U SCIENCE TECH
FACULTAT DE CIÈNCIES
I TECNOLOGIA
UVIC-UCC

Treball de Fi de Grau

Platonic

CMS modular basat en Laravel 5

Gerard Reches Urbano

Grau en Multimèdia

Tutor/a: Raymond Lagonigro Bertran

Vic , maig de 2016

Índex

RESUM TREBALL DE FI DE GRAU

GRAU EN MULTIMEDIA

Títol: *CMS modular basat en Laravel 5*
Paraules clau: *CMS*, *Laravel*, *Framework CSS*, *aplicacions web*, *open source*
Autor: Gerard Reches Urbano
Tutor: Raymond Lagonigro Bertran
Data: Juny de 2016

En aquest Treball de Fi de Grau es realitza el desenvolupament d'una aplicació web CMS modular i un *framework* CSS per dotar-la d'interfície gràfica. Els objectius principals són la modularitat de l'aplicació, el desenvolupament d'un panell d'administració i la cerca d'unes convencions CSS que facin el codi intuïtiu. La creació de l'aplicació es porta a terme utilitzant tecnologies web client-servidor i el *framework* PHP Laravel, mentre que el *framework* CSS es construeix sobre el preprocessador Less. Els resultats aconseguits són un CMS totalment modular i un *framework* CSS modern i adaptable. El projecte realitzat durant aquest treball disposa de moltes línies d'ampliació futures com a projecte de codi obert.

This Degree's Final Project develops a modular CMS web application and a CSS framework in order to provide a graphic interface to the application. The main goals of this project are the modularity of the application, the development of an admin panel and the research of some CSS conventions that makes an intuitive framework. The application is developed using client-server web technologies and Laravel PHP Framework, whilst the CSS framework is built using the Less preprocessor. The obtained results are a completely modular CMS and a responsive and modern CSS framework. This has multiple lines of future expansion as an open source project.

1 Introducció

Aquest és un Treball de Fi de Grau realitzat en el 4rt curs del Grau en Multimèdia de la Universitat de Vic – Universitat Central de Catalunya.

La primera idea sobre la qual realitzar el treball era un projecte transmèdia relacionat amb el videojoc Destiny, que constaria d'aplicacions adaptades per Android, iOS i navegador web així com també una identitat a les xarxes socials i fòrums per a la comunitat. Degut a que la popularitat d'un joc té un temps de vida limitat i després d'adonar-me que hi havia una necessitat real de disposar d'una eina per a la creació de revistes digitals per al món dels videojocs, vaig decidir encarar el meu TFG cap al desenvolupament d'una plataforma web amb aquesta finalitat. Donada la quantitat ingent de tasques a realitzar i per tal de treure més profit a les hores que s'hi dedicaria, la idea final va esdevenir realitzar una aplicació PHP modular reutilitzable acompanyada d'un nou framework CSS.

Formada per un nucli que servís com a base reutilitzable en diferents projectes, l'aplicació permetria a l'usuari personalitzar la seva plataforma web afegint els mòduls que més s'adaptessin a les seves necessitats. Aquests mòduls, creats per terceres persones, aprofitarien les funcionalitats que el nucli de l'aplicació proporciona, facilitant un entorn de desenvolupament perfectament funcional seguint les directrius indicades per aquest. Així doncs, es crearia una aplicació de codi obert que complís algunes funcions de Sistema Gestor de Continguts i que proporcionés un entorn de desenvolupament senzill per a la creació de nous mòduls que estenguessin les funcionalitats de la mateixa. En quant al framework CSS, per tal d'evitar ser un entre molts d'altres es buscaria un enfocament diferent al significat del mateix.

L'aplicació utilitza com a base el framework de PHP Laravel, juntament amb un paquet que permet separar una aplicació en mòduls. Consta d'un panell d'administració per a la gestió de continguts des del qual es poden administrar els

mòduls afegits a l'aplicació, la gestió d'usuaris de la plataforma i la personalització de les opcions de l'aplicació. El framework CSS està desenvolupat utilitzant el llenguatge Less, que es tracta d'un preprocessador de CSS.

En general es tracta d'un projecte amb moltes línies d'ampliació, que s'espera que segueixi evolucionant al publicar-se a GitHub com un projecte de codi obert i col·laboratiu, permetent a altres desenvolupadors aportar noves idees i actualitzacions per a l'aplicació. Encara que s'han de polir força coses, actualment ja permet desenvolupar alguns mòduls extra que funcionin amb l'aplicació.

En quant al framework CSS, cal dir que es tracta d'un enfocament diferent a l'habitual en quant al que hauria de proporcionar. L'objectiu és oferir un conjunt d'elements que s'utilitzen amb un llenguatge més natural. S'eviten els noms curts i els abreujaments, a favor de noms més llargs però senzills d'entendre al anomenar-se tal i com s'escriuria el que es vol obtenir, és a dir, ve a ser un llenguatge de programació comprensible que funciona més per intuïció que per coneixement. Una aposta arriscada, però que podria arribar a ser acceptada per part de la comunitat de desenvolupadors, especialment els principiants.

2 Marc teòric i conceptual

Amb aquest treball es busca crear una aplicació CMS modular a més d'un *framework* CSS dissenyat especialment per al CMS però alhora reutilitzable en altres projectes web.

A continuació en aquest apartat tractaré els conceptes principals del treball i les tecnologies fonamentals utilitzades en el projecte.

2.1 Definició i avantatges d'una aplicació modular

Una aplicació modular és un sistema de *software* que es pot separar en mòduls, on cada mòdul afegeix noves funcionalitats a l'aplicació base. Això aporta una gran flexibilitat a l'aplicació al permetre afegir només els mòduls necessaris, a més de facilitar el manteniment de cada una de les parts de l'aplicació i de fer-la més escalable.

2.2 Què és i què permet fer un CMS

Un CMS (*Content Management System*), també anomenat gestor de continguts, és un sistema de *software* que permet crear, editar i publicar continguts. Mentre que els primers CMS eren utilitzats per gestionar documents i fitxers en un mateix equip, la majoria de sistemes CMS actuals estan dissenyats exclusivament per gestionar continguts a la Web.

L'objectiu d'un CMS és el de proveir d'una interfície d'usuari intuïtiva per construir i modificar continguts d'una pàgina web. Cada CMS alhora proveeix la capacitat de permetre a un o més usuaris publicar actualitzacions en viu a la Web.

Els gestors de continguts estan disponibles com a aplicacions instal·lables i interfícies d'usuari basades en la web. La majoria de la gent prefereix una interfície web, ja que simplifica el procés d'actualització del lloc web. A més, la majoria de

CMS basats en la web són actualitzats automàticament, assegurant que tots els usuaris disposen de les eines més recents per gestionar els seus continguts.[1]

2.3 Gestors de continguts web més populars

Existeix una gran quantitat de gestors de continguts web, però hi ha alguns que destaquen per sobre de la resta per la seva popularitat. En especial cal destacar WordPress, Joomla i Drupal, que són els tres CMS més utilitzats actualment segons les estadístiques[2] reunides a la Figura 1.

Top in Content Management System · Week beginning May 30th 2016

Name	10k	100k	Million	Entire Web
WordPress	⬆2,712	⬇22,254	⬇300,850	⬇12,233,915
Joomla!	– 64	⬇1,404	⬇25,919	⬆2,018,386
Drupal	⬆596	⬆4,490	⬆32,161	⬆576,399

FIGURA 1. PRINCIPALS SISTEMES DE GESTIÓ DE CONTINGUTS

2.3.1 WordPress

Llargament el més popular dels CMS actuals, amb més de 12 milions de llocs web que l'utilitzen, és una eina tant per dissenyadors i desenvolupadors web com per gent amb coneixements mínims d'informàtica. Degut a la seva intuïtiva interfície d'usuari i a totes les possibilitats que ofereix és un gestor de contingut fàcil d'utilitzar pels usuaris finals.

Es tracta d'un CMS modular, ja que permet instal·lar diferents *plugins* (components que afegeixen funcionalitats extra) a més de que existeixen multitud de temes que poden canviar dràsticament l'aspecte de la pàgina web així com també afegir nous tipus de continguts que poden ser gestionats pel CMS.

[1] http://techterms.com/definition/cms (Consulta: 30 de maig de 2016).
[2] http://trends.builtwith.com/cms (Consulta: 30 de maig de 2016).

Tots aquests *plugins* i temes són desenvolupats per terceres persones amb WordPress com a objectiu, utilitzant les eines que aquest ofereix per a tals propòsits.

2.3.2 Joomla

Joomla és un CMS amb un nivell de dificultat d'utilització major que WordPress, sobrepassant els 2 milions de llocs web que l'utilitzen, està més enfocat a desenvolupadors web que a usuaris finals, tot i que aquests últims poden igualment gestionar els continguts de la web sense gran complicació, però amb una interfície d'usuari no tant clara com la de WordPress.

Al igual que WordPress també és modular permetent instal·lar *plugins* i temes, però com a avantatge es podria considerar el fet de que per defecte incorpora la gestió de continguts en múltiples idiomes entre altres característiques.

2.3.3 Drupal

Drupal ocupa el tercer lloc dels CMS més utilitzats a la Web a l'actualitat amb prop de 600 mil llocs web que l'utilitzen. Amb un plantejament molt més modular, després de la instal·lació t'insta a estendre l'aplicació amb tots els mòduls que puguis necessitar. No precisament pensat per usuaris finals, el procés de instal·lació i configuració seria recomanable que el realitzés algú amb coneixements en aplicacions d'internet. En quant a la interfície d'usuari és més primitiva que la de WordPress i similar però probablement menys intuïtiva que la de Joomla.

