**Dawsharing grup 4**

**Tutorial per nous programadors**

**Arnau, Jonathan, Martí i Josep.**

1. Eines de desenvolupament
2. Base de dades
3. Backend
4. Frontend
5. **Eines de desenvolupament**

Eines de desenvolupament del projecte utilitzades:

* Netbeans
* SourceTree
* Github
* Plugins per NetBeans

El projecte DawSharing es basa en crear un frontend i un backend d’una pàgina de compartir vehicles per a grups escolars, gent etc.

Per a fer això d’una manera més organitzada, necessitem el **Laravel**.

El Laravel és un framework de codi obert per a desenvolupar aplicacions i projectes web amb PHP.

Creant un compte al **github** i tenint permisos de lectura del nostre projecte, podràs descarregar directament tot el contingut d’aquest amb l’organització del laravel inclosa, amb el programa **SourceTree**. Aquest projecte el podràs obrir des de el **NetBeans** o el IDE que qualsevol esculli.

Seria bó, tenir la carpeta del NetBeans, com a repositori personal del GitHub, de tal manera que tots els pulls/push que es facin, no tinguis que canviar de carpetes i s’actualitzi tot al mateix moment.

El enllaç del nostre projecte és el següent:

<https://github.com/grup4salesians/DawSharing_SleepingOwl>

Aquest enllaç el pots utilitzar per a descarregar el projecte i començar a treballar sobre ell.

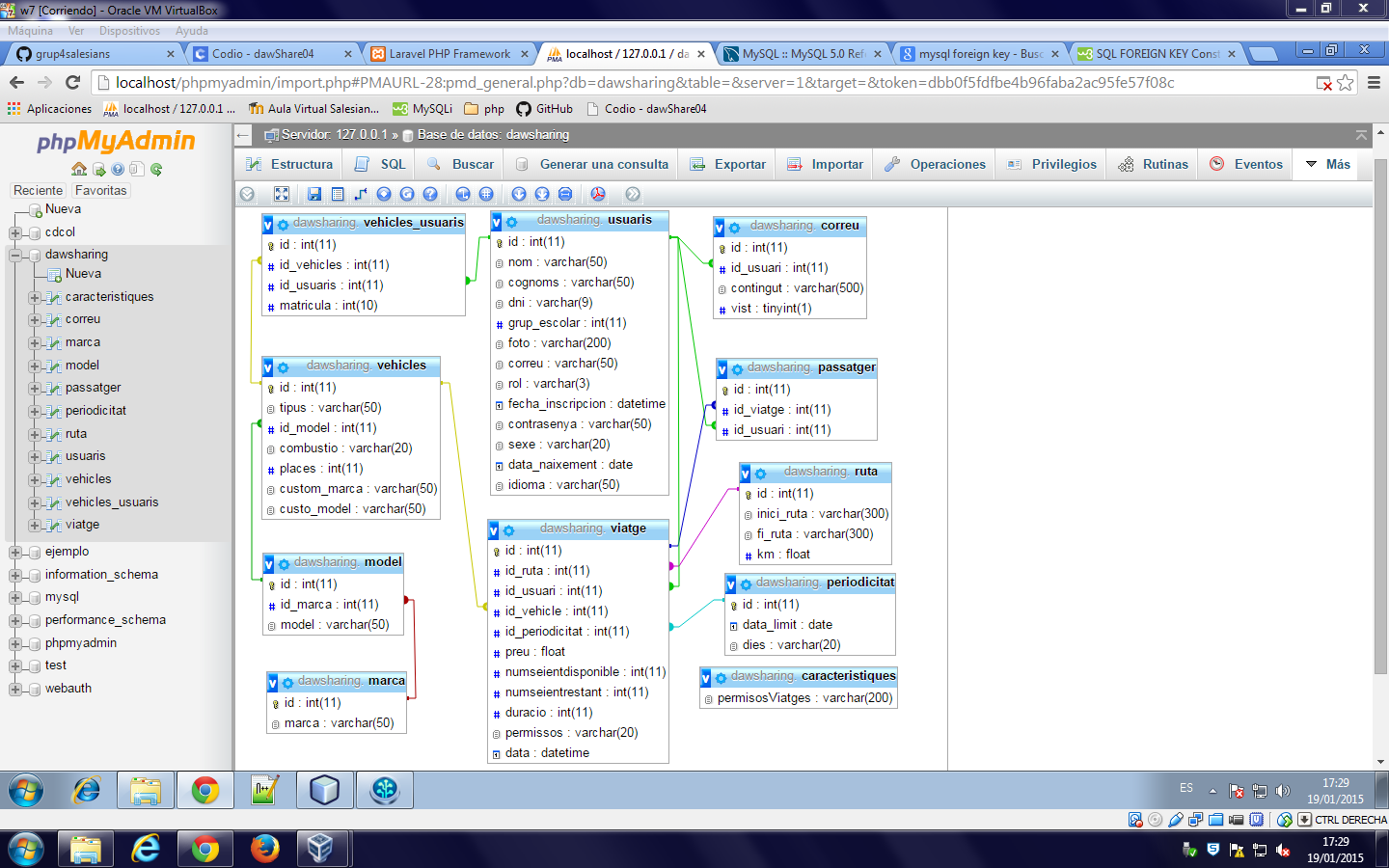
Primer de tot cal saber com funciona el Laravel per a poder ajudar en aquest servei de vehicles.

