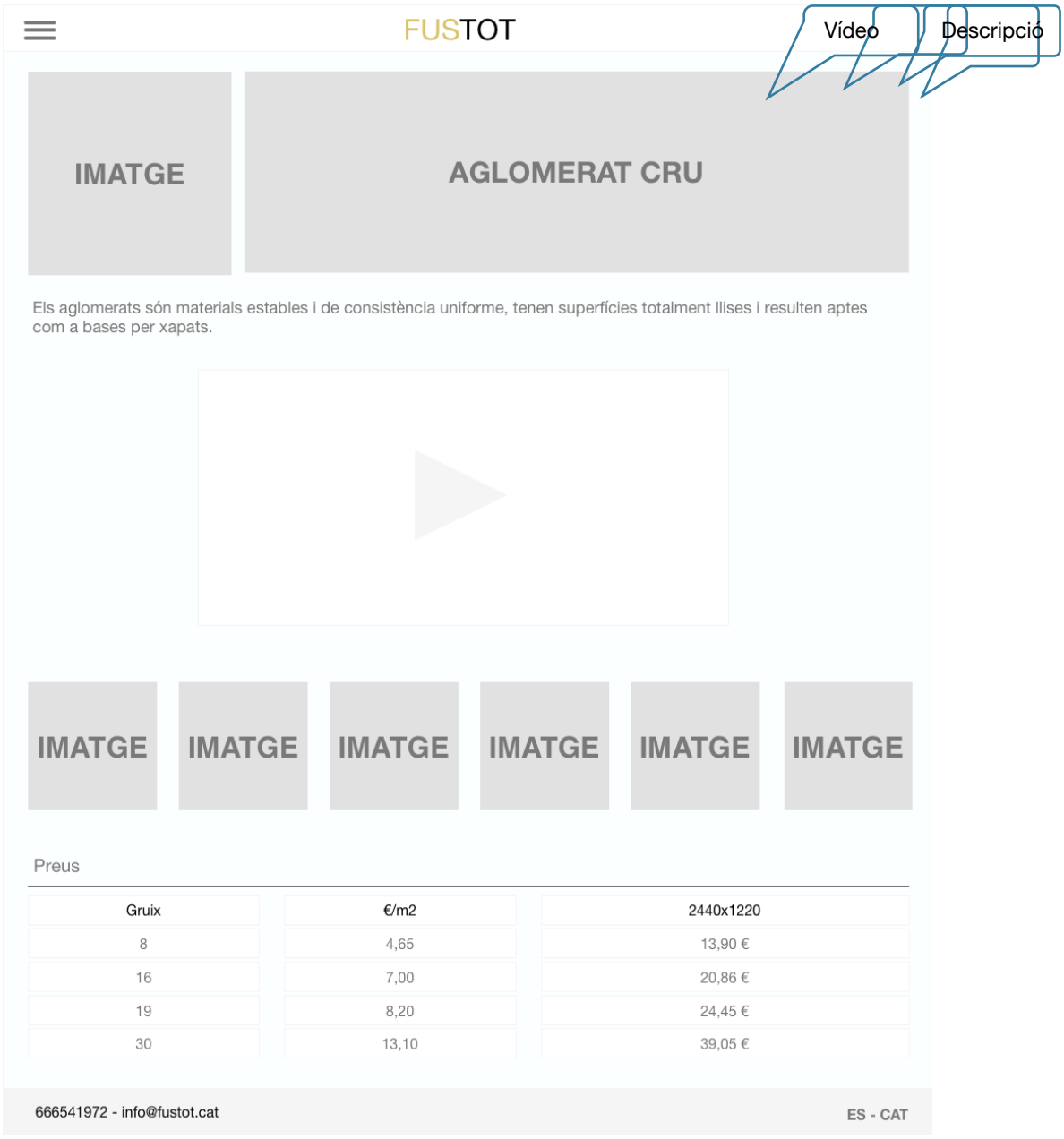
La pàgina de producte es compondrà per diverses seccions que componen la informació. Aquestes seccions són les següents: la capçalera (que sempre apareix), la descripció del producte, el vídeo, una galeria d'imatges i el llistat de preus.

En la part superior trobarem l'encapçalament amb el producte de la imatge i el seu nom.

Després, aniran apareixent apartats segons si existeix la seva informació. Per exemple, si el producte no té vídeo vinculat, no es mostrarà en l'HTML res que faci referència a aquest.

El vídeo té la possibilitat de posar-se en pantalla completa, ja que els serveis de vídeo per Internet contemplen sempre aquesta opció.

En el cas de la galeria d'imatges, també es podrà veure a pantalla completa i es podrà navegar per les fotos a través d'aquesta.


FUSTOT
Vídeo
Descripció
IMATGE
AGLOMERAT CRU
Els aglomerats són materials estables i de consistència uniforme, tenen superfícies totalment llises i resulten aptes com a bases per xapats.
IMATGE
IMATGE
IMATGE
IMATGE
IMATGE
IMATGE
Preus
Gruix
€/m2
2440x1220
8
4,65
13,90 €
16
7,00
20,86 €
19
8,20
24,45 €
30
13,10
39,05 €
666541972 - info@fustot.cat
ES - CAT