2.4 Què és un framework CSS

Un *framework* CSS és un conjunt de fitxers de codi, principalment CSS, que, generalment comprimit en un sol fitxer CSS, pot ser utilitzat per ajudar en el desenvolupament de pàgines web com a base per començar a maquetar visualment un lloc web.

El fitxer de codi CSS que s'enllaça a la pàgina web conté un ampli conjunt d'estils CSS, dels quals alguns són aplicats per defecte en elements HTML i molts altres són utilitzables com a classes CSS.

2.5 Actualitat dels framework CSS

Existeixen molts *frameworks* CSS, però no són gaires els que tenen certa popularitat entre els desenvolupadors. Això és probablement degut en part a que les diferències entre uns i altres són poques i poc rellevants. Crear un nou *framework* CSS i seguir les mateixes pautes que els ja existents no representaria cap innovació encara que els estils proposats fossin diferents, pel que quedaria entre la multitud de *frameworks* CSS clònics i tindria una probabilitat pràcticament nul·la de destacar. Així doncs, és necessària una proposta que destaqui en algun sentit.

2.6 Cerca d'un tret diferencial

Una de les raons que es poden tenir en consideració al qüestionar-se per què els desenvolupadors solen utilitzar sempre el mateix *framework* CSS és la corba d'aprenentatge. Aprendre a utilitzar un *framework* amb naturalitat requereix de cert temps dedicat a l'aprenentatge del mateix, i tant el temps com la corba d'aprenentatge són factors importants a tenir en compte.

D'aquest raonament em sorgeix la següent pregunta:

"Quina acceptació tindria entre els desenvolupadors un *framework* CSS amb una corba d'aprenentatge nul·la o pràcticament nul·la?"

Amb corba d'aprenentatge nul·la em refereixo al fet de que la corba d'aprenentatge sobre el temps esdevingués plana ja al iniciar l'aprenentatge, significant que només començar ja es disposaria dels coneixements necessaris. A la Figura 2 es pot veure en primer lloc un exemple del que podria ser una corba

d'aprenentatge habitual, i a continuació es mostra el que anomeno una corba d'aprenentatge nul·la.

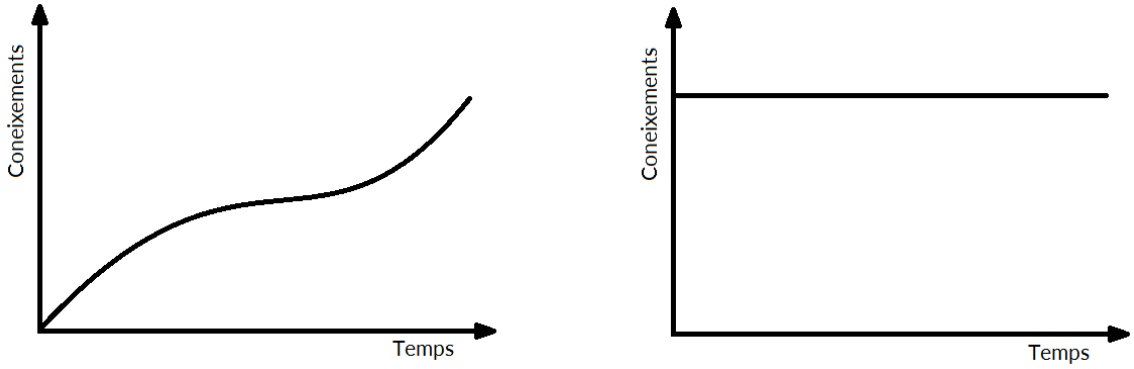

FIGURA 2. CORBA D'APRENENTATGE HABITUAL I CORBA D'APRENENTATGE IDEAL

En primera instancia pot semblar un cas idíl·lic, i de fet ho és, ja que cada persona és diferent, pel que hi pot haver molts factors que, donats en diferents individus, impedeixin complir en la seva totalitat l'objectiu d'aconseguir una corba d'aprenentatge nul·la. Tot i així, això no significa que no sigui possible reduir potencialment el temps necessari per assolir el punt màxim de coneixements de la corba d'aprenentatge. A la Figura 3 es pot veure un exemple d'aquest cas.

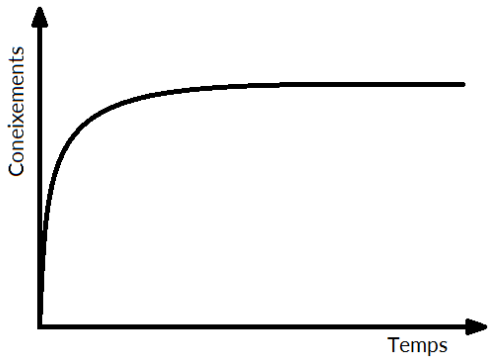

FIGURA 3. CORBA D'APRENENTATGE AMB UNA RÀPIDA ADQUISICIÓ DE CONEIXEMENTS

Per acostar-se a aquest propòsit cal utilitzar uns coneixements dels que la gent ja disposi i proposar unes pautes que indiquin com aplicar aquests coneixements al

cas d'un *framework* CSS. En conclusió als raonaments plantejats, apostaré pel llenguatge natural, i quan parlo de llenguatge natural em refereixo al llenguatge que utilitzem les persones tant per pensar com per comunicar-nos.

2.7 Tecnologies fonamentals per al desenvolupament del projecte

2.7.1 Tecnologia web client-servidor

La tecnologia client-servidor (Figura 4) consisteix en una arquitectura d'execució d'aplicacions en la que un equip client sol·licita uns recursos o serveis a un equip servidor i aquest li dona una resposta.

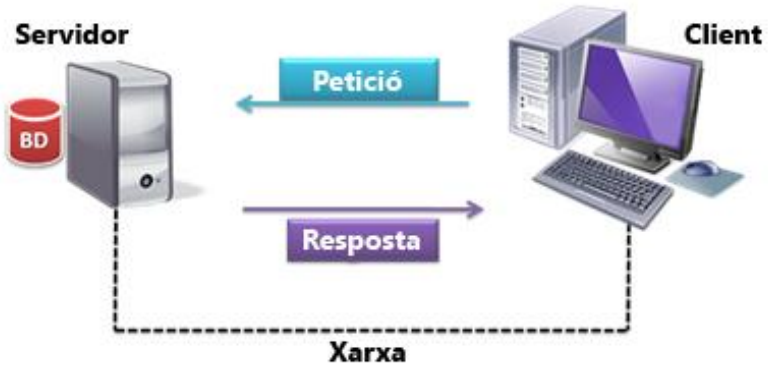

FIGURA 4. ESQUEMA DE FUNCIONAMENT DE LA TECNOLOGIA CLIENT-SERVIDOR

Els avantatges d'aquesta tecnologia consisteixen en mantenir els recursos centralitzats, major seguretat, administració només a nivell de servidor i una xarxa escalable.[3]

És una tecnologia òptima per a desenvolupar un CMS gracies als avantatges esmentats.

[3] http://es.ccm.net/contents/148-entorno-cliente-servidor (Consulta: 26 de maig de 2016).

2.7.2 HTML

El llenguatge HTML (HyperText Markup Language) és l'element de construcció més bàsic d'una pàgina web i s'utilitza per crear i representar visualment una web. Serveix per determinar l'estructura i el contingut d'una pàgina web.[4]

Es tracta de l'element fonamental en tot projecte web i generalment es veu acompanyat per altres llenguatges que el complementen.

2.7.3 CSS

"CSS (Cascading Style Sheets), és un mecanisme simple que descriu com es mostrarà un document a la pantalla, o com s'imprimirà, o fins i tot com serà pronunciada la informació present en aquell document a través d'un dispositiu de lectura. Aquesta forma de descripció d'estils ofereix als desenvolupadors el control total sobre l'estil i format dels seus documents." [5]

Més comunament conegut com el llenguatge de maquetació web que permet donar estil visual als elements HTML, així com modificar el disseny de la pàgina web sense modificar-ne el seu contingut.

Imprescindible el seu ús en la gran majoria de projectes web degut a les necessitats estètiques i de disseny, es tracta d'un dels elements essencials d'aquest projecte, ja que és el llenguatge base del *framework* CSS.

2.7.4 PHP

PHP (*Pre Hypertext Processor*) és un llenguatge de codi obert molt popular especialment adequat pel desenvolupament web i que pot ser incrustat en HTML.

[4] https://developer.mozilla.org/es/docs/Web/HTML (Consulta: 26 de maig de 2016).
[5] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo (Consulta: 26 de maig de 2016).

El codi és executat en el servidor, generant HTML i enviant-lo al client. El client rebrà el resultat d'executar el codi, però no se sabrà quin era el codi original.[6]

Aquest és el llenguatge de banda servidor escollit per al projecte degut a la seva popularitat, la facilitat per trobar-ne informació i a que ja hi he treballat anteriorment.

[6] http://php.net/manual/es/intro-whatis.php (Consulta: 25 de maig de 2016).

3 Objectius

3.1 Objectius de l'aplicació CMS modular

El primer i principal objectiu del treball és crear una aplicació web reutilitzable amb un panell d'administració i un sistema d'usuaris.

Aquesta aplicació ha de ser modular i els mòduls han de ser gestionables des del panell d'administració.

Els usuaris han de poder-se registrar, iniciar sessió i tancar sessió de l'aplicació.