El primer que hem de tenir en compte es que el Laravel té una organització d’arxius concreta i no es pot modificar a “libre albedrío”, és a dir, les pàgines html van en un lloc(views), les fulles d’estils van en un altre, les direccions de la pàgina enruten en un arxiu en concret (routes.php) etc, dit això, cal veure com hem estructurat la nostre base de dades.

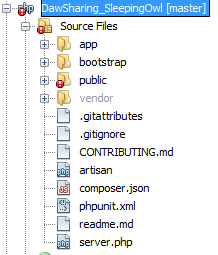
En la secció del frontend extrendrem el contingut del Laravel.

1. **Base de dades**

La base de dades utilitzada per el DawSharing es la següent.



I aquesta l’estructura del Laravel vista des de el NetBeans:



1. **Backend**

Arribat fins aquí, el primer que fem es començar a instal·lar el backend. Hem utilitzar una eina que incorpora taules, administració, etc, aquesta es diu Sleeping Owl, per al Laravel.

Creiem que aquesta era la millor eina que s’adaptava a les nostres necessitats ja que complia amb els requisits necessaris per poder utilitzar-lo amb orientació a objectes i amb el Laravel com a framework.

Aquí tenim informació de com l’hem instal·lat pas a pas:

<https://github.com/sleeping-owl/admin>

I aquí tenim la documentació d’aquesta, per afegir columnes, editar formularis, contrasenyes, amb programació orientada a objectes:

<http://sleeping-owl.github.io/>

Aquí argumentem un mini resum del sleeping-owl:

Tot el backend està situat a APP/ADMIN, tot el que hi ha dins, son les pagines editables de cada contingut del backend, no es gens complicat ja que hi ha poques.

