|  |
| --- |
| DAW\_Sharing |
| Aplicació per compartir recursos |
| **SdS Software Factory; Dep. Informàtica - Salesians Sarrià Arnau , Jonatan, Martí i Josep** |

Índex

[Registre de conductors / passatgers 3](#_Toc413263772)

[Manteniment de cotxes / motos 4](#_Toc413263773)

[Alta de la ruta, com a conductor o com a passatger 4](#_Toc413263774)

[Localització d’una ruta, entrant la població d’origen i la de destí. 4](#_Toc413263775)

[Manteniment de les meves dades 5](#_Toc413263776)

[Compartir un recurs en una ruta 5](#_Toc413263777)

[Càlcul de l’estalvi 5](#_Toc413263778)

[Backend de gestió de l’aplicació 5](#_Toc413263779)

Casos d’ús…………………………………………………………………………………………………………………………………7

Manual del programador ……………………………………………………………………………………………………….10

Presupost del treball ……………………………………………………………………………………………………………..32

DAW\_Sharing

L’empresa SdS Software Factory ha decidit crear una nova divisió de software enfocat al sector d’aplicacions per internet i que anomenarem DAW\_Sharing

Caldrà fer una aplicació que permeti compartir recursos, inicialment cotxes / motos.

En una primera fase cal implementar les següents funcionalitats:

* **Registre de conductors i/o passatgers, hauran de ser alumnes de l’escola.**
* **Manteniment de cotxes / motos associats a conductors.**
* **Alta de la ruta, com a conductor o com a passatger.**
* **Localització d’una ruta, entrant la població d’origen i la de destí.**
* **Manteniment de les meves dades.**
* **Càlcul d’estalvi.**
* **Backend de gestió de l’aplicació.**

Cal que l’aplicació web presenti un entorn professional amb l’aspecte, logos i expressions adaptades al servei.

# Registre de conductors / passatgers

Aquest mòdul permetrà donar d’alta i modificar les dades dels conductors i/o passatgers. Les dades s’hauran de validar mitjançant un servei web que connectarà a la base de dades de l’escola. Un mateixa persona pot ser alhora conductor i passatger ( en dies diferents ).

Les dades mínimes a consignar seran:

* Nom i cognoms
* DNI (integer)
* Grup escolar ( p.e. S2W )
* Foto
* Correu electrònic

Caldrà habilitar un sistema de cerca de clients per DNI, o per grup escolar, o per nom i primer cognom. Aquesta cerca es farà de forma exclusiva per alguna de les 3 condicions expressades anteriorment. Quan ens donem d’alta se’ns enviarà un correu electrònic per validar el procés.

# Manteniment de cotxes / motos

Cal fer el manteniment dels cotxes / motos d’un conductor . Un mateix conductor podrà tenir diversos recursos. Per fer-ho registrarem les dades bàsiques de cada cotxe / moto:

* Marca cotxe / moto
* Model cotxe / moto
* Color cotxe / moto
* Matrícula cotxe / moto
* Tipus cotxe / moto ( gasolina, dièsel, híbrid, elèctric )

# Alta de la ruta, com a conductor o com a passatger

Caldrà gestionar les rutes on s’oferirà compartir els recursos. Mitjançant tres passos ( 1.- On vols anar 2.- Quan hi vols anar 3.- Característiques ) es definirà la ruta i es mostrarà en un mapa.

Les dades bàsiques a consignar en el pas 1 són:

* Si ets conductor o passatger.
* Si ets conductor >> selecciones el recurs
* Població des d’on s’inicia el viatge
* Població on finalitza el viatge

Les dades bàsiques a consignar en el pas 2 són:

* La freqüència del viatge ( data concreta / viatge periòdic ). Si és una data concreta caldrà seleccionar mitjançant un calendari. Si és un viatge periòdic, caldrà poder seleccionar diferents dies de la setmana.
* El tipus de viatge ( d’anada / d’anada i tornada ). Tan si és d’anada com d’anada i tornada caldrà indicar la hora i minuts.

Les dades bàsiques a consignar en el pas 3 són:

* Si el equipatge pot ser gran o no.
* Si es permet fumar o no.
* Si es pot menjar/beure o no.
* El número de places disponibles.
* El preu del servei per plaça.
* Un comentari opcional ( com a molt 255 caràcters ).