L'aplicació ha de funcionar com a gestor de continguts, permetent als usuaris crear, editar i publicar continguts.

Altres objectius són poder configurar algunes opcions de l'aplicació i poder personalitzar alguns estils CSS utilitzant aquestes opcions.

3.2 Objectius del framework CSS

Un dels principals objectius del treball és crear un *framework* CSS per proveir a l'aplicació CMS modular d'una interfície gràfica avançada, moderna i pròpia.

Com a *framework* CSS ha de ser fàcil d'implementar en qualsevol projecte web i mantenir certa harmonia visual entre els seus components.

El *framework* ha d'incorporar característiques per al disseny *responsive* (adaptatiu a les diferents característiques de pantalla) i pensat en primer lloc per als dispositius mòbils. A més ha de ser *cross-browser*, és a dir, compatible entre els diferents navegadors web principals.

L'objectiu principal del *framework* CSS és aconseguir una sintaxi intuïtiva que faciliti potencialment el seu aprenentatge als usuaris que l'utilitzin.

Altres objectius a tenir en compte són mantenir un codi lleuger i no requerir de llenguatge JavaScript per al funcionament de cap dels seus components.

3.3 Objectius personals

La motivació principal al realitzar aquest treball és l'oportunitat de posicionar-me dins el sector del disseny i desenvolupament web, creant projectes de codi obert que puguin ser d'utilitat per a altres persones, alhora que aporto un important valor afegit al meu currículum al poder mostrar aquests projectes.

Adquirir la capacitat de prendre decisions importants en projectes que han de ser utilitzats per altres desenvolupadors, així com també ampliar els meus coneixements del *framework* Laravel i del llenguatge PHP.

Conèixer les tecnologies, software i conceptes més utilitzats en el disseny i desenvolupament web actualment i descobrir les solucions més actuals per alguns dels problemes més habituals del llenguatge CSS.

Crear una aplicació CMS i un *framework* CSS propis i moderns amb els que estigui totalment familiaritzat i que pugui utilitzar per agilitzar el desenvolupament de futurs projectes i el seu manteniment.

4 Marc metodològic

4.1 Platonic CMS

Platonic CMS és l'aplicació desenvolupada en aquest treball. El nom sorgeix d'una pluja d'idees en la cerca d'un nom per a una aplicació de *software* que serveixi com a plataforma web (d'aquí la part de *Plat-* del nom). La finalitat d'aquest *software* és oferir més facilitats respecte a altres CMS's i *frameworks*, sovint proposant-se uns objectius que s'acosten a un cas idíl·lic; d'aquí sorgeix el nom de Platonic.

4.1.1 Creació d'un imagotip

En el procés de creació d'un imagotip es buscava transmetre una imatge moderna i elegant, però alhora una mica desenfadada. Alhora es volia referenciar d'alguna manera que es tracta principalment d'una aplicació modular.

FIGURA 5. IMAGOTIP DE PLATONIC

El resultat és el vist a la Figura 5, que també disposa de versions en blanc i en negre pels diversos usos que se li puguin donar. L'isotip representa la modularitat i les relacions entre mòduls, mentre que el logotip utilitza una tipografia moderna i elegant, de tipus *Sans Serif* per tal de treure serietat al tractar-se d'un projecte en constant evolució.

4.1.2 Less

Less és un preprocessador de CSS, el que significa que estén el llenguatge CSS, afegint funcionalitats que permeten utilitzar variables, mescles, funcions i moltes

altres tècniques per tal de crear un CSS més fàcil de mantenir, de personalitzar i d'ampliar.[7]

Utilitzar aquest llenguatge és un punt clau per assolir l'objectiu d'aconseguir uns estils personalitzables per a algú que desconegui el funcionament del llenguatge CSS, a més de mantenir el *framework* CSS més net i escalable.

4.1.3 Laravel PHP Framework

Laravel és un *framework* per a crear aplicacions web amb una sintaxi elegant i expressiva. Es va crear amb la certesa que el desenvolupament ha de ser una experiència de la que gaudir alhora que creativa per tal de ser totalment satisfactòria. Laravel intenta eliminar els maldecaps del desenvolupament facilitant tasques habituals en la majoria de projectes web, com pot ser l'autentificació d'usuaris, rutes de l'aplicació, sessions, col·locar processos en cua i emmagatzemament en cache.

"Laravel is accessible, yet powerful, providing tools needed for large, robust applications. A superb inversion of control container, expressive migration system, and tightly integrated unit testing support give you the tools you need to build any application with which you are tasked." [8]

Es tracta d'un dels *frameworks* PHP més populars del moment i continua evolucionant dia a dia amb noves actualitzacions. Permet implementar el patró MVC (explicat en el següent apartat) i proporciona eines que faciliten i agilitzen enormement el desenvolupament. A més és totalment extensible al permetre instal·lar paquets PHP que afegeixen noves funcionalitats que poden ser necessàries en segons quins projectes.

[7] http://lesscss.org/ (Consulta: 26 de maig de 2016).
[8] https://github.com/laravel/laravel (Consulta: 25 de maig de 2016).

He tingut el plaer de treballar anteriorment amb aquest *framework* i l'experiència va ser sobradament satisfactòria com per escollir el seu ús per a aquest projecte.

4.1.4 Patró MVC

El patró MVC (Model-Vista-Controlador) és una proposta d'arquitectura de software que fomenta la facilitat de manteniment i la reutilització del codi a més de la separació de conceptes (veure Figura 6) per tal de crear un software més robust i amb un cicle de vida més adequat.[9]

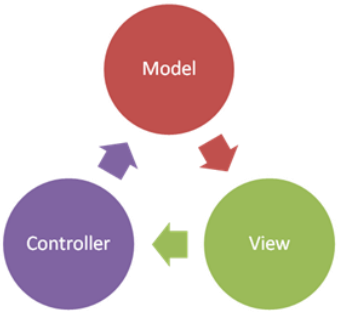

FIGURA 6

Estaríem abusant d'aquest terme si diguéssim que Laravel segueix un patró MVC. Si bé Laravel utilitza models, vistes i controladors, també trobem que intervenen molts altres elements en l'arquitectura del seus projectes, pel que no és purament un MVC. Taylor Otwell, creador de Laravel, va confirmar aquest mateix fet en una conversació de Twitter [10] (veure Figura 7).

[9] http://www.desarrolloweb.com/articulos/que-es-mvc.html (Consulta: 26 de maig de 2016).
[10] Twitter: Popular xarxa social en que les publicacions no poden ocupar més de 140 caràcters.

FIGURA 7. TAYLOR OTWELL COMUNICANT QUE LARAVEL NO ÉS UN FRAMEWORK MVC

De fet, a la versió 5 de Laravel va deixar d'existir el directori específic per als models, deixant a lliure elecció del desenvolupador l'estructura del seu projecte. Així doncs, tot i que en el projecte s'incorporen els elements que formen aquest patró, no ens referirem més a Laravel com si fos un *framework* MVC, ja que és molt més que això.

4.1.5 Entorn de desenvolupament

Per complir els requisits tècnics de l'aplicació i facilitar el desenvolupament del projecte és necessari un entorn de desenvolupament adequat.

Es pot utilitzar algun IDE (*Integrated Development Environment*) com PHPStorm per al desenvolupament amb Laravel, però es tracta de programes comercials i el que ofereixen no mereix el seu cost en comparació a altres alternatives gratuïtes.

4.1.5.1 Sublime Text

Al treballar amb múltiples llenguatges web alhora el més recomanable sol ser utilitzar un dels molts editors de texts que existeixen. La decisió de quin utilitzar

és una qüestió de preferència personal, però és bo tenir en compte què ens ofereix cada un d'ells i quin s'adapta més al nostre *workflow* [11].

Personalment l'editor que s'adapta més al meu *workflow* és el Sublime Text degut a la capacitat de personalització de la que disposa gracies al Package Control, que permet instal·lar i gestionar *plugins* [12] dissenyats especialment per a aquest editor de text.

Altres editors de text populars com Brackets i Atom també ofereixen una important capa de personalització a base de *plugins*, però estan més enfocats al desenvolupament *Front-End* i per defecte no ofereixen suport per PHP.

Deixant a una banda la capacitat de personalització, Sublime Text tot i no ser un IDE és una potent eina que ofereix diverses funcionalitats que agilitzen molt les tasques de programació una vegada les coneixes, com l'ús d'*snippets* [13], les seleccions múltiples i la cerca de fitxers que continguin una paraula indicada.

4.1.5.2 Composer

Composer és una eina per a l'administració de dependències en PHP. Permet declarar les dependències de les que depèn el teu projecte i s'encarrega d'instal·lar-les i actualitzar-les per tu.[14]

Laravel utilitza aquesta eina en moltes situacions, i especialment durant el procés de desenvolupament d'un projecte.

4.1.5.3 Vagrant

Vagrant és una eina per crear entorns de desenvolupament al complet. Amb un workflow fàcil de seguir i enfocat a l'automatització, Vagrant redueix el temps necessari per configurar un entorn de desenvolupament, augmenta la qualitat del

[11] Workflow: Fluxe de treball.
[12] Plugin: Complement de software que aporta noves funcionalitats.
[13] Snippet: Tros de codi reutilitzable a partir d'un disparador.
[14] https://getcomposer.org/doc/00-intro.md (Consulta: 27 de maig de 2016).

desenvolupament i la producció, i converteix l'excusa "en el meu equip funciona" en una relíquia del passat.[15]

4.1.5.4 VirtualBox

Vagrant requereix d'una màquina virtual per establir un entorn de desenvolupament, i amb aquesta finalitat ofereix dues opcions de software a escollir: VMware o bé VirtualBox. La principal diferència és que VMware és comercial, mentre que VirtualBox és lliure en llicència GPL.