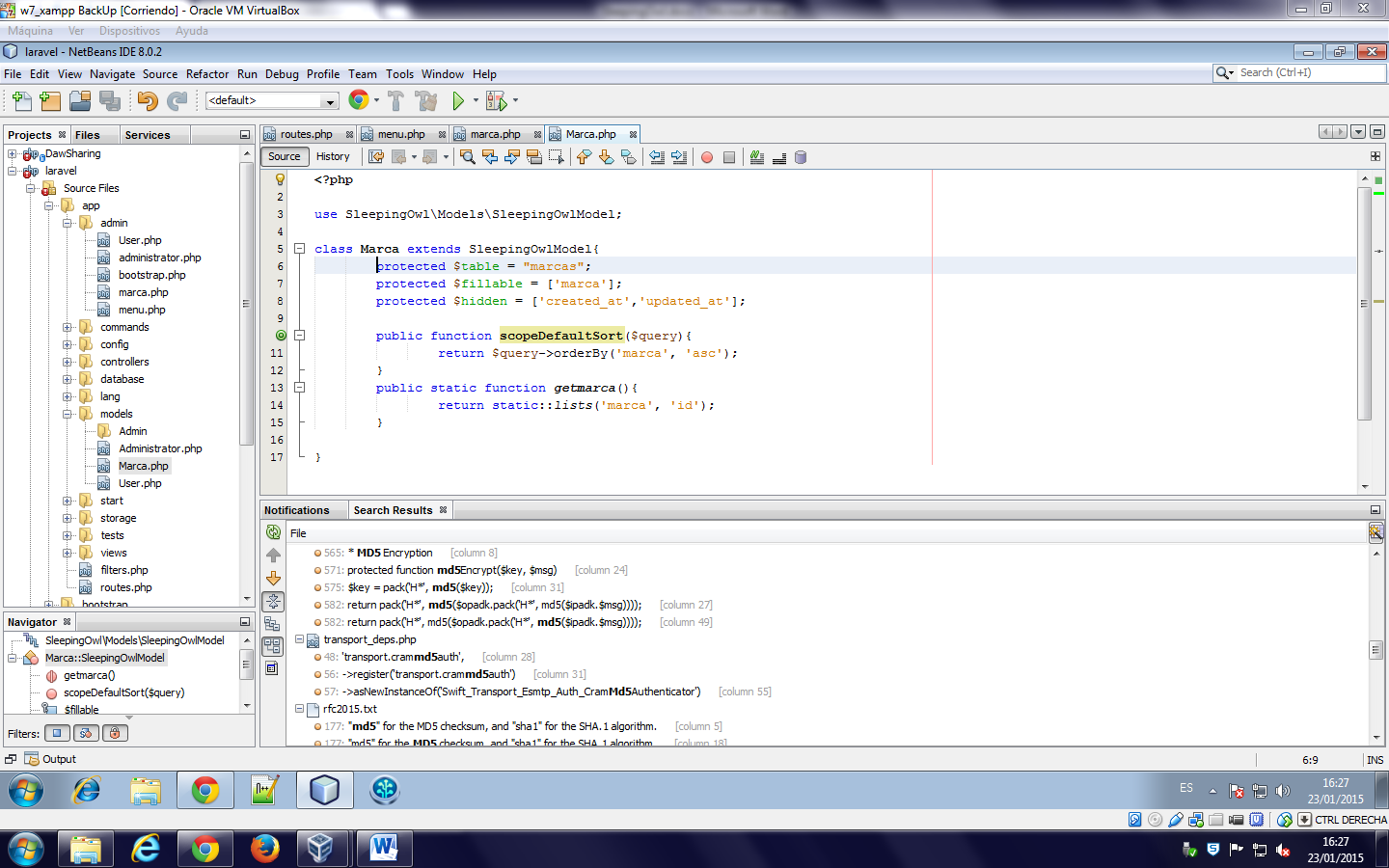
*App/admin/menu.php 🡨 Serveix per configurar la pantalla principal del backend*

*Vendor/sleeping-owl/admin/src/views 🡨 Carpeta on posar les nostres pagines html*