2.4 Navegació de l'administració

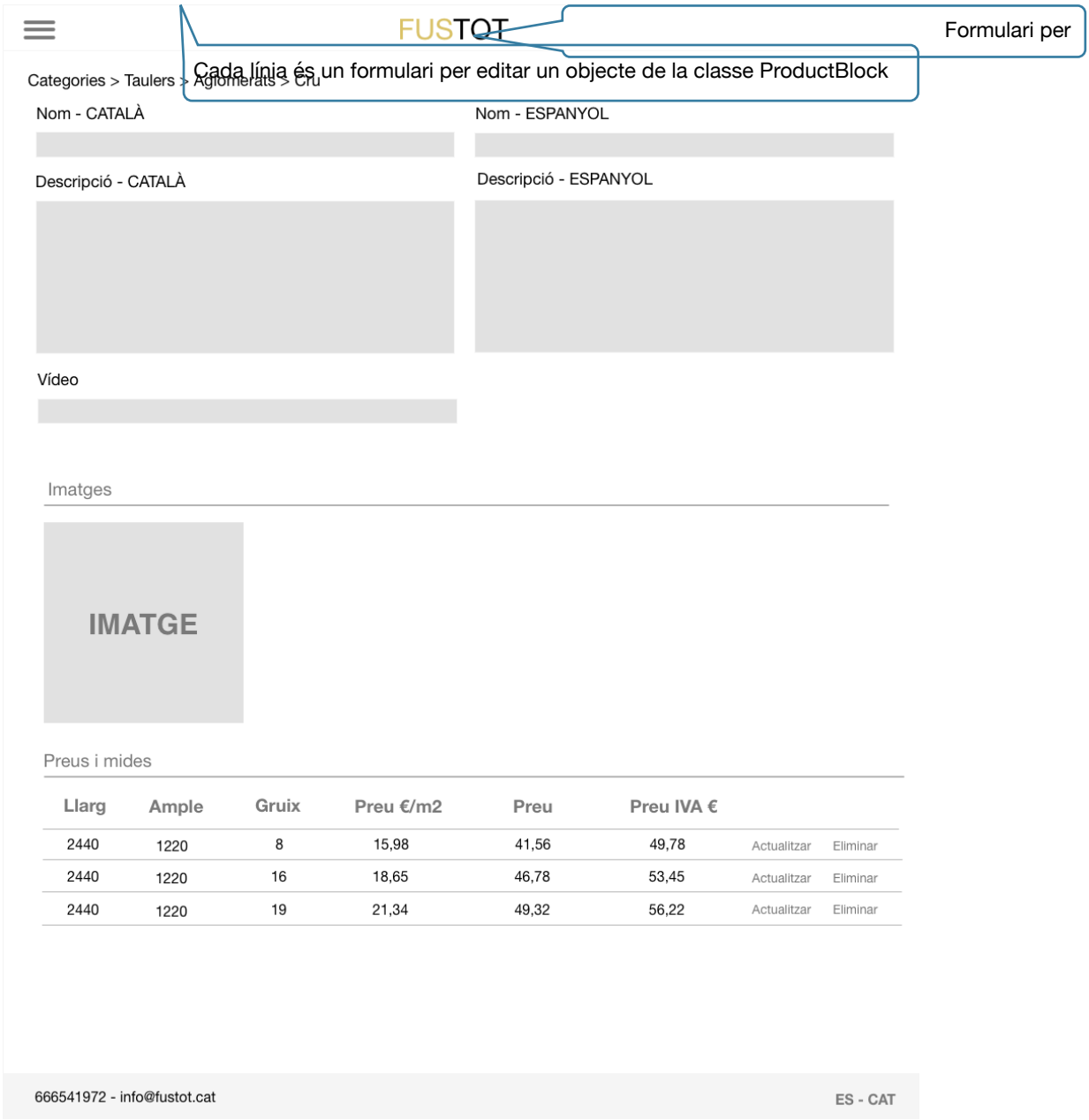
Quan creem o editem una categoria, subcategoria o producte ens apareixerà un formulari per introduir el nom d'aquests en els dos idiomes que l'aplicació treballa. Des d'aquí també podrem eliminar l'element en el qual ens trobem o crear una subcategoria si ens trobem dins la categoria o crear un producte si ens trobem dins d'una subcategoria. L'aplicació no permet crear un producte fins que tot l'arbre sigui complet.

2.5 Gestió de producte

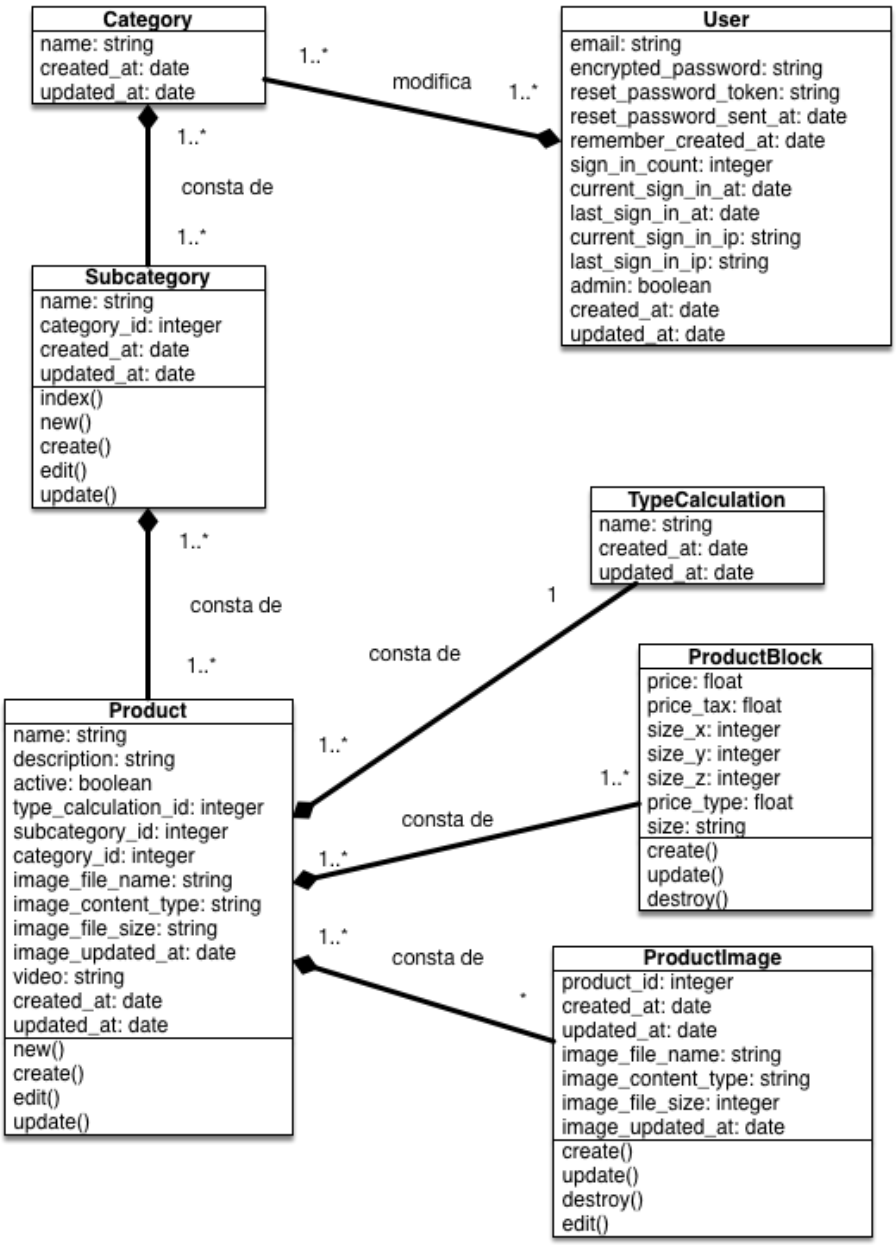
La gestió de producte dins l'administració es compon de 3 diferenciades seccions.

La primera d'elles és el formulari del producte, on tenim els camps de nom i descripció pels dos idiomes, també en aquest formulari es pot afegir un vídeo i la imatge de producte, si el camp del vídeo o descripció es deixa en blanc no és problema, perquè no es mostrarà en la pàgina del producte.