En aquest cas en concret ens decantem per VirtualBox per qüestió de cost ja que no té gran importància quin dels dos escollim.

4.1.5.5 Homestead

Laravel s'esforça en fer el més agradable possible l'experiència de desenvolupar en PHP, incloent el teu entorn de desenvolupament local. Vagrant proveeix una manera simple i elegant de gestionar i aprovisionar Màquines Virtuals.

Laravel Homestead és un entorn de desenvolupament oficial de Vagrant que et proveeix de tot el software necessari per desenvolupar amb Laravel sense la necessitat de que instal·lis tot aquest software al teu equip. Els entorns de desenvolupament de Vagrant estan ideats per ser d'un sol ús, si qualsevol cosa va malament pots destruir i tornar a crear l'entorn en pocs minuts.

El següent software està inclòs en la instal·lació de Laravel Homestead:[16]

- Ubuntu 14.04
- Git
- PHP 7.0
- HHVM
- Nginx
- MySQL

[15] https://www.vagrantup.com/about.html (Consulta: 27 de maig de 2016).
[16] https://laravel.com/docs/5.2/homestead (Consulta: 27 de maig de 2016).

- MariaDB
- Sqlite3
- Postgres
- Composer
- Node (Amb PM2, Bower, Grunt i Gulp)
- Redis
- Memcached
- Beanstalkd

El procés d'instal·lació de Laravel Homestead està detalladament explicat a l'enllaç https://laravel.com/docs/master/homestead#installation-and-setup.

4.1.6 Git: Sistema de Control de Versions

Els Sistemes de Control de Versions són una categoria d'eines de *software* que ajuden als equips de desenvolupadors a gestionar el codi font al llarg del temps. El *software* de control de versions manté el seguiment de qualsevol modificació del codi en un tipus de base de dades especial. Si es comet un error, els desenvolupadors poden tornar enrere en el temps i comparar el codi actual amb antigues versions del codi per ajudar a solucionar el problema sense interrompre el desenvolupament dels altres membres de l'equip.[17]

Git és un Sistema de Control de Versions d'ús lliure i codi obert dissenyat per gestionar eficientment des de petits fins a grans projectes.[18]

Utilitzo Git com a eina de control de versions per al projecte i allotjo el repositori a Bitbucket com a privat mentre duri la fase de desenvolupament, ja que no vull que sigui públic mentre no hi hagi una versió completament estable. Una vegada arribats a la primera versió publicable serà traslladat al servei de GitHub com a repositori públic i seguirà rebent actualitzacions al llarg del temps.

[17] https://www.atlassian.com/git/tutorials/what-is-version-control (Consulta: 31 de maig de 2016).
[18] https://git-scm.com/ (Consulta: 31 de maig de 2016).

4.1.7 Dependències PHP seleccionades

Laravel proporciona un fitxer *composer.json* en el que es poden afegir dependències PHP requerides per al projecte i ampliar així les funcionalitats bàsiques que incorpora l'aplicació. En futures actualitzacions pot ser que facin falta més, però fins ara són només quatre les dependències que he necessitat:

- caffeinated/modules

  Permet separar l'aplicació de Laravel en diferents mòduls. Cada mòdul és totalment auto-suficient concedint l'habilitat de simplement col·locar un mòdul dins l'aplicació per al seu ús. Els mòduls disposen del seu propi fitxer *module.json* en el que definir algunes propietats del mòdul. Un fitxer de configuració permet també configurar algunes opcions generals dels mòduls com la seva localització dins l'estructura del projecte.

  Constitueix una part fonamental de l'aplicació modular.

- oyejorge/less.php

  Aquesta llibreria és una conversió del processador JavaScript de Less a una versió en PHP d'ell mateix. Permet compilar codi Less així com modificar-ne les variables a través de PHP abans de compilar el CSS.

- laravelcollective/html

  Recull algunes funcionalitats útils de Laravel 4 que ja no tenen suport oficial a Laravel 5.

- creativeorange/gravatar
  Una petita dependència PHP dissenyada per Laravel que ajuda a adquirir una imatge de Gravatar[19] associada a una direcció e-mail. Utilitzada per assignar una imatge de perfil automàticament al crear un nou usuari.

[19] https://es.gravatar.com/ (Consulta: 1 de juny de 2016).

4.1.8 Estructura del projecte

L'estructura de Platonic CMS no dista massa de l'estructura original de Laravel tot i que s'ha simplificat. La carpeta *database* s'elimina de l'estructura general i també s'elimina el fitxer *gulpfile.js* ja que no s'utilitzarà, i en cas de ser necessari es crearà automàticament. Aquestes són les úniques dues diferències que es poden trobar a primera vista.

Dins la carpeta *app* és on es troba l'estructura de fitxers de l'aplicació a desenvolupar sobre un projecte de Laravel. En el cas de Platonic CMS només consta d'una carpeta *Core* (Veure Figura 8) que és el mòdul base de l'aplicació. En paral·lel a aquest anirien els demés mòduls que es volguessin utilitzar en funció de les necessitats de cada client.

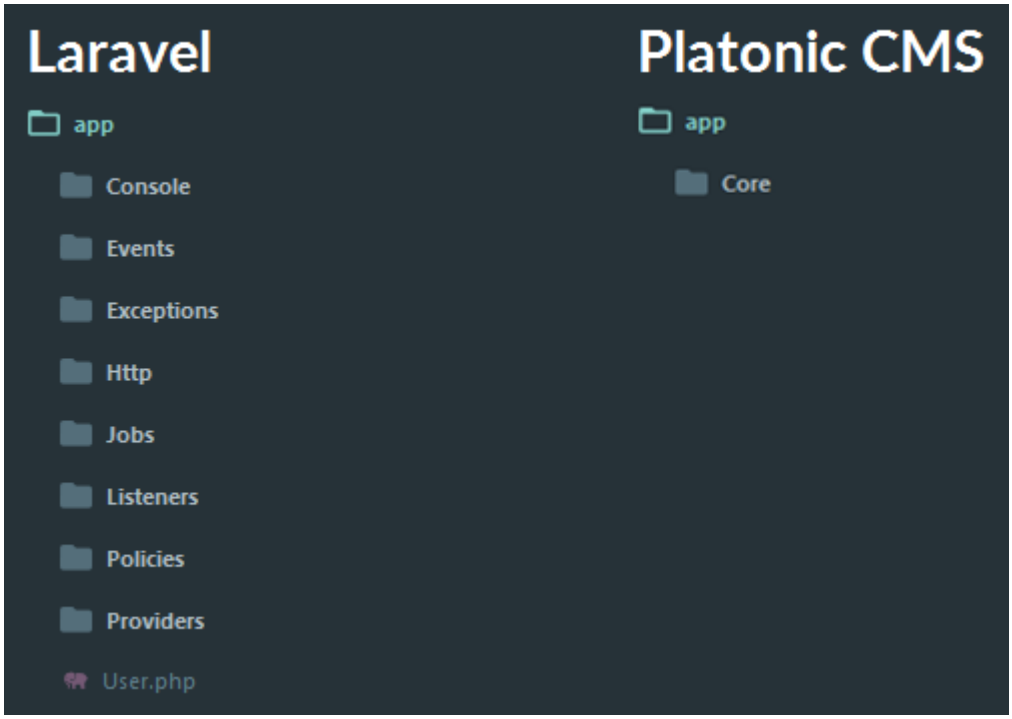

FIGURA 8. COMPARACIÓ DE LA CARPETA *APP* ENTRE LARAVEL I PLATONIC CMS

4.1.9 Estructura dels mòduls

Cada mòdul està contingut dins una carpeta amb el seu nom. Dins aquesta carpeta s'hi troba l'estructura que es pot veure a la Figura 9.

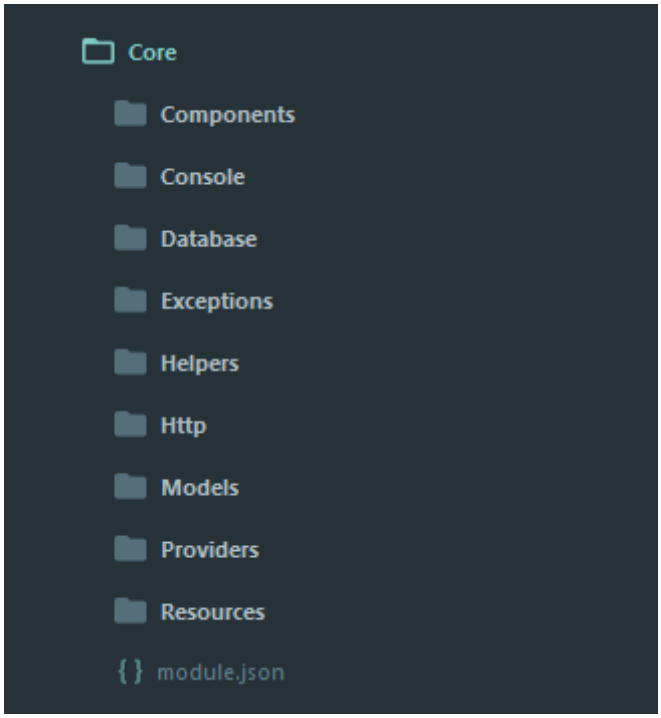

FIGURA 9. ESTRUCTURA D'UN MÒDUL DE PLATONIC CMS

Habitualment la carpeta *Http* conté un fitxer *routes.php* en el que es defineixen les rutes de l'aplicació. Els mòduls de Platonic CMS tenen dos fitxers de rutes agrupats en el directori *Http/Routes*, que són el *module.php*, en el que es defineixen les principals rutes que proporciona el mòdul, i el *dashboard.php*, en el que s'inclouen les rutes de gestió de continguts per al panell d'administració.