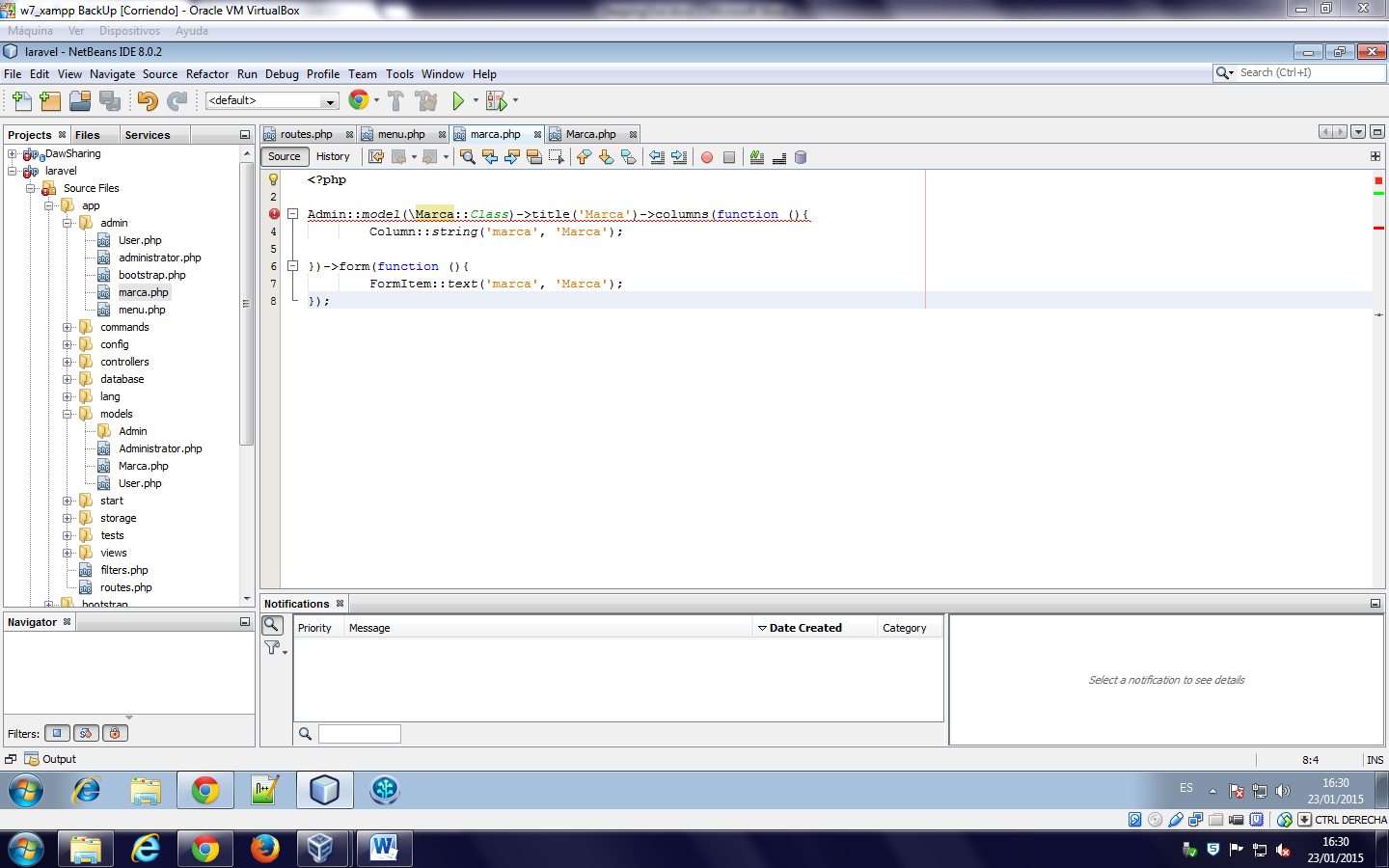
*Vendor/sleeping-owl/admin/src/SleepingOwl/Admin/Controllers 🡨DummyController.php 🡨funcions utilitzades en les nostres págines.*