La segona secció fa referència a la galeria d'imatges i permetrà afegir i eliminar imatges en aquesta, en cas de deixar aquesta secció en blanc a la pàgina de producte no es mostrarà galeria, de la mateixa manera que passa amb el vídeo i la descripció.


FUSTOT
Formulari per
Cada línia és un formulari per editar un objecte de la classe ProductBlock
Categories > Taulers > Aglomerats > Cru
Nom - CATALÀ
Nom - ESPANYOL
Descripció - CATALÀ
Descripció - ESPANYOL
Vídeo
Imatges
IMATGE
Preus i mides
Llarg Ample Gruix Preu €/m2 Preu Preu IVA €
2440 1220 8 15,98 41,56 49,78 Actualitzar Eliminar
2440 1220 16 18,65 46,78 53,45 Actualitzar Eliminar
2440 1220 19 21,34 49,32 56,22 Actualitzar Eliminar
666541972 - info@fustot.cat
ES - CAT

3. Diagrama de classes



Totes les classes tenen els atributs de created_at i updated_at que es defineixen de manera predeterminada a Ruby on Rails.

Category

La classe Category és la classe mare de tota la jerarquització de productes i només conté un atribut, que és el del nom. En el model de classes, dins la Ruby On Rails, hi ha definit que el model de Subcategory depèn del de Category, per tant, si s'elimina un registre d'aquest, també s'eliminen tots els registres de les subcategories d'aquella categoria.

Subcategory

Subcategory és la classe filla de Category i conté un atribut anomenat *name* a més a més de la referència de la categoria a la qual pertany, que és l'atribut *category_id*. En Ruby On Rails, el model de classes amb el que es treballa conté les mateixes propietats que el de disseny de les taules relacionals.

La classe *Product* depèn de la de Subcategory, i com he dit en el punt anterior, si eliminem una Subcategory, s'eliminen tots els registres de la base de dades de productes.

Product

La classe Product és la més extensa de l'aplicació i conté molts atributs per descriure-la. El nom, descripció, imatges o tipus de calculació.

Continuant fent referència a les dependències entre classes, les classes ProductBlock i ProductImages depenen de Product, per tant, si eliminem un producte, també eliminem les seves dependències.

User

Els atributs de la classe User venen heretats en la seva majoria de la gem Devise (més endavant explicaré què és una gem).

Type Calculation

TypeCalculation és una classe que ens informa del tipus de calculació es compondrà el preu del producte, en el cas de la nostra aplicació hi han 4: metro lineal, metro quadrat, metro cúbic o per unitat.

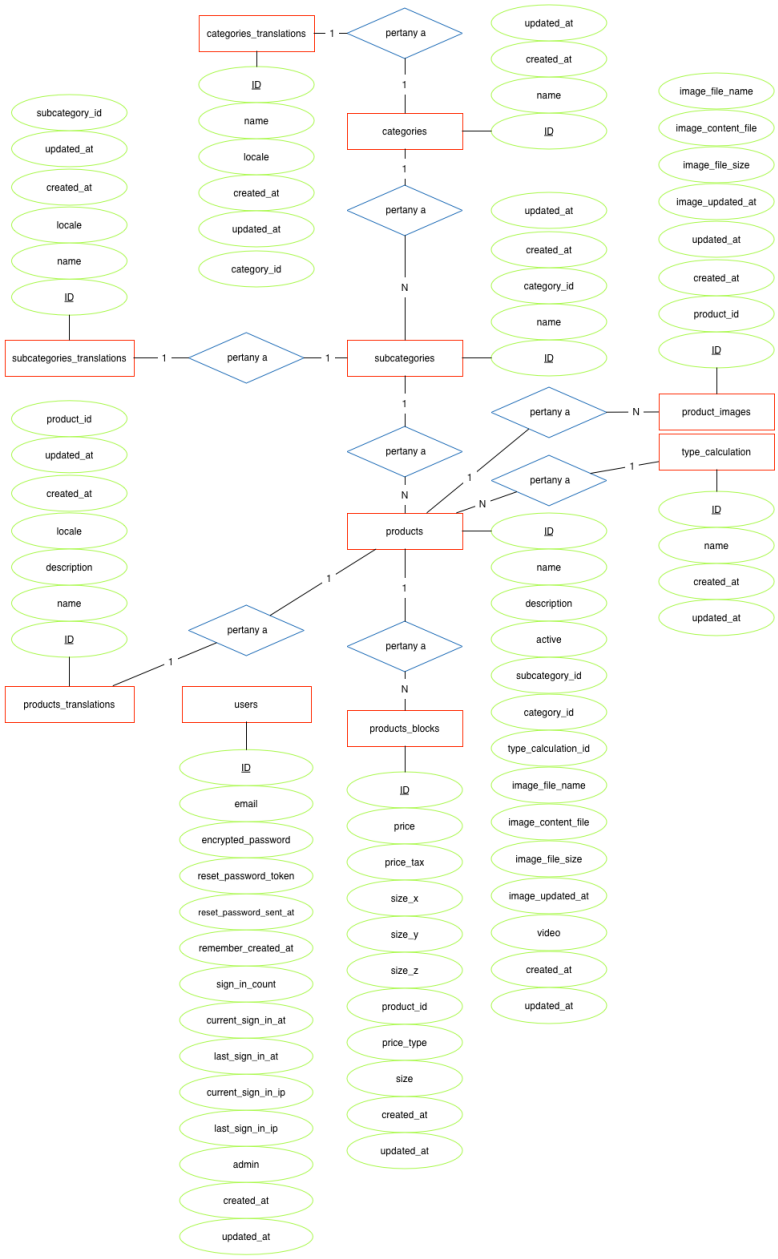
ProductBlock

ProductBlock és una classe filla de Product i en aquesta, s'estructuren les dades de cada mesura única de producte amb el seu corresponent preu.

ProductImage

ProductImage és una classe que filla de Product, de la mateixa manera que ProductBlock, i conté la informació de les imatges que componen la galeria d'imatges d'un producte.

4. Disseny entitat-relació



Abans de fer un repàs per cadascuna de les taules del disseny de l'entitat relació, comentar que en Ruby On Rails, cada taula té l'atribut identificador ID per defecte i serà sempre present en les taules que creem.

categories

Com en el model de classes, la taula categories és la mare de la jerarquització de tota l'estructura de dades d'un producte.

Conté, com a cada taula, la clau primària ID i pot tenir moltes subcategories associades.

categories_translations

Aquesta taula no és creada explícitament en les migracions realitzades per mi en la base de dades, sinó, que es creen quan en el model de cada classe informem dels atributs que han de ser traduïts, llavors, al crear el primer registre de categories, automàticament l'atribut *name*, que és en aquest cas, el que requereix traducció es guarda en aquesta taula, que fa referència a la taula de categories amb l'atribut category_id.