4.1.10 Convencions de la base de dades

De ben segur la majoria de mòduls creats per a Platonic CMS necessitaran actualitzar o crear noves taules en la base de dades. Per tal de facilitar el manteniment i l'ús de la base de dades es defineixen unes convencions que són les següents:

- Els noms de les taules s'escriuen en format *snake_case* i amb el *slug* del mòdul com a prefix.
- Els noms de les columnes s'escriuen en format *camelCase* i, en cas de ser una columna afegida a una taula que no és del propi mòdul, amb el *slug* del mòdul com a prefix.

En cas de ser necessari eliminar manualment les restes que un mòdul ha deixat a la base de dades, aquestes convencions ajuden a identificar ràpidament quines són les taules i/o columnes que pertanyien al ja inexistent mòdul.

4.1.11 Estructura de la base de dades

L'estructura de la base de dades del mòdul base (Veure Figura 10) és en un principi simple, però alhora extensible.

La taula d'usuaris és força completa per defecte, guardant força informació de cada usuari per utilitzar en diversos casos, però pot ser igualment ampliada per altres mòduls.

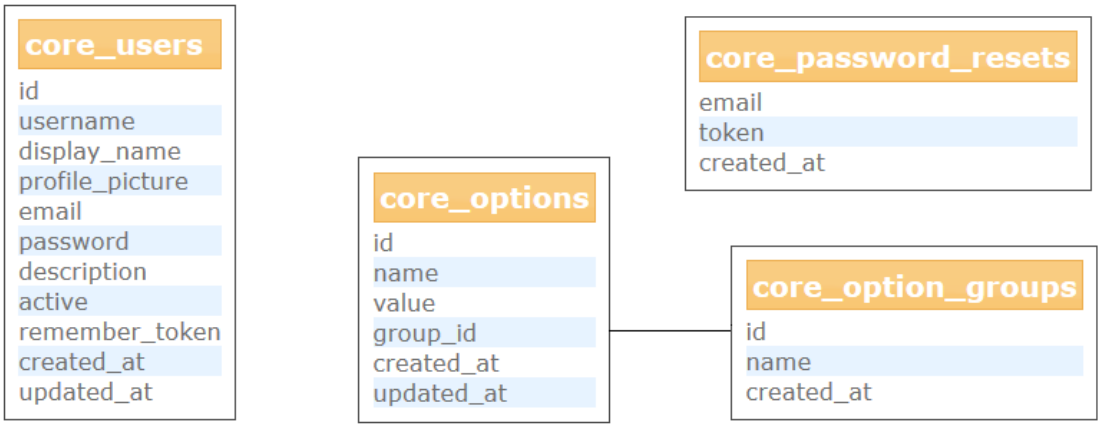

FIGURA 10. ESTRUCTURA DE LA BASE DE DADES DEL MÒDUL BASE

La taula d'opcions conté un nom i un valor, així com les dates de creació i última actualització de cada opció. Conté també una clau forana que referencia a un identificador de la taula de grups d'opcions.

També hi ha una petita taula per gestionar les sol·licituds de canvi de contrasenya.

4.1.12 Panell d'administració

Platonic CMS, com a sistema gestor de continguts, requereix d'una interfície en la que poder gestionar els continguts del lloc web, sent la millor opció un panell d'administració.

El panell d'administració creat per a l'aplicació segueix el format de la majoria de panells d'administració actuals, consistint en una barra lateral amb els menús per gestionar els diferents tipus d'opcions i continguts del CMS, una barra superior amb opcions que han de ser accessibles des de qualsevol pàgina del panell d'administració i la resta de pàgina per a mostrar els continguts de cada apartat. Tot això mirant d'oferir un disseny i una gama de colors moderns (Veure Figura 11).

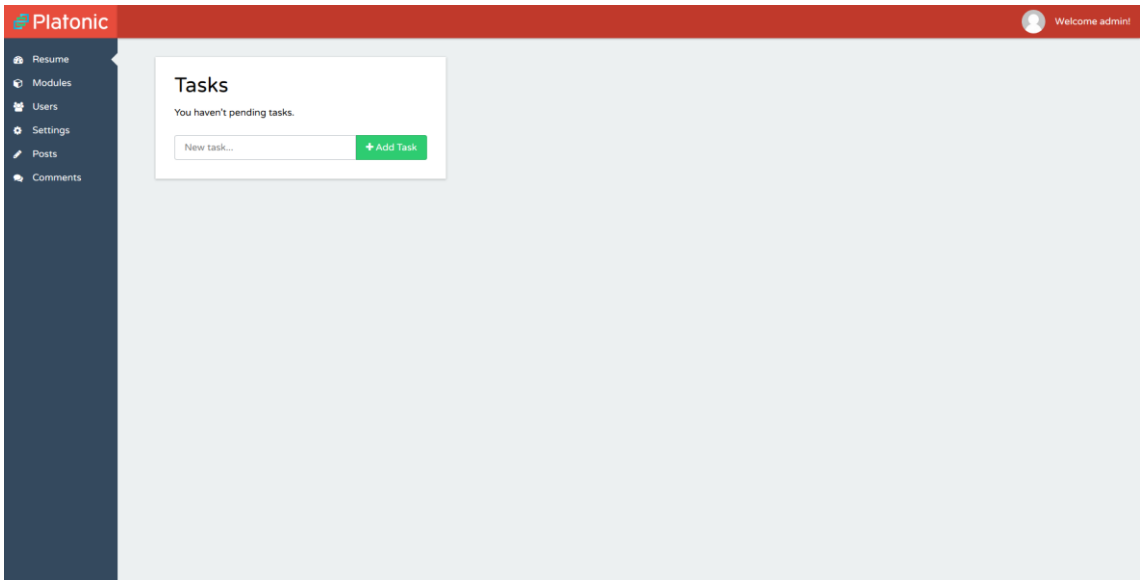

FIGURA 11. PÀGINA PRINCIPAL DEL PANELL D'ADMINISTRACIÓ DE PLATONIC CMS

La gràcia del menú lateral del panell d'administració és que és dinàmic en funció dels mòduls de l'aplicació activats i tot això es fa sense manipular ni guardar elements de menú en una taula de la base de dades. Això s'ha aconseguit mitjançant l'eina de Laravel *IoC container* que pot ser utilitzat com a *singleton*. Un *singleton* permet crear una instància única d'una classe que és accessible des de tota l'aplicació.

Aprofitant que els mòduls de Platonic CMS tots tenen el seu propi Proveïdor de Serveis que és carregat automàticament en cada execució de l'aplicació, es crea una instancia en el *singleton* de la classe *DashboardMenu* (Veure Figura 12) en el mòdul base.

```php
class DashboardMenu{

    protected $items = array();

    public function addItem($title, $icon, $route){
        $item = new DashboardMenuItem($title, $icon, $route);
        array_push($this->items, $item);
    }
}
```

FIGURA 12. CLASSE DASHBOARDMENU

Aquesta instancia conté una taula d'objectes de la classe *DashboardMenuItem*, que s'omple en el mètode *boot()* del Proveïdor de Serveis de cada mòdul de l'aplicació (Veure Figura 13).

```php
public function boot(){
    DashboardMenu::addItem(
        'Users',
        'fa fa-users',
        route('dashboard::modules::index')
    );
}
```

FIGURA 13. AFEGINT UN ELEMENT DE MENÚ EN EL MÈTODE *BOOT()*

4.1.13 Autentificació i gestió d'usuaris

El sistema d'usuaris inclou el registre dels mateixos així com inici i tancament de sessió. Les rutes per a cada una d'aquestes accions estan dissenyades de manera que en cas necessari puguin ser modificades.

Els usuaris poden ser gestionats des del panell d'administració pels administradors del lloc web.

4.1.14 Gestió de mòduls

Com a CMS modular els mòduls són una part fonamental de l'aplicació i és necessari poder-los gestionar, el que significa que com a mínim s'ha de poder activar i desactivar cadascun d'ells.

El panell d'administració disposa d'un apartat en el menú que permet veure els mòduls instal·lats i la seva informació, i com a accions permet activar-los (en cas d'estar desactivats) i desactivar-los (en cas d'estar activats). El color sobre el que es troba la informació del mòdul serveix com a indicador del seu estat (Veure Figura 14).