# Localització d’una ruta, entrant la població d’origen i la de destí.

Aquesta opció permetrà entrar la població d’origen i la població destí, i cercarà tots els recursos disponibles, ja siguin com a conductors o com a passatgers. Ha de permetre uns filtres fins per tal de poder ajustar la cerca.

# Manteniment de les meves dades

Aquesta opció permetrà veure les meves rutes ( tant com a passatger com a conductor ), els meus vehicles. Podrem fer les modificacions que calgui, inclús esborrar el que convingui, però només quan sigui posterior a la data actual. Es podran consultar les rutes històriques on hauré participat com a passatger o com a conductor.

# Compartir un recurs en una ruta

Una vegada registrat, podrem com a passatger o com a conductor cercar una ruta/acompanyant que ens convingui, i un cop trobada, ens podrem posar en contacte o bé amb el conductor o bé amb l’acompanyant. L’aplicació ens permetrà enviar una sol·licitud a la persona. Si aquesta sol·licitud s’accepta caldrà actualitzar les places disponibles del viatge.

# Càlcul de l’estalvi

Aquesta opció del web permetrà a qualsevol persona ( per tant en obert ) calcular quin estalvi tindrà si comparteix el recurs. Haurem d’entrar la població origen, la destinació i per afinar el càlcul: el tipus de combustible, el preu de combustible (en €/l), el consum (l/100 Qm), el preu dels peatges ( en € ), i altres despeses ( en € ). Un cop entrades aquestes dades, ens visualitzarà el cost en € de fer el viatge sol, o bé acompanyat amb una, dues o tres persones.

# Backend de gestió de l’aplicació

El backend ha de permetre als administradors del servei consultar qualsevol dada de l’aplicació. El sistema ha de ser prou intuïtiu i usable perquè li sigui molt fàcil trobar la informació.

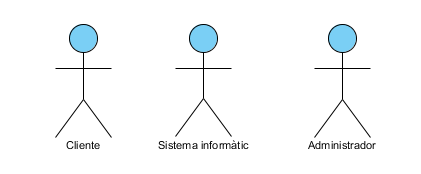
Podrà treballar amb persones ( passatgers, conductors ), recursos ( motos, cotxes ), rutes.

L’aplicació permetrà realitzar diverses estadístiques respecte el servei. Per exemple, quines son les rutes més utilitzades, a quines hores es comparteix més els recursos, quin és el conductor que ha portat més persones.

Detall de casos d’ús DawSharing Grup 4

**Donar d’alta una ruta**

**Actors**



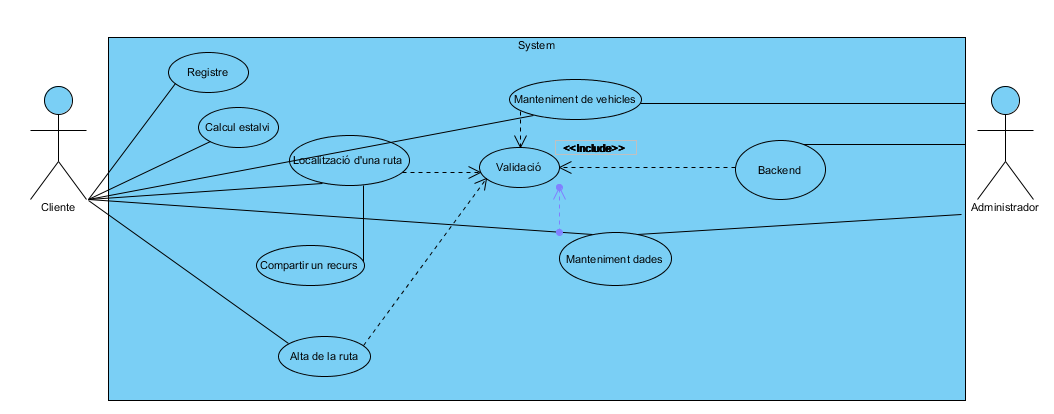
**Client: Usuari que utilitza la pàgina web.**

**Sistema informàtic**: Sistema que gestiona i automatitza les diferents possibilitats que té el usuari dins la pàgina web, entre les quals principals, son buscar i publicar una ruta.