*Per posar una taula en el nostre backend (CRUD):*

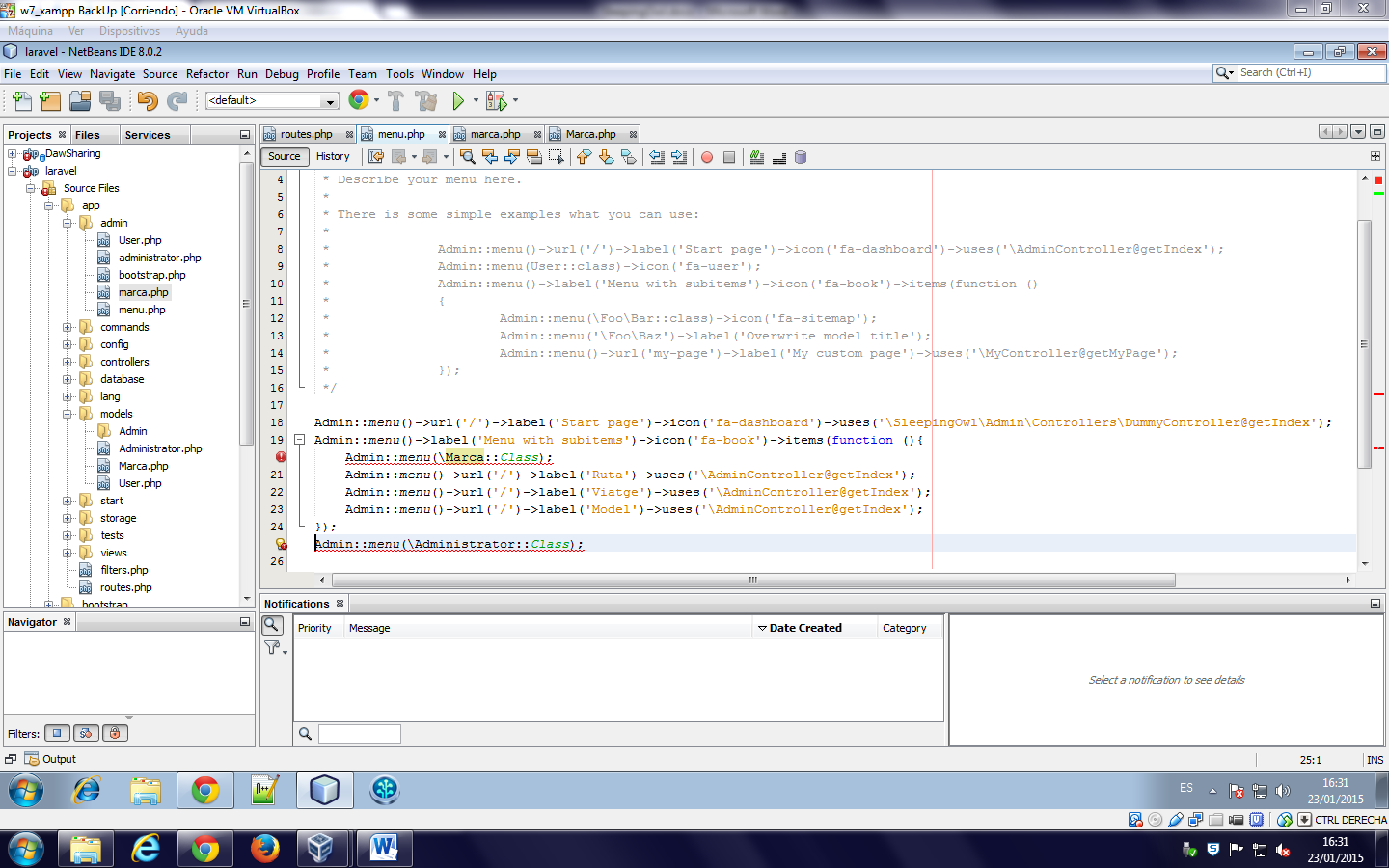
*El primer pas serà crear un model a la ruta app/models. El nom de la classe i d’arxiu han de ser igual que el nom de la taula i en singular. En canvi el nom a la base de dades ha de ser en plural.*

**

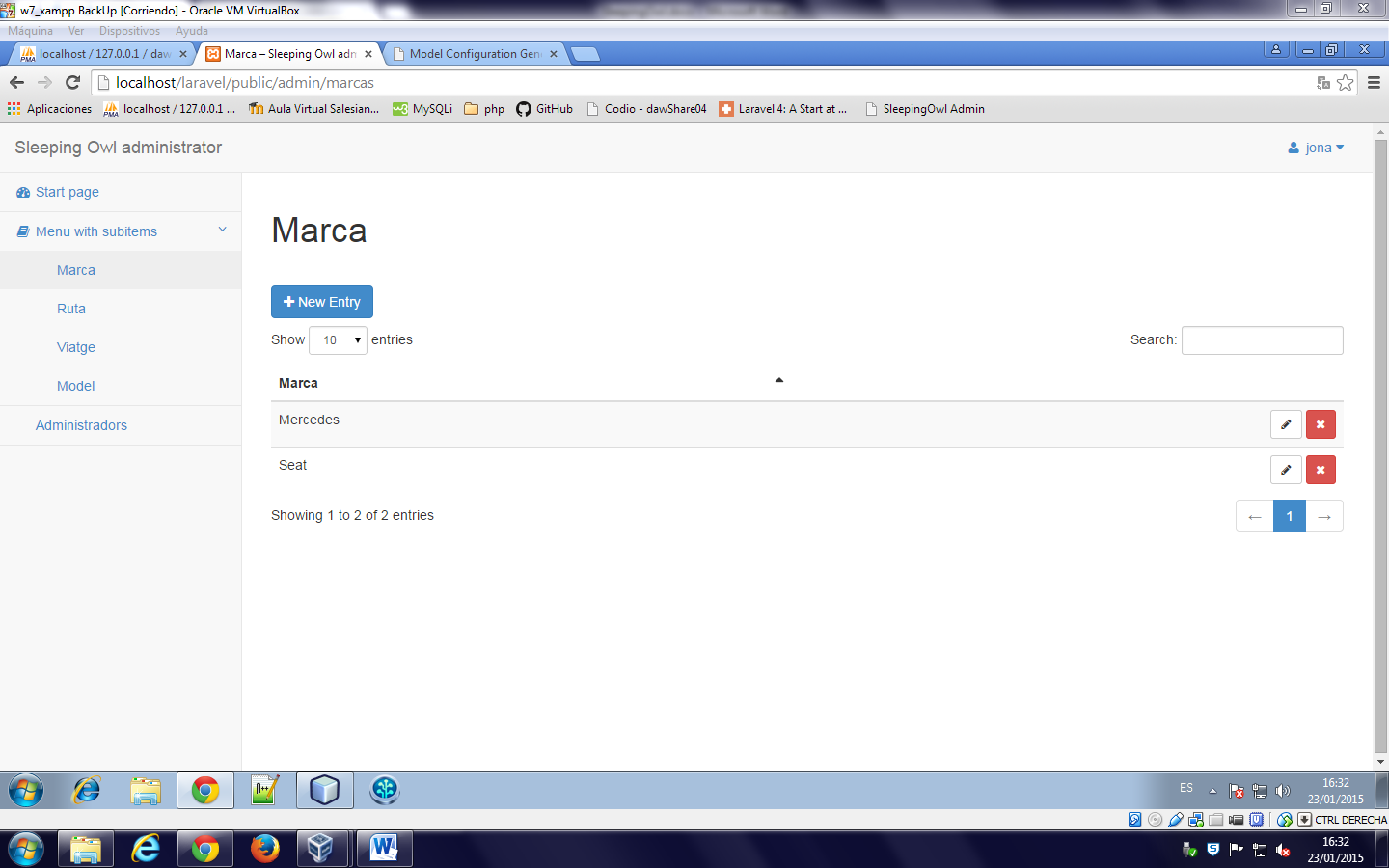
*Després, una vegada fet el model, creem un arxiu (admin/) per a posar les columnes a mostrar de la taula desitjada.*