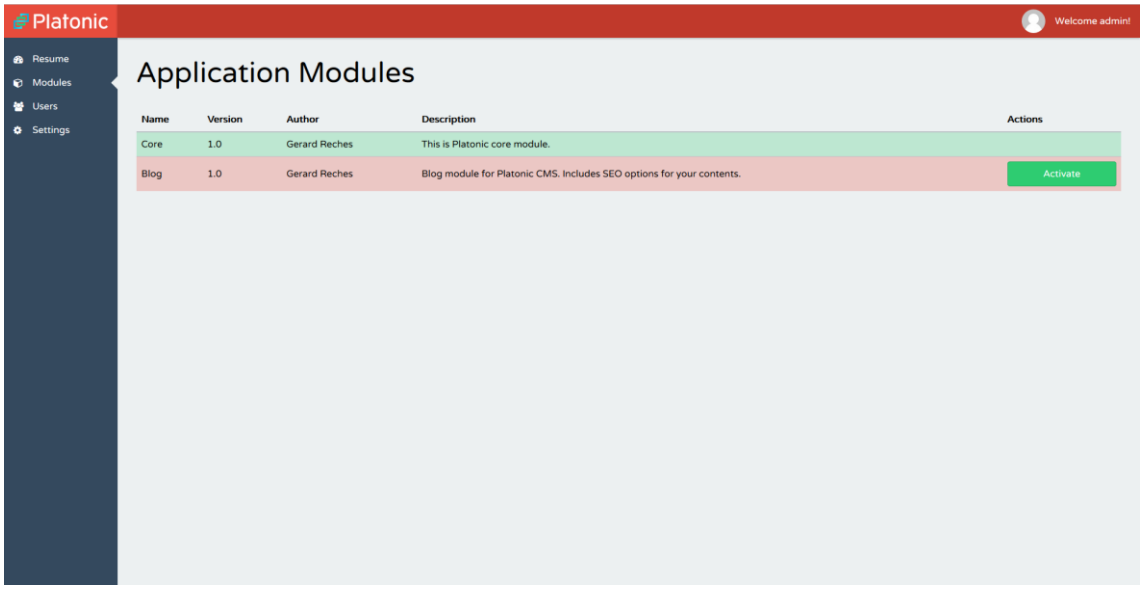

FIGURA 14. GESTIÓ DE MÒDULS DE PLATONIC CMS

El *Core*, com a mòdul base, no pot ser desactivat ja que deixaria de funcionar l'aplicació.

La gestió de mòduls s'aconsegueix mitjançant un arxiu *.json* generat per la dependència caffeinated/modules en que queda registrat l'estat actual de cada mòdul.

4.1.15 Gestió d'opcions del CMS

Les opcions configurables i personalitzables de l'aplicació es poden gestionar des del seu apartat corresponent del menú del panell d'administració. Cada opció és una estructura clau-valor que pertany a un grup d'opcions per tal d'organitzar millor aquestes i saber diferenciar correctament el seu ús.

4.1.16 CSS personalitzable del lloc web

Existeix un grup d'opcions del CMS anomenat "*Site CSS*". Aquestes opcions contenen valors de variables de l'arxiu *site.less* i els valors donats sobreescriuen els valors de les variables del fitxer al compilar-lo, procés que es fa automàticament al actualitzar les opcions del CMS.

Aquest sistema permet realitzar algunes modificacions del CSS del lloc web dinàmicament, permetent personalitzar els estils CSS des del panell d'administració sense tocar-ne el codi.

4.1.17 Funcionalitats com a framework en la creació de mòduls

El mòdul base de Platonic CMS conté una sèrie de fitxers proveïdors de serveis que faciliten el desenvolupament d'altres mòduls al encarregar-se de carregar els arxius de certes carpetes de l'estructura d'un mòdul, estalviant així la programació de varis proveïdors de serveis en els mòduls. Les carpetes de l'estructura d'un mòdul quals fitxers són carregats per Platonic CMS són les següents:

- Database/Factories

- Http/Routes
- Components/Facades
- Helpers

Tots els fitxers del mòdul continguts en aquests directoris es carregaran automàticament en l'execució de l'aplicació.

4.2 Platonic CSS Framework

Un dels objectius del treball és el de crear un *framework* CSS paral·lelament a l'aplicació modular i utilitzar aquest per a la mateixa.

Així mateix cal destacar que el *framework* CSS de Platonic, tot i estar integrat dins la pròpia aplicació, està dissenyat per poder ser utilitzat en qualsevol projecte web, de la mateixa manera que s'utilitzen altres *frameworks* CSS populars com per exemple Bootstrap.

4.2.1 Funcionament del llenguatge natural i convencions escollides

Anteriorment, en el marc teòric, he parlat sobre la necessitat d'aportar uns valors innovadors diferenciats de la resta de *frameworks* CSS del mercat. La proposta de valor escollida fomenta una sintaxi molt més intuïtiva basada en el nostre llenguatge natural. Cal recordar que amb llenguatge natural em refereixo al llenguatge que utilitzem per comunicar-nos.

Es requereixen certes convencions que cal tenir en compte per tal de que el llenguatge natural utilitzat al *framework* Platonic CSS resulti intuïtiu. A continuació les cito i explico el perquè les he escollit:

- El llenguatge utilitzat és l'anglès.

  Es tracta d'un *framework* CSS de codi obert, i l'anglès és el llenguatge més estès a nivell global i conegut per la gran majoria de desenvolupadors, degut sobretot a que la majoria de documentacions de software estan escrites en

aquest idioma. No utilitza caràcters especials, com per exemple accents, pel que és un idioma molt pla que facilita l'escriptura.

- Les paraules s'escriuen senceres, sense abreujaments.

  Es busca aconseguir un llenguatge natural i entenedor, pel que les abreviacions anirien en contra dels principis d'aquest *framework*. Classes CSS amb noms més llargs proporcionen una millor comprensió.

- Una classe CSS pot estar formada per més d'una paraula.

  Per suposat no tot es pot definir amb una sola paraula. Però això no justifica que s'utilitzin més paraules de les necessàries per definir una classe CSS.

- Sempre s'escriuen els noms de les classes en minúscules.

  Per tal d'evitar malentesos en la sintaxi i alhora agilitzar l'escriptura.

- Els espais entre paraules són substituïts per guions.

  Una paraula després d'un espai seria interpretada per l'HTML com una classe CSS diferent, pel que és necessari substituir els espais entre les paraules d'una mateixa classe amb algun caràcter permès. Els guions són els escollits en la majoria de casos amb contexts similars, i el caràcter '-' és més fàcil de teclejar que el caràcter '_'.

- Els noms de les classes CSS representen un resultat, no una acció.

  Per tal d'acotar el rang de possibilitats al descriure el que es vol, s'eviten les formes en que es sol·liciti una acció. A l'hora de revisar les classes CSS aplicades a un element HTML, s'entén més ràpidament un resultat que una acció, ja que, en el cas d'una acció, mentalment processaríem quin és el resultat de la mateixa per tal d'entendre què esta fent la classe CSS. Al utilitzar com a nom de la classe CSS el resultat del que s'obté en comptes

de l'acció que ho porta a terme, estalviem un pas al nostre cervell a l'hora de revisar el codi i per tant ens resulta més comprensible.

Per posar un exemple de la diferència entre definir com una acció o com un resultat, posem per cas una classe CSS utilitzada per alinear un text a la dreta. Definida com una acció seria "*align-text-to-right*", mentre que si es defineix com un resultat obtindríem "*right-aligned-text*".

- Sempre es mirarà de dir a la classe CSS de la mateixa manera que escriuries en llenguatge real (en anglès) el que vols aconseguir amb ella.

  Volem que el llenguatge natural sigui com el llenguatge real per tal de que la sintaxi ja sigui coneguda per tothom. Una ajuda a l'hora d'entendre quina seria la forma correcta de la classe CSS seria fent-se la pregunta "*I want a / an .....*". Aquí diversos exemples:

I want a / an ...	*centered-text* *underlined-text* *white-text* *red-background* *orange-background-on-hover* *fluid-container* *panel* *bordered-table* *grid* *warning-button*

- Sempre que la classe CSS es pugui definir d'una manera que només utilitzi noms i adjectius es definirà com a tal, evitant les formes que utilitzin altres paraules com articles i preposicions.

  Aquesta també és una convenció per acotar el rang de possibilitats i alhora per afegir consistència al anomenar les classes. El mateix exemple d'abans serveix per explicar aquest punt: tant "*text-to-right*" com "*right-aligned-text*"

defineixen el resultat d'alinear un text a la dreta, però "*text-to-right*" utilitza una preposició, mentre que "*right-aligned-text*" només utilitza noms i adjectius, per tant la segona opció és la correcta.

Seguint aquestes pautes i convencions hauria de resultar intuïtiu saber el nom d'una classe CSS que faci el que es desitja.

4.2.2 Normalize.css

Normalize.css és un projecte de codi obert creat per en Nicolas Gallagher i en Jonathan Neal. Es tracta d'un petit fitxer CSS que sobreescriu alguns estils CSS per defecte dels diferents navegadors, fent que renderitzin els elements més consistentment entre ells i més en línia amb els estàndards actuals. Únicament s'enfoca en els estils que necessiten ser normalitzats.[20]

És habitual sobreescriure alguns estils per defecte del navegador quan es comença un fitxer CSS, Normalize.css s'encarrega d'això però ho fa d'una manera més òptima i consistent, fent que sigui ideal utilitzar-lo com a *reset* [21] en la base d'un framework CSS.

4.2.3 Font Awesome

Vivim a l'era dels píxels. Com a dissenyadors i desenvolupadors web, els píxels poden ser tant els nostres amics com els nostres enemics. Volem que totes les imatges es vegin bé i ben definides per a qualsevol persona que utilitzi les nostres pàgines web, però alhora necessitem reduir el pes dels fitxers per qüestions d'optimització. Actualment la manera més senzilla de d'implementar un sistema amb totes aquestes característiques és mitjançant el format SVG (Scalable Vector Graphics). Els SVG mantenen la mateixa qualitat en totes les resolucions de

[20] https://necolas.github.io/normalize.css/ (Consulta: 28 de maig de 2016).
[21] *Reset*: Reinici, reajustat.

pantalla, tenen un pes de fitxer molt baix, i poden ser fàcilment editats i modificats.[22]

Amb l'aparició dels SVG en el disseny i desenvolupament web van començar a aparèixer les fonts d'icones SVG per substituir la majoria d'imatges que podien ser representades simplement amb una icona. Hi ha força col·leccions d'icones SVG populars, però se'ns dubte la més popular i més completa és Font Awesome, així com l'escollida per el *framework* Platonic CSS.