**Administrador**: Persona que s’encarrega d’administrar el sistema informàtic de la web.

**Identificació de casos d’ús:**

* **Registre de conductors i/o passatgers:** Permet donar d’alta un usuari a la web.
* **Manteniment de cotxes / motos associats a conductors:** Permet gestionar els diferents vehicles del usuari.
* **Compartir un recurs en una ruta:** Permet enviar una sol·licitud per entrar en el trajecte d’un usuari.
* **Alta de la ruta, com a conductor o com a passatger:** Permet donar d’alta una ruta ja sigui com a conductor o com a passatger per a fer el trajecte.
* **Localització d’una ruta, entrant la població d’origen i la de destí:** Permet buscar una ruta introduint les dades de destinació i origen.
* **Manteniment de les meves dades:** Permet gestionar les dades personals del usuari.
* **Càlcul d’estalvi:** Permet calcular l’estalvi estimat del trajecte.
* **Backend de gestió de l’aplicació:** Permet gestionar tota la pagina web per al seu manteniment informàtic.



**Detall de casos d’us: Alta d’una ruta**

**Objectiu:**

L’objectiu és que l’usuari doni d’alta una ruta per a buscar companys i compartir el seu cotxe per a guanyar estalviant gasolina en un trajecte d’un origen a un destí seleccionant una sèrie de característiques.

**Precondició:**

L’usuari ha d’estar registrat.

**Flux bàsic**

.

1. L’usuari accedeix a la web.
2. L’usuari es valida al sistema.
3. L’usuari es acceptat pel sistema.
4. L’usuari indica un origen i un destí.
5. L’usuari accedeix a publicar la ruta.
6. L’usuari selecciona un cotxe [A1].
7. L’usuari selecciona unes característiques.[A2]
8. L’usuari publica el viatge.
9. El sistema registra el viatge i el publica a la pàgina web.
10. Altres usuaris poden o no, influir-se en el viatge i el publicador ha d’acceptar la petició.

**Fluxos alternatius**

[A1]: Si el usuari no té un cotxe ha d’afegir-hi un.

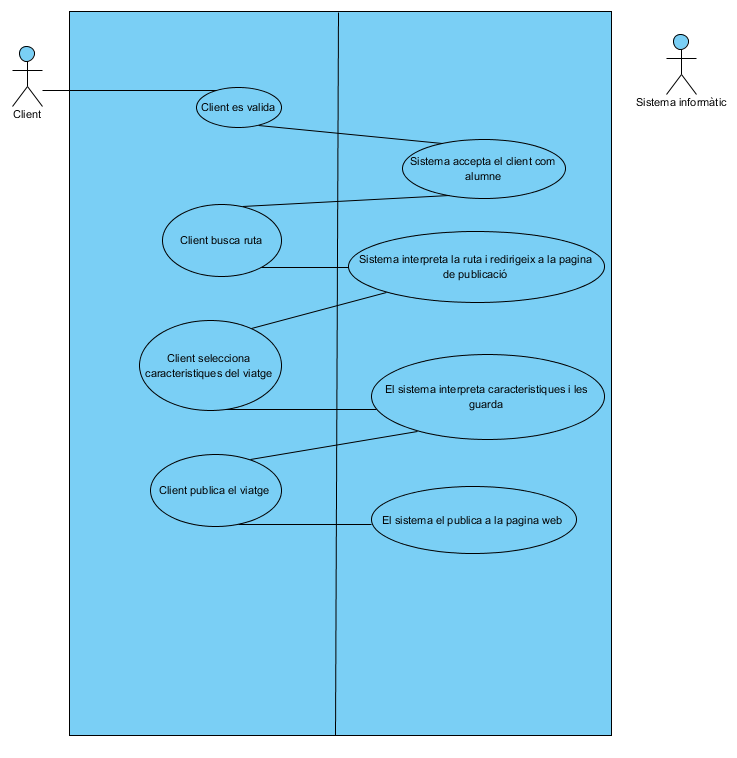
[A2]: L’usuari pot decidir si publicar un viatge com a conductor o passatger, després seleccionar entre característiques com poder fumar, portar maleta etc.