**

*I per últim, posem el ítem al menú (App/admin/menu.php), per a veure la taula.*



Aquest és el resultat final:



Una vegada fet això, el propi backend ens administrarà la taula en concret, en aquesta podem crear, editar i eliminar amb aquests passos recent esmentats.

També tenim arxius dins del projecte on esmenten tots els tipus de característiques que pots afegir a les taules, formularis..., com per exemple, que una data no mostri les hores, o una ciutat i país es mostri correcte, mostrar una taula amb fills(foranies) etc.

1. **FrontEnd**

En aquest apartat, esmentarem per sectors com fer diverses coses que puguin ser d’utilitat per el frontend, com crear pagines, crear formularis, connexions a base de dades, controls de ruta, pasar variables, lo necessari per que qualsevol persona nova en el projecte pugui entendre com funciona i després de llegir-ho sàpiga crear una pagina nova amb qualsevol contingut.

**Primer apartat:**

Com crear una pàgina

Com hem dit anteriorment, el framework laravel segueix unes pautes per a que el seu funcionament sigui optim i correcte en tots els seus aspectes, hem de saber que totes les pàgines noves amb contingut nou les crearem dins de la següent ruta:

App->Views->Pages.

Qualsevol pagina nova allà ha de tenir una extensió .blade.php. Això ens servirà per poder programar en php(laravel) dins d’aquestes pàgines.

Per poder crear la pàgina cal tenir fixat un “layout” per defecte, és a dir, una estructura de la pàgina general a la que incorporar-hi el nou contingut.

Aquest layout ha d’estar situat a views ->layouts.

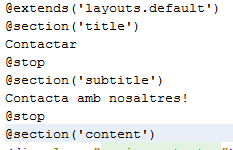
El nostre layout per defecte es el següent:



Com podem observar, el laravel ens permet fer pàgines molt mes optimes gracies a la utilització d’includes.

Aquest layout no s’ha de tocar però s’ha d’incorporar a qualsevol pàgina nova, el que podrem tocar serà el contingut, el apartat de “@yield(‘content’).

Per tant, totes les pagines comensaran per el següent format:



Extenem el layout default, posem la secció del titul amb el titul sota, tanquem el head, hi posem un subtítol, el tanquem, obrim la secció, hi posem tot el contingut que volguem i tanquem amb un @stop.

Automàticament es posara el footer i demes components de la pagina que té per defecte el nostre layout principal.