4.2.4 Paleta de colors

El *flat design*[23], tot i que comença a evolucionar cap al *material design*[24], segueix estant a l'ordre del dia i amb ell s'aconsegueixen dissenys simples i elegants. La decisió de colors per al *framework* Platonic CSS és una paleta de *flat colors*[25] *(Veure* Figura 15*)*.

FIGURA 15. PALETA DE COLORS PLANS

[22] https://svgontheweb.com/ (Consulta: 28 de maig de 2016).
[23] *Flat design*: Disseny pla, poc recarregat i amb suaus contrasts.
[24] *Material design*: Disseny de materials, és un estil de disseny creat per Google pel sistema operatiu d'Android i posteriorment utilitzat en altres entorns.
[25] *Flat colors:* Colors que presenten un dèficit de contrast amb la resta de colors.

Aquesta paleta està recollida a la web https://flatuicolors.com/ i és un recurs popular per a dissenyadors. Ja que la pròpia web serveix com a ràpida referència per saber com és anomenat cada un dels colors de la paleta, les classes CSS que apliquin colors utilitzaran aquests mateixos noms. Per exemple, es pot utilitzar una classe CSS *alizarin-background* per fer que el color de fons d'un element HTML sigui del color vermell d'aquesta paleta. Cal tenir en compte que en el *framework* CSS els únics colors purs que existeixen són el blanc i el negre, la resta són els *flat colors* d'aquesta paleta.

4.2.5 Decisions tipogràfiques

Tractant-se d'un *framework* CSS de codi obert i en constant desenvolupament fins adquirir una consistència equiparable a altres *frameworks* CSS populars, buscava una font amb bona llegibilitat i que oferís una imatge desenfadada alhora que moderna. Una bona elecció era la font *Gotham Rounded* [26], però és una font de pagament i en aquest cas es necessita una font que es pugui distribuir en un projecte de codi obert. La font més similar a la *Gotham Rounded* però de codi obert és la *Varela Round*, que es pot trobar i descarregar a Google Fonts[27].

La mida de la font (propietat *font-size* en CSS) és una decisió molt important a nivell de llegibilitat i accessibilitat. Una mida de 16 píxels em sembla la més adequada, mida suficientment gran per no haver de forçar la vista en la pantalla per llegir el text, i alhora suficientment petita per no semblar una mida exagerada pel text dels paràgrafs en quant a jerarquia visual es refereix.

L'alçada de la línia (propietat *line-height* en CSS) és un altre punt important en quant a llegibilitat i accessibilitat. És recomanable que es trobi entre els 1.5 i 2 punts (mai se sol utilitzar tant com 2 punts), i aquesta quantitat hauria de variar

[26] Es pot veure i obtenir la font a l'enllaç http://www.typography.com/fonts/gotham-rounded/styles/ (Consulta: 28 de maig de 2016).
[27] Enllaç a la font *Varela Round* de Google Fonts: https://www.google.com/fonts/specimen/Varela+Round (Consulta: 28 de maig de 2016).

segons l'ample de la línia de text, a més ample més alçada de línia. La decisió doncs és de 1.5 punts en mòbils i *phablets*[28], 1.6 punts en *tablets*[29], 1.7 punts en ordinadors portàtils i finalment 1.8 punts en pantalles d'escriptori.

4.2.6 Preprocessador i processat de codi CSS

Desenvolupar un *framework* CSS és molt més senzill, escalable i fàcil de mantenir si s'utilitza un preprocessador de CSS, que ofereix una sintaxi menys carregosa i més neta. Tot i que hi ha altres preprocessadors de CSS, els tres més populars són Sass, Less i Stylus, sent llargament Sass el més utilitzat, però la decisió sol ser més personal que no pas per les diferencies que ofereix un respecte als altres.

Per aquest projecte he escollit Less per una senzilla raó: és el més fàcil de compilar en viu en un entorn de producció, ja que el llenguatge ofereix un processador creat en JavaScript que transforma el codi en Less a codi en CSS.

El *framework* CSS està contingut dins l'aplicació modular CMS i per tant dins d'un servidor, el qual no executa JavaScript ja que aquest és executat en banda client, però afortunadament per a això utilitzo la dependència Less.php que permet complir l'objectiu de modificar el CSS del lloc web des de les opcions del CMS.

4.2.7 Jerarquia d'arxius

Quan el framework CSS disposi d'una versió suficientment completa i estable com per publicar-la ja inclourà tots els fitxers en un de sol. Mentrestant durant aquest treball l'ordre en que es carreguen les parts és el següent:

1. Normalize.css
2. Font Awesome
3. Platonic CSS Framework

[28] *Phablet:* Dispositiu mòbil amb una pantalla més gran de 5 polzades però més petit que una tablet.
[29] *Tablet*: Dispositiu portàtil que consta principalment d'una pantalla tàctil com a mecanisme d'interacció.

On el *framework* Platonic CSS està format pels següents components que són importats en el fitxer *platonic.less* en aquest mateix ordre:

1. reset.less
2. typography.less
3. responsive.less
4. containers.less
5. separators.less
6. images.less
7. colors.less
8. grids.less
9. forms.less
10. tables.less
11. headers.less
12. panels.less
13. popup.less
14. buttons.less
15. lists.less
16. sidemenu.less
17. helpers.less

A part d'aquests, hi ha els fitxers *variables.less* i *mixins.less* que proporcionen a cada un dels anteriorment citats les variables del *framework* i *mixins* (conjunts de codi reutilitzables) per facilitar la compatibilitat entre navegadors per als estils que ho requereixin.

4.2.8 Mobile first

El framework està dissenyat pensant en un disseny adaptatiu als diferents dispositius actuals, i ho fa seguint el principi *Mobile First*.

El *Mobile First Design* (Disseny prioritzat per a mòbils) és un concepte que sorgeix després del *Responsive Design* (Disseny adaptatiu) amb la mateixa finalitat d'adaptar el disseny a totes les pantalles, la principal diferència entre aquests dos termes és l'ordre en el que es porta a terme (Veure Figura 16).

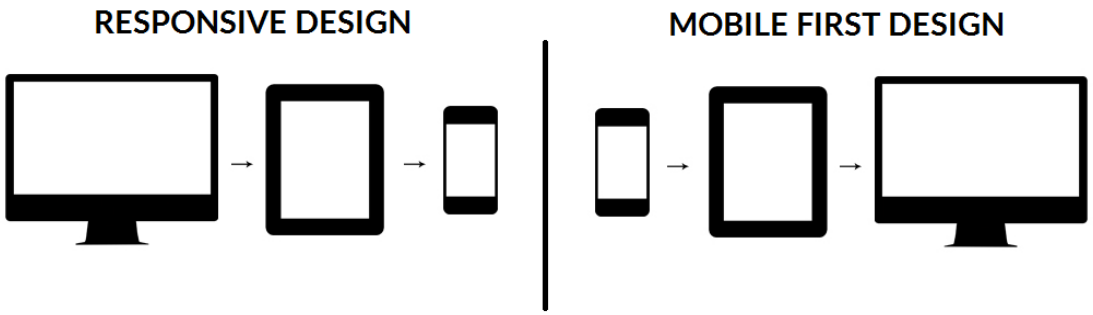

FIGURA 16. DIFERÈNCIES ENTRE RESPONSIVE DESIGN I MOBILE FIRST DESIGN

El *Responsive Design* intenta adaptar tots els continguts d'una web als diferents dispositius, mentre que el *Mobile First Design* crea un disseny amb uns continguts pensats per la pantalla d'un mòbil, i llavors va afegint coses a mesura que és necessari. L'avantatge del *Mobile First* a nivell de codi CSS és que es defineixen els estils pensant en les pantalles més petites i són aplicats a totes les pantalles, i llavors els dispositius amb pantalles més grans apliquen estils extra per aprofitar l'ample de pantalla, això evita que els dispositius mòbils hagin de processar grans quantitats d'estils que no necessiten.

4.2.9 Breakpoints i media queries

Al parlar de disseny adaptatiu en el context del llenguatge CSS, estem parlant de l'ús de *media queries* [30], que alhora impliquen definir uns *breakpoints* [31].

[30] *Media query:* Consulta de medis. Aplicat al context, és una consulta de les mides de la pantalla del dispositiu sobre el que es realitza la consulta.
[31] *Breakpoint*: Punt de trencament.