**Postcondició:**

El viatge es valida i s’ha d’esperar fins al dia i hora seleccionada per a que el sistema l’interpreti com ha acabat, igualment quedarà registrat a la base de dades però ja no sortirà disponible a la pàgina web.

**Requeriment especial:**

Cal tenir un cotxe per poder publicar viatge.



**Tutorial per nous programadors**

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.

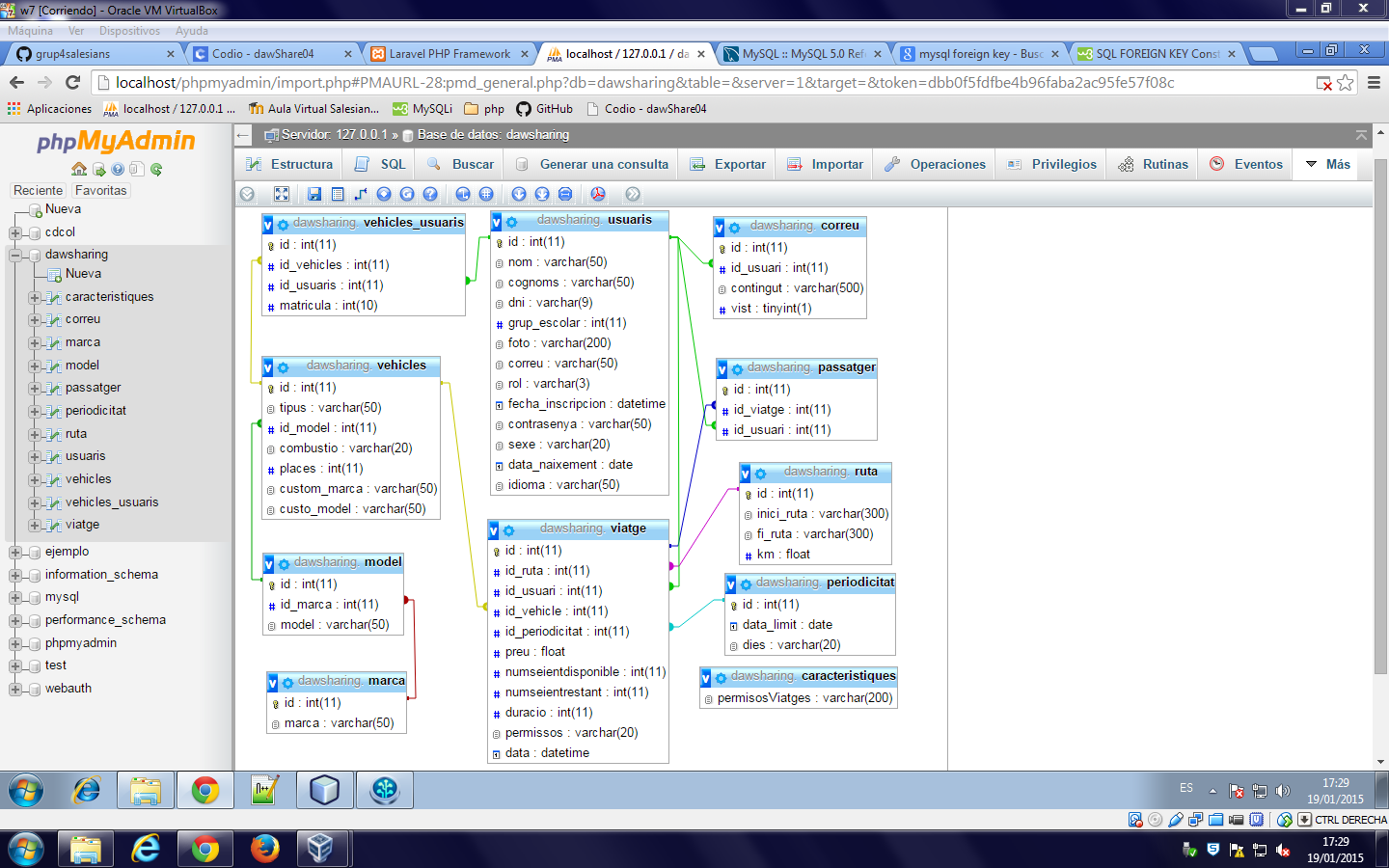
Primer de tot cal saber com funciona el Laravel per a poder ajudar en aquest servei de vehicles.