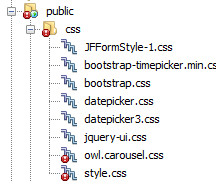
El nostre layout principal és tan optim gracies al haver creat uns includes en el apartat views->includes, com per exemple, el nostre footer-info.blade.php :



On podem veure que es una pàgina normal i corrent amb permisos de programació php(larvel) gracies a la extensió .blade.php.

Aquests includes si els toquem, modificarem totes les pàgines que el continguin, una manera molt optima de treballar i corregir errors.

* Els arxius de css i js que totes les pàgines tenen està situat a públic->css.



Una vegada hem estructurat la nostre nova pàgina cal definirla en una ruta, una ruta que qualsevol persona posi, li indiqui que anirà a aquella pàgina sense necessitat de posar la direcció de les carpetes ni la extensió, per fer això cal anar a **routes.php**, a la carpeta de views.

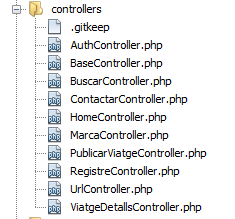
Aquest arxiu ens permetrà controlar totes les rutes de la nostre pagina web:



La imatge indica l’estructura que tenim nosaltres, cal posar el nom de la pàgina que volguem, i enviar-ho a un controlador.

Els controladors serveixen per, tal com diu el nom, controlar que es el que fa aquesta direcció, si mostrar una pàgina directament, si fer una sèrie de funcions, si aquesta pàgina ve despres d’un formulari serà un POST, etc...

Tots els controladors están situats a app->controllers:



Nem a veure un exemple del controlador que tenim al nostre buscarruta:



Aquest controlador és un arxiu amb una classe php, dins, hi posem la funció la qual hem determinat a la ruta, després d’un “@”.

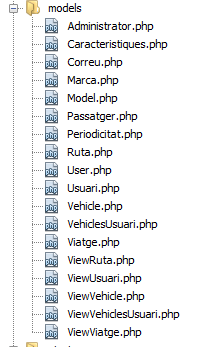
En aquest controlador, agafem una sèrie de variables per get que ens envia la pàgina de buscar ruta, aquestes variables les guardem en una array i ens envia a la pàgina “pages.buscarruta, $variables’.

Si tenim una pàgina creada amb el nom buscarruta on la direcció anterior, ens enviarà a aquesta, gracies al retorn View::make.

Els controladors serveixen per utilitzar la informació que enviem o agafem i treballar sobre aquesta, pasant una sèrie de controls, com per exemple, si no s’omple un formulari, ens retorni a la pàgina anterior amb un error.

**Configurar les nostres taules de la base de dades i la base de dades**

A la ruta App->Models, tenim creats per cada taula i vista, un arxiu associat a aquest:



L’estructura de cadascun d’aquests arxius es la següent:



Al posar “use sleepingOwl...”, ens permetrà que el nostre backend pugui utilitzar la taula en concret.

Cal crear una classe anomenada igual que la taula on indiqui el nom i els camps, després podem crear funcions per poder utilitzar dins del laravel i del nostre backend orientat al laravel.

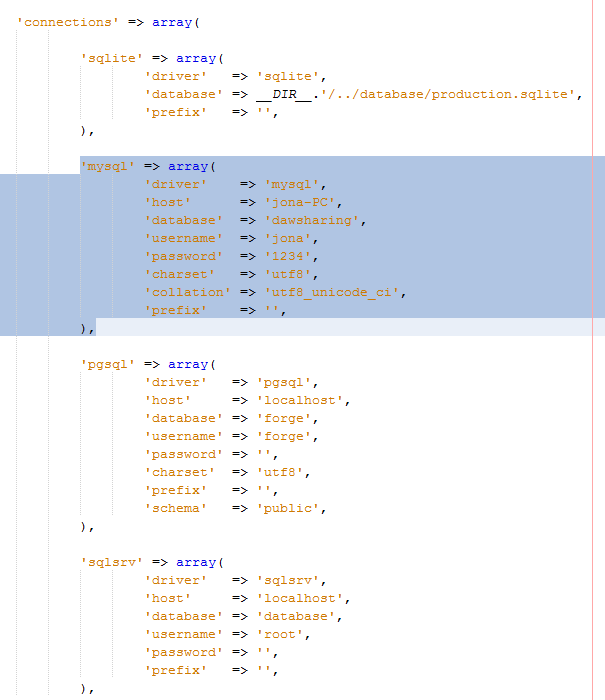
Una vegada tenim creada la classe, a la nostre pàgina amb extensió .blade.php hi podem accedir a aquestes taules amb el següent format:



Nom de la taula, funció, paràmetres i ->get(), el qual ens donarà per json els valors de la taula si es que els té.