A Internet es poden trobar molts debats sobre quins haurien de ser els *breakpoints* a utilitzar per a cada tipus de dispositiu, i la conclusió que se'n pot extreure és que no s'haurien de definir els *breakpoints* en funció de les mides de pantalla dels dispositius més coneguts, sinó que s'han de definir en funció de com es volen mostrar els continguts d'una pàgina web.

```
// Breakpoints
@size-for-phablet: 32rem;   // 512 píxels
@size-for-tablet: 48rem;    // 768 píxels
@size-for-laptop: 64rem;    // 1024 píxels
@size-for-desktop: 80rem;   // 1280 píxels
```

FIGURA 17. *BREAKPOINTS* DE PLATONIC CSS

Tenint això em compte, he definit els *breakpoints* en unitats *rem* (unitats relatives a la mida de font tipogràfica definida a l'arrel del document, 16px en el cas d'aquest *framework*) tal i com es pot veure a la Figura 17, amb una diferència de 16*rem* entre cada tipus de dispositiu. Aquestes decisions les he pres tenint en compte diferents recomanacions a l'hora d'escollir els *breakpoints* i donant un cop d'ull als utilitzats als *frameworks* CSS més populars. Les mides reals d'aquests *breakpoints* són relatives a la mida de la font del document, per tant l'exemple donat en píxels només s'aplicarà en cas de no modificar la propietat CSS *font-size* a l'arrel del document.

```
// Queries
@mobile: ~"screen and (max-width: @{size-for-phablet} - 1px)";
@phablet: ~"screen and (min-width: @{size-for-phablet})";
@tablet: ~"screen and (min-width: @{size-for-tablet})";
@laptop: ~"screen and (min-width: @{size-for-laptop})";
@desktop: ~"screen and (min-width: @{size-for-desktop})";
```

FIGURA 18. MEDIA QUERIES DE PLATONIC CSS

Utilitzant els *breakpoints* anteriorment definits, genero les *media queries* que es mostren a la Figura 18 i que s'utilitzen al llarg de tot el *framework* CSS.

5 Resultats

Al llarg d'aquest treball s'ha desenvolupat una aplicació utilitzant el *framework* Laravel que, d'acord amb els objectius plantejats inicialment, funciona com a CMS i segueix un disseny totalment modular. Aquesta aplicació s'ha batejat com Platonic CMS i és totalment reutilitzable en diferents llocs web.

El panell d'administració de Platonic CMS ofereix una interfície intuïtiva i moderna, amb un menú lateral que permet navegar entre les diferents seccions del CMS i una barra d'estat superior. Les seccions del menú són dinàmiques i poden aparèixer o desaparèixer en funció dels mòduls activats o desactivats, de manera ràpida i sense manipular la base de dades.

Per defecte es poden gestionar mòduls, usuaris i opcions del CMS, i els mòduls que es dissenyin per al CMS poden fàcilment ampliar les funcionalitats de l'aplicació. Platonic CMS facilita el desenvolupament i implementació de nous mòduls amb algunes directrius i ajudes per minimitzar els fitxers necessaris i mantenir una base de dades neta i fàcil de mantenir.

Els usuaris es poden registrar en l'aplicació, iniciar sessió per accedir al panell d'administració, gestionar els seus perfils i desconnectar-se quan desitgin.

Un *framework* CSS ha estat creat especialment per proveir Platonic CMS d'una interfície gràfica moderna i visualment agradable. Aquest *framework* s'anomena Platonic CSS Framework (també simplement Platonic CSS) i pot ser utilitzat en el desenvolupament de qualsevol projecte web amb un sol fitxer CSS, sense cap component de JavaScript.

Platonic CSS segueix el patró de disseny adaptatiu *Mobile First* i és totalment compatible amb les versions més recents dels diversos navegadors web.

El desenvolupament amb aquest *framework* ofereix una sintaxi basada en el llenguatge natural que, seguint unes convencions, facilita el seu aprenentatge.

```
<div class="fluid-grid centered-items">
    <div class="half-width panel">
        <div class="panel-header">Títol</div>
        <div class="panel-content">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
            Recusandae, eligendi fugiat necessitatibus illo deleniti,
            dignissimos voluptates nam incidunt est impedit animi eum
            vel consectetur non nulla provident sunt, architecto
            deserunt.</p>
        </div>
    </div>
</div>
```

FIGURA 19. EXEMPLE DE CODI HTML UTILITZANT EL FRAMEWORK PLATONIC CSS

A la Figura 19 es crearia un contenidor d'ample complet que funciona com a grid i centra els elements que conté. Dins aquesta graella hi ha un panell que n'ocupa la meitat de l'ample. Aquest panell té una capçalera amb un títol i un contingut.

6 Discussió

El projecte actualment disposa d'un panell d'administració pel CMS amb diverses opcions i un sistema d'usuaris funcional i la capacitat de desenvolupar i afegir nous mòduls. El *framework* CSS ofereix els estils bàsics necessaris i alguns components dels més utilitzats habitualment.

Tot i que el projecte ja podria ser distribuït, segueix massa subjecte a canvis importants en la seva estructura i presenta algunes mancances que han de ser solucionades.

Actualment el sistema d'usuaris resultant manca de rols d'usuari. Un usuari normal només hauria de poder accedir a unes característiques determinades del panell d'administració, mentre que un usuari administrador hauria de tenir accés a totes les opcions del panell. És un punt pendent de tractar que es pot solucionar modificant la base de dades per afegir una nova taula de rols i una columna referent a un rol a la taula d'usuaris, o bé una taula de rols i una taula pivot en cas de que un usuari hagi de tenir diferents rols.

Al activar un mòdul s'hauria de fer les migracions necessàries a la base de dades. Al desactivar un mòdul, s'hauria de donar la opció de revertir les migracions fetes a la base de dades o bé conservar-les. La dependència PHP que s'encarrega dels mòduls actualment està desenvolupant uns mètodes que es criden al activar i desactivar un mòdul, que serviran per implementar els canvis en la base de dades al realitzar una de les accions citades.

El CSS personalitzable del lloc web hauria de ser editable manualment a través d'un formulari, com a funcionalitat a part de les opcions configurables.

En el desenvolupament d'aquest treball es volia desenvolupar també un mòdul que proporcionés les funcionalitats d'un bloc a l'aplicació. Aquest mòdul actualment existeix però no està finalitzat i únicament ha estat utilitzat per

comprovar algunes de les funcionalitats del mòdul base, pel que no s'ha inclòs el seu desenvolupament en aquesta memòria.

En quant al llenguatge natural de Platonic CSS, tot el plantejament realitzat, tot i que pugui estar ben fonamentat, no deixa de ser teòric. La intenció és que sigui un llenguatge totalment intuïtiu, però a la pràctica existeix la possibilitat de que els desenvolupadors no siguin capaços de percebre-ho com a un llenguatge intuïtiu. En tal cas l'objectiu buscat amb el llenguatge natural no s'hauria assolit.

7 Conclusions

Aquest treball m'ha presentat una visió més amplia i alhora detallada de la presa de decisions en el disseny i desenvolupament d'un projecte web de certa envergadura, especialment quan aquestes decisions afectaran directament als col·laboradors i als usuaris finals. En aquests casos cal cercar i analitzar totes les possibilitats, i seguidament prendre la decisió més adequada amb visió de futur.

El sector de les noves tecnologies evoluciona ràpidament i les necessitats d'adaptar-se a nous patrons de disseny de *software* es fan més evidents. Les solucions complexes als problemes del disseny i desenvolupament web utilitzades un parell d'anys enrere, esdevenen obsoletes a favor de solucions més simples, assolides gracies a l'evolució dels llenguatges web. Afortunadament he pogut aprendre molt sobre aquestes últimes tendències durant la realització d'aquest Treball de Fi de Grau.

Com s'ha pogut veure a l'apartat de discussió, els resultats obtinguts tenen alguns punts febles, però aquests punts febles tenen per contrapartida uns punts forts com són la modularitat i el fàcil manteniment i ampliació del projecte, que en línia de futur permetran solucionar els punts febles progressivament.

Encara que aquest treball acabi aquí, el projecte continuarà en desenvolupament durant un temps indefinit. Se solucionarà els punts febles actuals i es desenvoluparan noves funcionalitats fins aconseguir el que es pugui considerar una primera versió estable. Quan es disposi d'una versió estable el projecte serà hostatjat a GitHub públicament com a projecte de codi obert per al seu ús lliure. Es seguirà treballant en ell periòdicament i publicant-ne noves versions basades en el feedback rebut o en noves funcionalitats que s'hi puguin afegir. Tampoc cal oblidar que Platonic CMS requereix de mòduls que ampliïn les seves funcionalitats, pel que també es crearan nous mòduls com el de bloc.

8 Bibliografia i webgrafia

Laravel Docs

Documentació oficial del framework Laravel.

https://laravel.com/docs/master

Laracasts

Plataforma web d'aprenentatge per a desenvolupadors orientada principalment a Laravel i al llenguatge PHP. Mitjançant lliçons en format screencast en Jeffrey Way explica a la perfecció gran part de les funcionalitats de Laravel. Alhora la plataforma també disposa d'un fòrum especialitzat per a resoldre qualsevol dubte.

https://laracasts.com

Wiki de caffeinated/modules

Documentació del paquet caffeinated/modules per Laravel a GitHub.

https://github.com/caffeinated/modules/wiki

Learning Laravel

Llibres en format web en els que s'ensenya a crear aplicacions amb Laravel a mode de tutorial pas per pas, així com la integració de certes funcionalitats.

http://learninglaravel.net/

Laravel IO

Fòrum per a la comunitat de Laravel on resoldre dubtes relacionats amb el framework.

http://laravel.io/

Stack Overflow

Coneguda plataforma on els desenvolupadors resolen dubtes entre ells. Una de les principals fonts d'informació quan es té algun problema o dubte de programació.

http://stackoverflow.com/

Mozilla Developer Network

La plataforma neix amb la intenció de compartir tot el coneixement de la web oberta. S'hi poden trobar referencies als principals llenguatges de programació web i es tracta d'una alternativa que s'està imposant per sobre del W3Schools.

https://developer.mozilla.org