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.

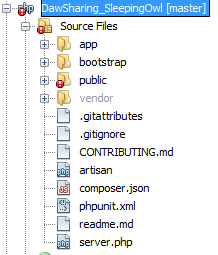
En la secció del frontend extrendrem el contingut del Laravel.

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.

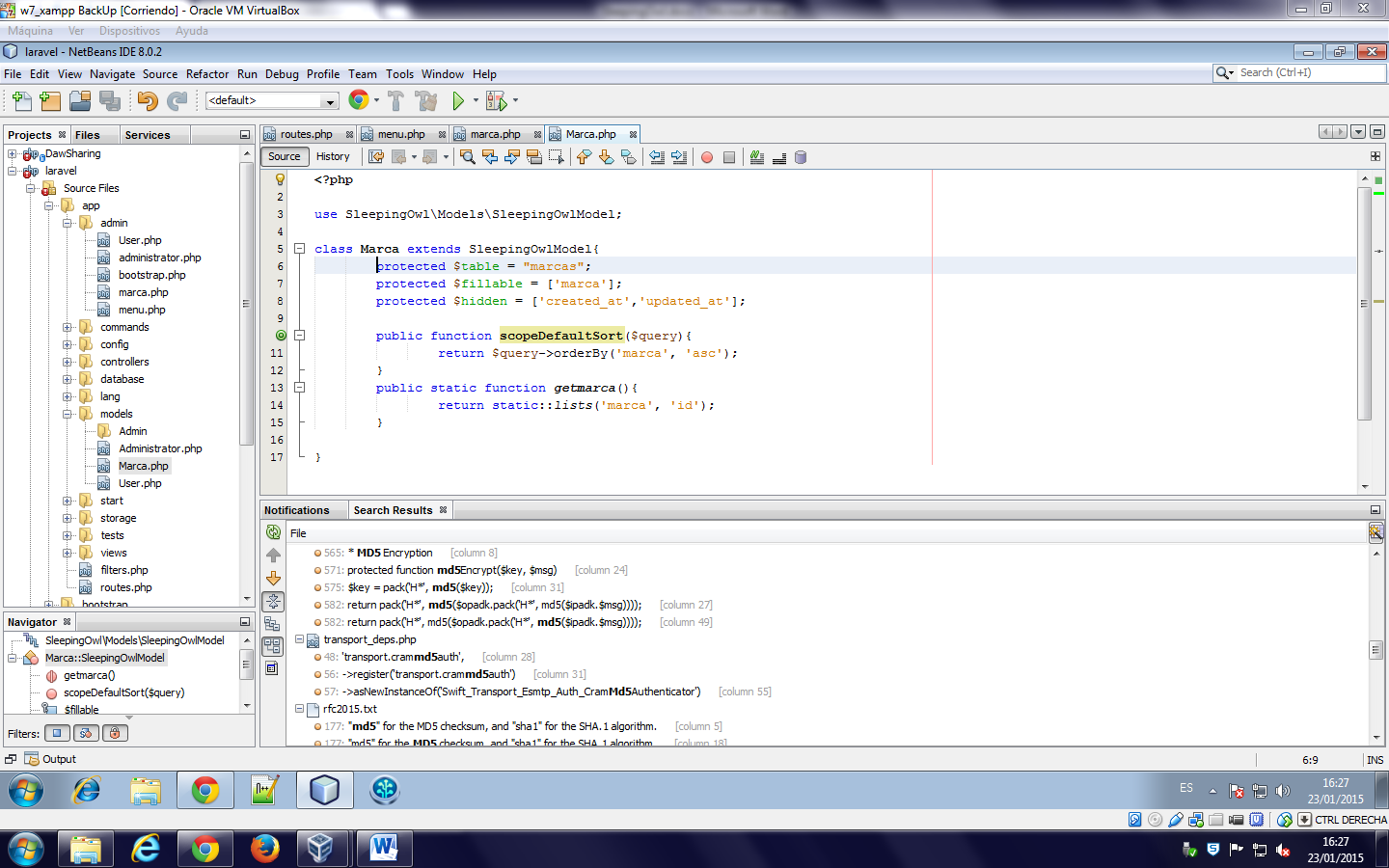
*App/admin/menu.php 🡨 Serveix per configurar la pantalla principal del backend*