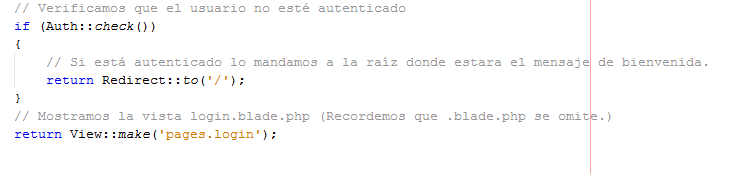
Dit això, per poder fer-ho, cal tenir configurada una base de dades en el framework, aquesta la podem editar en un arxiu anomenat database.php a app->config.

Aquest es el arxiu:

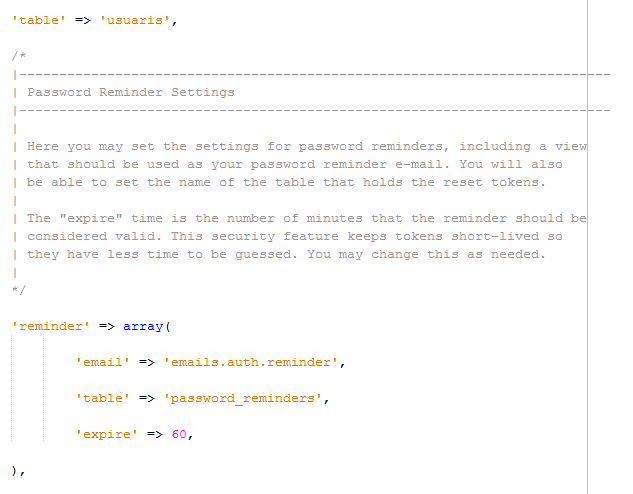


Subratllat amb color, tenim la nostre base de dades principals, on indiquem el nom que volguem utilitzar, direcció, usuari, contrasenya etc.

En aquesta mateixa carpeta hi podem trobar el arxiu auth.php, aquest s’encarregarà d’autentificar els usuaris a la nostre web per després poder fer funcions en els controladors com la següent:



En aquest arxiu només hem de dir quina taula utilitza i quins camps han de concordar, amb el temps que dura aquesta sessió:



En els arxius principals de laravel, tots venen documentats amb ingés per a qualsevol modificació que hi volguem fer, sabem relativament que es el que estem tocant.

**Com enviar correus i com configurar un “template” de correus.**

Gracies al framework de laravel, ens permet enviar e-mails d’una manera molt sencilla. Exposaré un exemple:

Tenim una pàgina amb el nom contactar.blade.php, aquesta conté un formulari de contacte:



Aquest formulari ens enviarà a contactar, una vegada li donem al botó d’enviar.

Després el nostre arxiu de rutes ens envia al nostre controlador d’aquesta pàgina per POST i a la seva funció postFormulari:



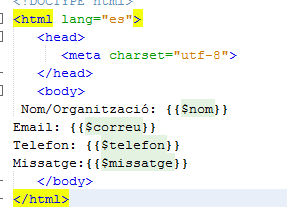
La nostre funció del controlador es la següent:



En aquest controlador hi guardem en un array, les variables del formulari anterior, després de pasar una validació de controls sobre el formulari, utilitzem la funció **Mail::send+paràmetres.**

Aquests paràmetres són els següents:

Página per defecte de l’estructura d’aquest email, en el nostre cas es “emails.contacto”, que conté la següent estructura creada a views->emails->contacto.blade.php



Després de seleccionar la pàgina hi introduïm les variables que volem utilitzar en aquesta, seguida de la funció “message” del laravel.

Després podem modificar paràmetres com a qui li enviem, el títol, subtítol, de part de qui ve, etc.

Nosaltres ho enviem a [dawsharing04@gmail.com](mailto:dawsharing04@gmail.com) amb el subjecte de “Missatge via contactar”.

Després de fer això ens retornarà a la pàgina principal, el nostre e-mail serà enviat amb les variables corresponents i amb la plantilla corresponent.