El punt important i a destacar d'aquesta taula està en l'atribut *locale*. La manera en com tracta la internacionalització de les dades, formats i llengües en Ruby on Rails és a través de la *gem I18n*, aquesta incorpora, en àmbit global del sistema, una variable anomenada *locale* que ens informarà del llenguatge utilitat. Doncs bé, cada vegada que guardem en aquesta taula un registre, en l'atribut *locale* es guardarà l'idioma del qual s'està fent referència.

subcategories

La taula de subcategories conté com a clau forana category_id, que fa referència a la categoria a la qual pertany, la resta d'atributs els componen el nom, created_at i updated_at.

subcategories_translations

De la mateixa manera que hem vist en la taula categories_translations, també en tenim una altra taula amb les mateixes característiques però fent referència aquest cop a les subcategories.

products

La taula products és, juntament amb la de users, la més gran de la base de dades, en aquesta trobem tres claus foranes que referencien a la categoria, subcategoria i al tipus de calculació en les que es basarà el preu, també hi ha els atributs que fan referencia a la imatge de producte, el tipus de calculació del preu, i per descomptat el nom i la descripció.

products_translations

Igual que ens el dos anteriors casos, amb la diferencia que aquí guardem el nom i la descripció.

product_images

Aquesta taula fa referència a les galeries d'imatges, s'ha afegit la clau forànea product_id perque poguem obtenir les imatges d'un producte determinat.

product_blocks

Cada registre d'aquesta taula forma la dada que inclou el preu i les mides d'un producte.

price i *price_tax* fan referència al preu sense impostos i al preu amb els impostos inclosos respectivament. size_x, size_y i size_z contenen la informació de les mides en els eixos x, y i z, si hi ha un producte que només ens interessa el valor d'un eix la resta

poden estar buits. L'atribut *size* és un atribut compost que guarda en fomat *string* les dades de les mides dels 3 eixos.

Un altre atribut a comentar és price_type, aquest fa referència al preu que té segons el tipus de calculació de preu que té aquell producte.

M'explicaré millor amb un exemple: el producte “llistó de fusta” té com a type_calculation el valor de ml (metre lineal), doncs el price_type farà referència al preu del metre lineal d'aquella mida en concret.

Conté una clau forana product_id per guardar la referència del producte.

type_calculations

type_calculations és una taula que es guarden el tipus de calculacions que l'aplicació pot calcular, hi ha 4 tipus: metre lineal, metre quadrat, metre cúbic i preu unitari.

users

La taula users en la que actualment hi ha un sol usuari, que és l'administrador de la web, conté una sèrie d'atributs heretats de la gem Devise, que ens permet saber informació interessant respecte els usuaris, com l'última vegada que s'ha identificat, quants cops ho ha fet i amb quina IP. Són dades que ara no són rellevants perquè només hi ha un usuari, però és molt probable, que en un futur, sobri el registre d'usuaris en la web pública i el sistema ja estarà preparat per això.

L'atribut admin l'he afegit per identificar aquells usuaris que poden accedir a l’administració.

5. Desenvolupament

5.1 Llibreries utilitzades

Les llibreries a Ruby On Rails tenen una molt fàcil implementació i un dels mètodes més utilitzats i recomanats és a través de gems. Les gems són paquets de llibreries per a Ruby que s'instal·len al sistema i queden preparades per ser usades.

El fitxer en el qual gestionem les gems en un projecte de Ruby On Rails s'anomena Gemfile, en aquest llistarem tots els paquets que volem que formin part del nostre projecte. Dins del fitxer, podem cridar a l'última versió d'un paquet o una versió concreta d'aquest.

Un exemple del Gemfile del projecte:

```
gem 'globalize', git: 'https://github.com/globalize/globalize'
gem 'activemodel-serializers-xml'
gem "paperclip", "~> 5.0.0"
gem ‘blueimp-gallery'
```

Un cop introduïdes les gems cal escriure una comanda en la consola per instal·lar-les. La comanda és *bundle install*.

A continuació llistaré totes les llibreries que en un inici no implementa Ruby On Rails 5 i que he afegit per a dur a terme el projecte:

HAML

https://github.com/haml/haml

Haml és un motor de plantilles per a HTML. Està dissenyat perquè sigui més fàcil i més agradable escriure documents HTML eliminant la redundància, també proporciona una sintaxi elegant i fàcil d'entendre.

DEVISE

https://github.com/plataformatec/devise

Devise és una solució d'autenticació flexible per rails basat en Warden que està basada en Rack permetent una solució completa MVC en projecte Ruby on Rails.

GLOBALIZE

https://github.com/globalize/globalize

Globalize es basa en l'API de I18n en Ruby on Rails per afegir model de traduccions als models d'ActiveRecord.

FACTORY_GIRL_RAILS

https://github.com/thoughtbot/factory_girl_rails

Factory Girl permet crear objectes i les seves associacions fent servir els models de l'aplicació Ruby on Rails en entorn de test.

DATABASE_CLEANER

https://github.com/DatabaseCleaner/database_cleaner

Database Cleaner és un conjunt d'estratègies per a la neteja de la base de dades en Ruby. En el cas del nostre projecte es fa servir per netejar la base de dades de test cada vegada que es finalitza un.

RSPEC-RAILS

https://github.com/rspec/rspec-rails

Rspec és un framework de test per Ruby on Rails que proporciona una sintaxis més àmplia i amb més possibilitats que la que proporciona Ruby on Rails de manera nativa.

CAPYBARA

https://github.com/teamcapybara/capybara

Capybara ajuda a provar les aplicacions web mitjançant la simulació de com un usuari real podria interactuar amb la web en entorn de test.

SELENIUM-WEBDRIVER

https://rubygems.org/gems/selenium-webdriver/versions/2.53.0

Selenium WebDriver és un navegador virtual per entorn de test. El seu objectiu és imitar el comportament d'un usuari real, i com a tal, interactua amb el codi HTML de l'aplicació.

PAPERCLIP

https://github.com/thoughtbot/paperclip

Paperclip proporciona crear una biblioteca d'arxius adjunts fàcil per ActiveRecord. En el cas que ens ocupa, ens facilita la creació de vincular imatges als productes del web.

BLUEIMP GALLERY

https://github.com/Phifo/blueimp-gallery

Blueimp és una llibreria per a crear galeries d'imatges pensada per tot tipus de dispositius, inclosos els dispositius tàctils.

FOUNDATION-RAILS

https://github.com/zurb/foundation-rails

Foundation Rails és un framework frontend que proporciona múltiples components CSS i plugins JS per facilitar el desenvolupament.