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

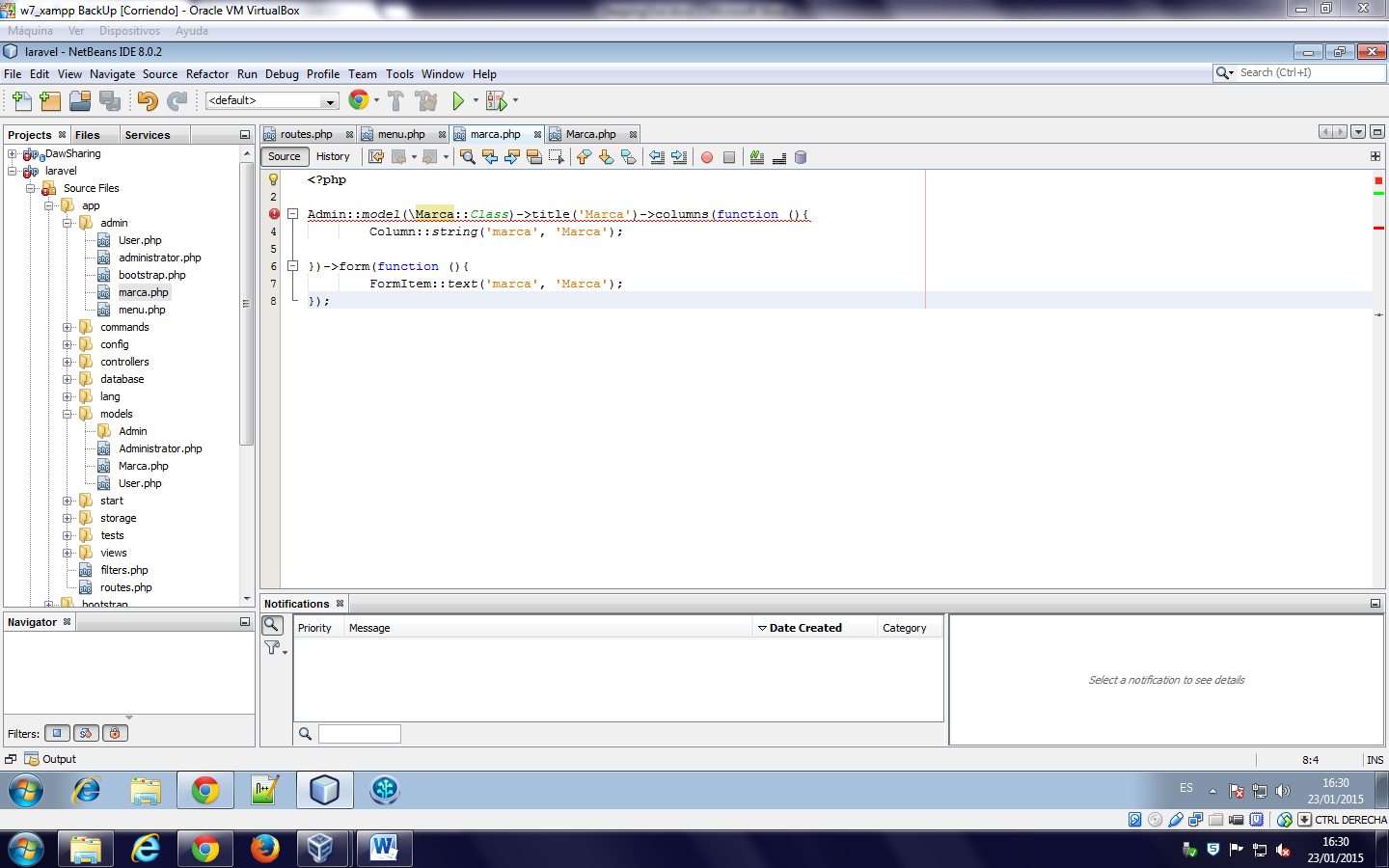
*Vendor/sleeping-owl/admin/src/SleepingOwl/Admin/Controllers 🡨DummyController.php 🡨funcions utilitzades en les nostres págines.*