La integració d'aquesta llibreria tan important en el nostre projecte ve molt ben documentada, es tracta de fer 3 senzills passos, que són els següents:

Afegir la llibreria al fitxer Gemfile:

```
gem 'foundation-rails'
```

Instal·lar i actualitzar les gems del projecte:

```
bundle install
```

Foundation ens facilita la feina i ens proporciona una comanda de configuració que ens generarà els fitxers necessaris per fer córrer el framework:

```
rails g foundation:install
```

Amb això ja podrem fer ús de tots els components i pluguins que es proporciona el framework de l'empresa Zurb.

5.2 Extractes de codi

5.2.1 Breadcrumb

```
def sitemap(controller_name, action)

  sitemap_not_found = ['']
  case controller_name

  when 'admin/categories'
    case action
    when 'new'
      sitemap_values = [sitemap_element('Crear categoria')]
    when 'edit'
      sitemap_values = [sitemap_element(@category.name)]
    else
      sitemap_values = sitemap_not_found
    end

  when 'admin/subcategories'
    case action
    when 'new'
      sitemap_values = [
        sitemap_element(link_to(@category.name, edit_admin_category_path(@category))),
        sitemap_element('Crear subcategoria')
     ]
   else
      sitemap_values = sitemap_not_found
   end
   when 'admin/products'
    case action
    when 'new'
      sitemap_values = [
        sitemap_element(link_to(@category.name, edit_admin_category_path(@category))),
        sitemap_element(link_to(@category.name, edit_admin_category_path(@category))),
        sitemap_element('Crear producte')
```

```
      ]
    else
      sitemap_values = sitemap_not_found
    end

  if controller_name.include?("admin/")
    levels = sitemap_values.inject([content_tag(:li, link_to('Admin', '/admin'))], :<<)
  else
    levels = sitemap_values.inject([content_tag(:li, link_to('Home', '/'))], :<<)
  end

  levels
end

def sitemap_element(content, options = {})
  content_tag(:li, content, options)
end
```

Aquest extracte de codi, on hi han dos mètodes, és l'encarregat de crear i mostrar el breadcrumb o fil d'ariadna en català. Es pot observar que ens trobem en dos "case", el primer d'ells comprova en quin controlador estem, i un cop detectem el controlador busquem quina és l'acció que ha executat el mètode. Per cada acció hi ha configurada les opcions a mostrar del breadcrumb d'aquella pàgina concreta.

5.2.2 Selecció de parcials

```
%h2= "Preus i mides"
= render partial: "product_block_heading_#{@type_calculation}"
= render partial: "product_blocks_list_#{@type_calculation}"
= render partial: “product_block_new_#{@type_calculation}”
```

En primera instància direm que un parcial és un bloc de codi que conté un fitxer determinat. Aquest ens ajuda a organitzar el nostre codi i facilita el seu manteniment.

En el moment de seleccionar diferents parcials i evitar afegir un excés de condicionals en el codi s'ha optat per afegir el valor d'una variable com a part del nom de l'arxiu, en aquest cas @type_calculation dins del "string" que comporta el nom del parcial. Perquè això sigui possible, cadascun dels parcials del codi ha de mantenir un mateix patró i tenir el valor de la variable com a part de nom de fitxer.

5.2.3 Layout

```
!!!
%html
  %head
    %meta{ charset: "utf-8" }
    %meta{ name: "viewport", content: "width=device-width, initial-scale=1.0" }
    %title= content_for?(:title) ? yield(:title) : 'Fustot'
    = stylesheet_link_tag "application"
    = javascript_include_tag "application", 'data-turbolinks-track' => true
    = csrf_meta_tags
  %body
    = render partial: 'layouts/partials/structure/header'
    %main{ role: :main, class: 'expanded row' }
      = render partial: 'layouts/partials/structure/navigation'
      #content.small-12.medium-9.large-9.large-offset-3.columns
        = yield
    = render partial: 'layouts/partials/structure/footer'
```

El marcat HTML que compon el layout de l'aplicació està escrit amb Haml, com hem vist en la secció de llibreries utilitzades, HAML és un motor de plantilles per a HTML. Està dissenyat perquè sigui més fàcil i més agradable escriure documents HTML, i com es pot observar en el codi, una de les seves característiques més destacades és que les etiquetes HTML no es tanquen, només cal cridar-la amb el signe % i després el nom de l'etiqueta.

Dins l'etiqueta head trobem l'etiqueta title, que dóna nom al document i les crides als assets, és a dir, als fitxers JavaScript i CSS.

El JavaScript incorpora funcionalitats de manipulació del DOM o l'estructura del document, i el CSS, dit també fulls d'estil en cascada, ens aporta les regles visuals de l'aplicació.

Dins l'etiqueta body s'aprecia que hi ha 3 elements fills:

header

És una crida a un parcial que incorpora la capçalera de l'aplicació.

main

Aquí anirà a parar tot el contingut del web.

footer

El peu de pàgina, que actualment hi podem trobar l'opció dels idiomes.

5.2.4 Llistat de preus

```
%table
  %tr
    %th= I18n.t("size_in_mm")
    %th= "€/ml"
  - @product.product_blocks.each do |product_block|
    %tr
      %td= "#{product_block.size_x}x#{product_block.size_y}"
      %td= product_block.price_type
```

Quan cridem a aquest parcial és perquè volem pintar en l'HTML la taula de preus d'un producte determinat. Com es pot observar, els preus es representen mitjançant l'etiqueta table, després, en la primera fila (tr) afegim les dues cel·les capçaleres (th).

A partir d'aquí, recorrem les dades del model ProductBlock d'un producte determinat i anem afegint files de dues cel·les, la primera d'elles introduïm la mida, i en la segona, el preu.

5.2.5 Navegació de productes

```
= render partial: 'layouts/partials/structure/breadcrumb'

%ul.list-square
  - Product.where(subcategory_id: params[:subcategory_id]).each do |product|
    %li
      = link_to product_path(product) do
        %figure
          %figcaption
            %h2= product.name
          = image_tag product.image.url
```

En la navegació de producte (no la del sidebar lateral), dins l'acció index del controlador Product, llistem tots els productes d'una subcategoria determinada (la informació d'aquesta ve dins la variable global params).

Primer fem la crida al breadcrum, tot seguit es comença a construir en l'HTML una llista desordenada on dins d'aquesta es crearà un loop que anirà iterant per tots els productes trobats per la subcategoria que ens informi la variable params[:subcategory_id].

Per cada iteració, crearem un element de llista, dins d'aquesta, un enllaç que enllaça a la pàgina de producte, i dins l'enllaç, la imatge i el seu corresponent nom.

L'estil de com es veurà queda determinat a la classe .list-square que tot seguit repassarem en el CSS.

5.2.6 Mixin list-square

```
@mixin list-square {
  .list-square {
    list-style-type: none;
    margin: 0 -10px;
    li {
      width: 25%;
      float: left;
      position: relative;
    }
    a {
      border: 1px solid #f5f5f5;
      display: block;
      margin: 10px;
      transition: all 200ms linear;
    }
    h2 {
      background-color: #fff;
      bottom: 10px;
      color: #000;
      font-size: 18px;
      font-weight: 400;
      left: 10px;
```

```
      opacity: .7;
      padding: .5em;
      position: absolute;
      width: 80%;
    }
    img {
      width: 100%;
      max-width: 100%;
      height: auto;
    }
  }
}
```

Abans de res, explicaré què és un mixin de SASS. El mixin de SASS és un bloc de codi que es pot reutilitzar repetidament al llarg de l'aplicació, i que qualsevol canvi en ell quedarà palès a totes les crides on s'ha realitzat. També, ajuda a millora l'organització de tota la infraestructura de preprocessador SASS.

Doncs bé, en aquest mixin, tenim les regles que apliquen a la classe *.list-square*, vista anteriorment en l'anterior secció.

Bàsicament, el que s'ha aconseguit amb aquestes regles és que els elements del llistat no tinguin cap mena de vinyeta i que cadascun d'ells ocupi un 25% d'amplada. També, gràcies a la regla *opacity* i a la regla *transition* s'ha creat una transició suau de l'opacitat del fons on està situat el nom del producte.

Respecte a les imatges, s'ha determinat que ocupin l'espai del contenidor que l'embolcallen.

5.2.7 Pàgina de producte

```
= render partial: 'layouts/partials/structure/breadcrumb'

.mod-product

  .header.row.expanded.collapse
    .image.small-3.columns
      = image_tag @product.image.url
    .title.columns
      .row.expanded.align-middle
        %h1.column= @product.name

  %p.description= @product.description

  - if @product.video.present?
    .row.expanded.align-center
      .small-12.medium-7
        %section.video.responsive-embed.widescreen
          = @product.video.html_safe

  - if @product_images.present?
    #blueimp-gallery.blueimp-gallery
      .slides
      %h3.title
      %a.prev
      %a.next
      %a.close
      %a.play-pause
      %ol.indicator

    %h2= I18n.t("gallery_images")

    %ul.no-bullet#links.mod-gallery
      - @product_images.each do |product_image|
        %li
          = link_to product_image.image.url do
            = image_tag product_image.image.url(:thumb)

  %h2= I18n.t("prices")
  = render partial: "prices_list_#{@type_calculation}"
```

Aquest bloc de codi fa referència a la pàgina de producte. Un cop s'ha fet la crida al parcial de breadcrumb, pintem en l'HTML un element div amb la classe *mod-product*, que serà la classe que farà de paraigües a tots els elements que estan dins la informació del producte.

Després d'això, es va construint i escrivint totes les seccions que componen una pàgina de producte com la capçalera i la descripció.

Tot seguit hi ha la secció del vídeo però, en aquest cas, s'ha afegit una condició que diu que si aquell producte no té cap informació de vídeo doncs que no pinti res. Per tant, no tindrem codi d'estar per estar, només quant tenim la informació pintarem els blocs necessaris. Amb la galeria d'imatges succeeix el mateix.

Finalment, en última instància, fem una crida al parcial que correspongui al tipus de calculació de preu.

5.2.8 Home

```
.row.expanded
  .small-12.columns
    = image_tag 'slider/slider1.jpg', width: '100%'

  .small-12.columns
    %ul.list-square
      - Product.first(4).each do |product|
        %li
          = link_to product_path(product) do
            %figure
              %figcaption
                %h2= product.name
              = image_tag product.image.url
```

La pàgina principal o home es compon de dues columnes que ocupen el 100% de l'amplada del contenidor. Això és així perquè la classe small-12 fa que que ocupi 12 columnes, i com que el sistema de reixes és format per 12 columnes doncs per aquest motiu ocupa el 100%.

En la primera columna fem una crida a imatge_tag, un helper o també anomenat mètode que construeix HTML a partir d'uns paràmetres, en aquí pintem la imatge principal més gran amb una visualització d'amplada del 100%.

5.2.9 Controlador de Product

```
class Admin::ProductsController < AdminController

  before_action :set_product, only: [:edit, :update, :destroy]

  def new
    @product = Product.new
    @subcategory = Subcategory.find(params[:subcategory_id])
    @category = @subcategory.category

    @type_calculations = []
    TypeCalculation.all.each do |type_calculation|
      @type_calculations << [type_calculation.name, type_calculation.id]
    end
  end

  def edit
    @product_block = ProductBlock.new
    @product_blocks = ProductBlock.where(product_id: @product.id).sort_by {|pb| [pb.size_z,
pb.size] }
    @product_image = ProductImage.new
    @product_images = ProductImage.where(product_id: @product.id)
    @category = @product.subcategory.category
    @subcategory = @product.subcategory
```

```
    @type_calculation = TypeCalculation.find(@product.type_calculation_id).name
  end

  def create
    @product = Product.new(product_params)
    @product = set_attributes_languages(@product, ['name', 'description'])

    respond_to do |format|
      if @product.save
        format.html { redirect_to edit_admin_product_path(@product), notice: 'Product was
successfully created.' }
      else
        format.html { render :new }
      end
    end
  end

  def update
    @product = set_attributes_languages(@product, ['name', 'description'])

    respond_to do |format|
      if @product.update(product_params)
        format.html { redirect_to edit_admin_product_path(@product), notice: 'Product was
successfully updated.' }
      else
        format.html { render :edit }
      end
    end
  end

  def destroy
    @product.destroy
    respond_to do |format|
      format.html { redirect_to admin_root_path, notice: 'Product was successfully destroyed.' }
    end
  end
```

```
  private
    # Use callbacks to share common setup or constraints between actions.
    def set_product
      @product = Product.find(params[:id])
    end

    # Never trust parameters from the scary internet, only allow the white list through.
    def product_params
      params.fetch(:product, {}).permit(
        :name,
        :description,
        :active,
        :type_calculation_id,
        :subcategory_id,
        :image,
        :video
      )
    end
end
```

Comentarem alguns dels aspectes destacats del controlador, en un primer moment ens trobem amb la línia que diu: before_action :set_product, only: [:edit, :update, :destroy]

Això vol dir que abans d'entrar a qualsevol de les tres accions que hi ha esmentades (edit, update, destroy) es cridi el mètode set_product, que aquest guardarà en una variable @product l'objecte amb el que estem treballant.

En l'acció new i edit es preparen les variables per crear un objecte buit i per proporcionar al breadcrumb les dades necessàries per construir-lo.

Pel que fa als mètodes update i destroy son molt similars amb la diferència que un actualitza i l'altre esborra.

Un mètode a destacar és l'últim del codi anomenat product_params, en aquí estem determinant quins són els atributs que l'aplicació acceptarà. Per exemple, nosaltres tenim un formulari en la web, i una persona obre les eines de desenvolupament del navegador i canvia les dades i els atributs del formulari de manera malintencionada per introduir dades il·lícites, doncs qualsevol paràmetre que no estigui estipulat dins d'aquest mètode se li farà cas omís, aportant seguretat al sistema.

5.2.10 Application.scss

```
*= require_tree .
*= require_self
*= require foundation_and_overrides
*= require blueimp-gallery-all
@import 'partials/structure/header';
@import 'partials/structure/navigation';
@import 'partials/structure/general';
@include header;
@include navigation;
@include general;

@import 'partials/lists/list-square';
@include list-square;

@import 'partials/modules/mod-gallery';
@import 'partials/modules/mod-product';
@include mod-gallery;
@include mod-product;
```

Aquest és el fitxer mare i des d'on es fan les crides a tots els parcials de SASS, en un primer moment es fan les crides que component tots els components del framework Foundation 6 a través de *require*.

A sota fem les crides dels parcials de SASS de creació pròpia. Primer fem un @import d'un fitxer, però com que dins d'aquest fitxer el codi està embolcallat dins d'un mixin doncs després fem la crida del mixn a través del @*include*.

5.2.11 Mixin mod-product

```
@mixin mod-product {
  .mod-product {
    > * {
      margin-bottom: 1rem;
    }
    .header {
      color: white;
      font-weight: bold;
      text-transform: uppercase;
      font-size: 40px;
      padding: 0 15px;
      .image {
        max-width: 150px;
        margin-right: 15px;
      }
      .title {
        background-color: #a53b2a;
        & > .row {
          height: 100%;
        }
      }
    }
    .video {
      border-top: 1px solid #e8e8e8;
      border-bottom: 1px solid #e8e8e8;
    }
  }
}
```

El primer punt destacable que podem trobar és el selector >;; *, aquesta selector té la regla margin-bottom: 1rem que el que fa és que tots els elements que siguin fills directes de mod-product tinguin una distància de 1rem. Per aclarir, 1 rem és la mateixa unitat de mesura que té la mida de la lletra del document.

Dins del selector del header tenim la regla text-transform: uppercase, la seva funcionalitat és de posar en majúscules el text, és a dir, si des de la base de dades ens arriba el text “Tauler de contraxapat”, la web la mostrarà en majúscules gràcies al CSS.

5.2.12 Navegació Javascript

```
$("#navigation span").on("click", function(e){
    var elem = $(this).parent();
    $.ajax({
     url: elem.attr('href')
    })
    .done(function(data){
      $('#content').replaceWith($(data).find('#content'));
    });
    window.history.pushState(null, elem.attr('href'), elem.attr('href'));
  });
```

Aquest codi Javascript el que fa és cridar a un esdeveniment dins la navegació de productes.

En fer clic en l'etiqueta span dins del ID de navigation, farà una petició través d'Ajax a la ruta a la qual apunta l'enllaç.

Un cop es retorni una resposta amb les dades se substitueix el contingut de l'HTML que es trobi dins l'ID content.

S'actualitza l'historial del navegador perquè tingui en compte la nova petició.

6. Implementació del multi-idioma

A continuació faré una explicació de com ha estat implementada la funcionalitat de multi-idioma de la pàgina web de Fustot.

Primer de tot, hem d'incorporar les gems que ens ajudaran a crear la funcionalitat, en aquest projecte he confiat en la gem Globalize que et permet crear taules de traduccions en la base de dades per aquells models que requereixin atributs traduïts. Per tant, s'ha afegit la gem al fitxer Gemfile:

```
gem 'globalize', git: 'https://github.com/globalize/globalize'
```

Tot seguit, cal dir a l'aplicació en quins idiomes funcionarà i quina serà l'assignat per defecte.

```
config.i18n.default_locale = :es
I18n.config.available_locales = [:es, :ca]
```

Després, en el controlador Application, que és el pare de tots els controladors de l'aplicació s'hi ha afegit dos mètodes, un d'ells anomenat set_locale que s'executarà sempre abans que l'aplicació s'encamini una acció de qualsevol controlador, i el que farà és determinar quin és l'idioma en què ha de mostrar la web. Primer mirarà en la variable global de sessió, si no hi ha cap informació al respecte, anirà a la informació de la capçalera de la petició, si tampoc hi ha cap esmena, agafarà l'idioma per defecte, que és l'espanyol.before_filter :set_locale

```
private
  def extract_locale_from_accept_language_header
    request.env['HTTP_ACCEPT_LANGUAGE'].scan(/^[a-z]{2}/).first.to_sym
  end
  def set_locale
    I18n.locale = session[:locale] || extract_locale_from_accept_language_header ||
I18n.default_locale
    session[:locale] = I18n.locale
  end
```

En els models que tinguin atributs que hagin d'estar en els dos idiomes, com és el cas dels productes, cal afegir el següent en el model:

```
translates :name, :description
```

Això el que farà, en la primera vegada que s'enviï el formulari dels productes, crear una taula a la base de dades anomenada products_translations que emmagatzemarà el contingut amb la variable locale corresponent.

A continuació, s'ha creat dues rutes que permeten canviar d'idioma, que és la ruta a la qual apunten els enllaços del peu de pàgina.

```
get 'set_language_catalan', to: 'set_language#catalan'
get 'set_language_spanish', to: 'set_language#spanish'
```

Aquestes accions canvien la variable global I18n.locale:

```
class SetLanguageController < ApplicationController
  def locale_ca
    I18n.locale = :ca
    set_session_and_redirect
  end
  def locale_es
    I18n.locale = :es
    set_session_and_redirect
  end
  private
    def set_session_and_redirect
      session[:locale] = I18n.locale
      redirect_to :back
      rescue ActionController::RedirectBackError
        redirect_to :root
    end
end
```

7. Conclusions

Un cop finalitzat el treball, la meva conclusió del treball és positiva, ho és perquè he après moltes coses, primer de tot en termes de desenvolupament per a mi ha sigut tot un repte, ja que ni el llenguatge Ruby, ni el seu framework Ruby on Rails no els havia provat de manera profunda i haig de dir que estic molt content d'haver-lo escollit per varis motius:

Facilitat a l'hora de treballar amb metodologia MVC

Bona documentació oficial

Molta informació de problemes resolts (stack overflow, youtube..)

Moltes llibreries compatibles

Foundation 6 ha estat una novetat per a mi, sí que és cert que anteriorment havia treballat amb Bootstrap 3, i això m'ha ajudat a entendre'l més ràpidament com s'implementen cadascun dels components. Un dels motius que em van fer optar per aquest framework és perquè el sistema de layout es pot utilitzar fent servir el sistema Flexbox de CSS, cosa que Bootstrap 3 no (Bootstrap 4 sí, però està en estat Alpha).

Altres coses que he après és que no vaig estar molt encertat en la planificació inicial, sobretot quant a nombre de classes i taules a la base de dades, a mesura que el projecte ha anat avançant, he anat comprovant que el que estava planificat a l'inici es va quedar incomplet per satisfer els requisits proposats.

Els objectius plantejats a l'inici s'han complert, però si sóc franc, pensava que assolir-lo sense tantes dificultats i perquè no, anar una mica més enllà, però m'hi he anat trobant en situacions que he tardat molts dies a trobar una solució que em satisfés, com per exemple desenvolupar el sistema multi-llenguatge o la instal·lació de l'entorn d'algunes llibreries, que seguint els passos de la documentació no obtenia els mateixos resultats.

Entenc que aquesta part forma part de l'ofici i l'aprenentatge i que això fa augmentar la meva experiència.

8. Glossari

FRAMEWORK

L'entorn de treball, marc de treball o en anglès *framework* és una infraestructura de programari que, en la programació orientada a objectes, facilita la concepció de les aplicacions mitjançant la utilització de biblioteques de classes o generadors de programes.

LAYOUT

Esquema de distribució dels elements dins del disseny d'una pàgina web.

GIT

És un programari de control de versions dissenyat per Linus Torvalds, pensant en l'eficiència i la fiabilitat del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font.

SASS

Sass és un meta-llenguatge de CSS que s'utilitza per descriure l'estil d'un document de manera neta i estructural.

Sass proporciona una sintaxi més simple i elegant que CSS i implementa diverses característiques útils per a la creació de fulls d'estil manejables com pot ser l'ús de variables, funcions o mixins.

GEM

Les gems de Ruby són paquets de llibreries que s'instal·len en el sistema i queden llestes per ser usades.

ACTIVE RECORDS

Active record és un enfocament per a accés de dades en una base de dades. Una taula de la base de dades està embolicada en una classe i comparteixen atributs. Per tant, una instància d'un objecte està lligada a un únic registre a la taula. Qualsevol objecte carregat obté la seva informació a partir de la base de dades.

WORDPRESS

WordPress és un CMS gratuït que ens permet crear pàgines web de diferent tipus fàcilment. Actualment és el CMS més utilitzat amb una quota del 26% del total de pàgines web.

CMS

Un CMS (Sistema de Gestió de Continguts) és un programari que permet crear, editar, classificar i publicar qualsevol tipus d'informació en una pàgina web d'una manera senzilla.

CLASSE (PROGRAMACIÓ ORIENTADA A OBJECTES)

Una classe és un motlle o una plantilla per a la creació d'objectes de dades que es defineixen en un model. Amb les classes es representen entitats o conceptes a través d'atributs i mètodes.

CLAU PRIMÀRIA

Una clau primària és un atribut únic que identifica a una fila d'una taula d'una base de dades.

CLAU FORANA

Una clau forana és una columna o grup de columnes d'una taula que conté valors que coincideixen amb la clau primària d'una altra taula. Les claus foranes s'utilitzen per unir taules.

9. Bibliografia

José Ramón Rodríguez (2014), Gestió de Projectes. Barcelona. Oberta UOC Publishing, SL

Ruby On Rails Guides [en línia]. [Data de consulta: 1 de març de 2017] <http://guides.rubyonrails.org>

Wikipedia [en línia]. [Data de consulta: 15 de març de 2017] <https://es.wikipedia.org/wiki/Modelo_entidad-relación>

Wikipedia [en línia]. [Data de consulta: 16 de març de 2017] <https://es.wikipedia.org/wiki/Diagrama_de_clases>

SitePoint(2015). [Data de consulta: 12 d'abril 2017] <https://www.sitepoint.com/devise-authentication-in-depth/>

http://guides.railsgirls.com [en línia]. [Data de consulta: 12 d'abril 2017] <http://guides.railsgirls.com/>

LingoHub [en línia]. [Data de consulta: 30 d'abril de 2017] <https://lingohub.com/frameworks-file-formats/rails5-i18n-ruby-on-rails/>

RichOnRails (2013) [en línia]. [Data de consulta: 2 de maig de 2017] <https://richonrails.com/articles/getting-started-with-paperclip>

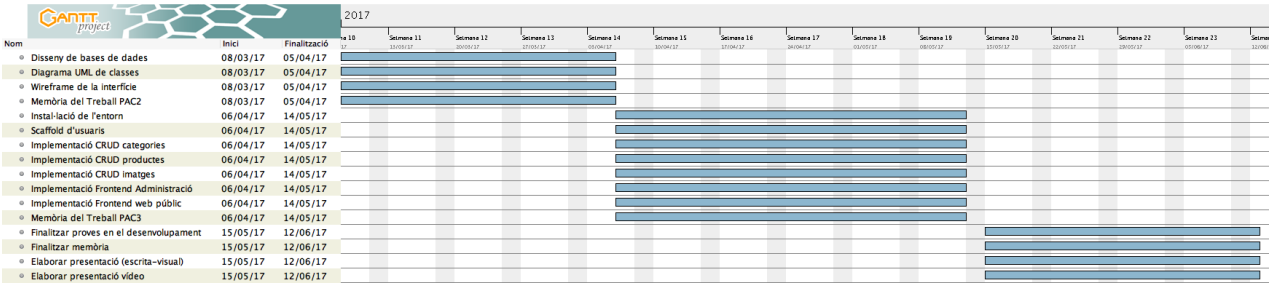
Wikipedia [en línia]. [Data de consulta: 20 de juny de 2017] <https://es.wikipedia.org/wiki/Sass_(lenguaje_de_hojas_de_estilo)>

Wikipedia [en línia]. [Data de consulta: 20 de juny de 2017] <https://es.wikipedia.org/wiki/Active_record>

Wikipedia [en línia]. [Data de consulta: 20 de juny de 2017] < https://es.wikipedia.org/wiki/Clave_primaria>

7. Annexos

ANNEX A



8. Vita

Sergi Beltran Prat, va néixer a Barcelona, Catalunya, el 13 de febrer de 1985. Fill de Sergi Beltran Bel i Dolors Prat Sada.

Va acabar els seus estudis de Màrqueting i Gestió Comercial reben la titulació de tècnic superior.

L'any 2010 va començar a cursar el Grau de Multimèdia a la Universitat Oberta de Catalunya.