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

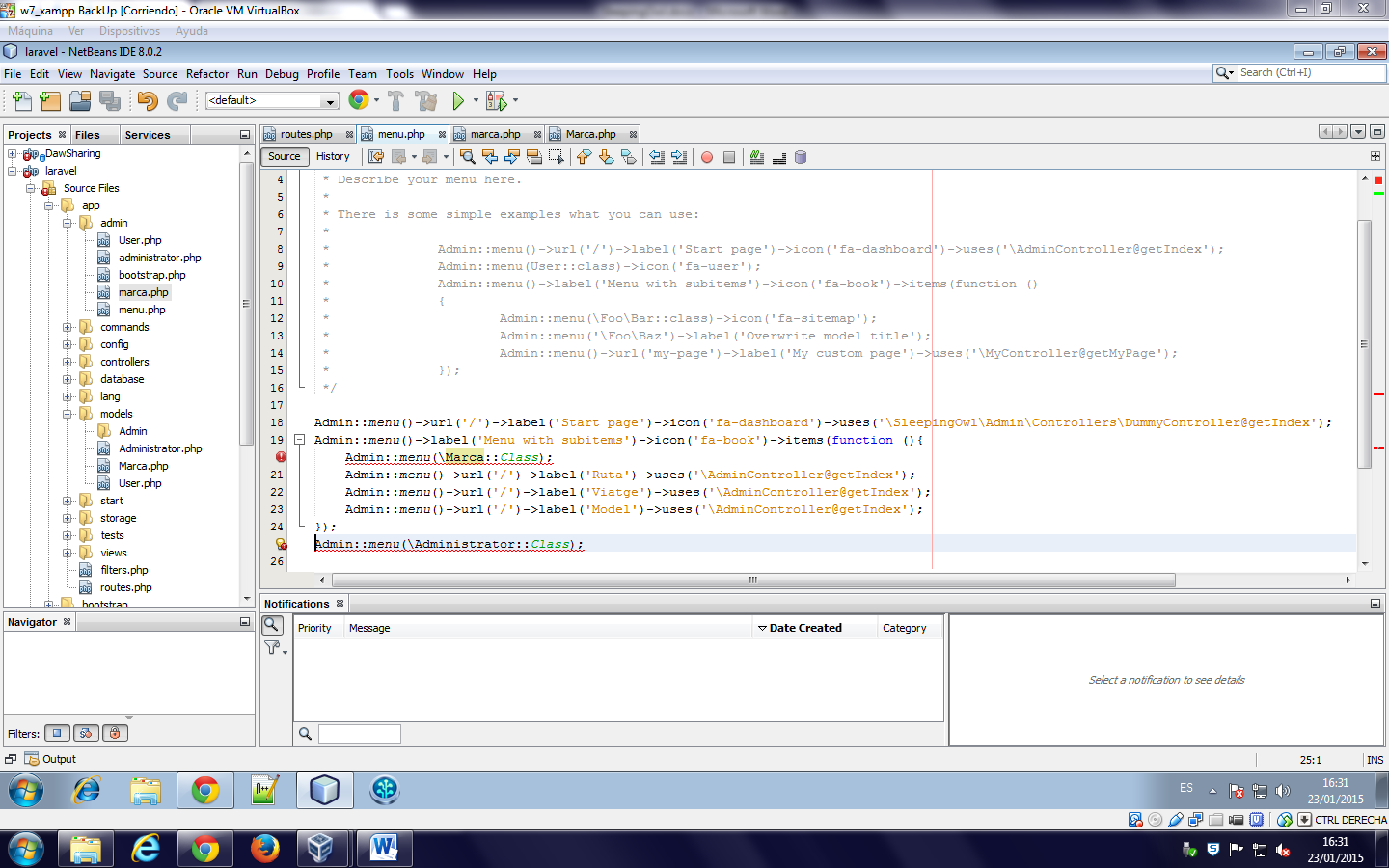
*El primer pas serà crear un model a la ruta app/models. El nom de la classe i d’arxiu han de ser igual que el nom de la taula i en singular. En canvi el nom a la base de dades ha de ser en plural.*

**

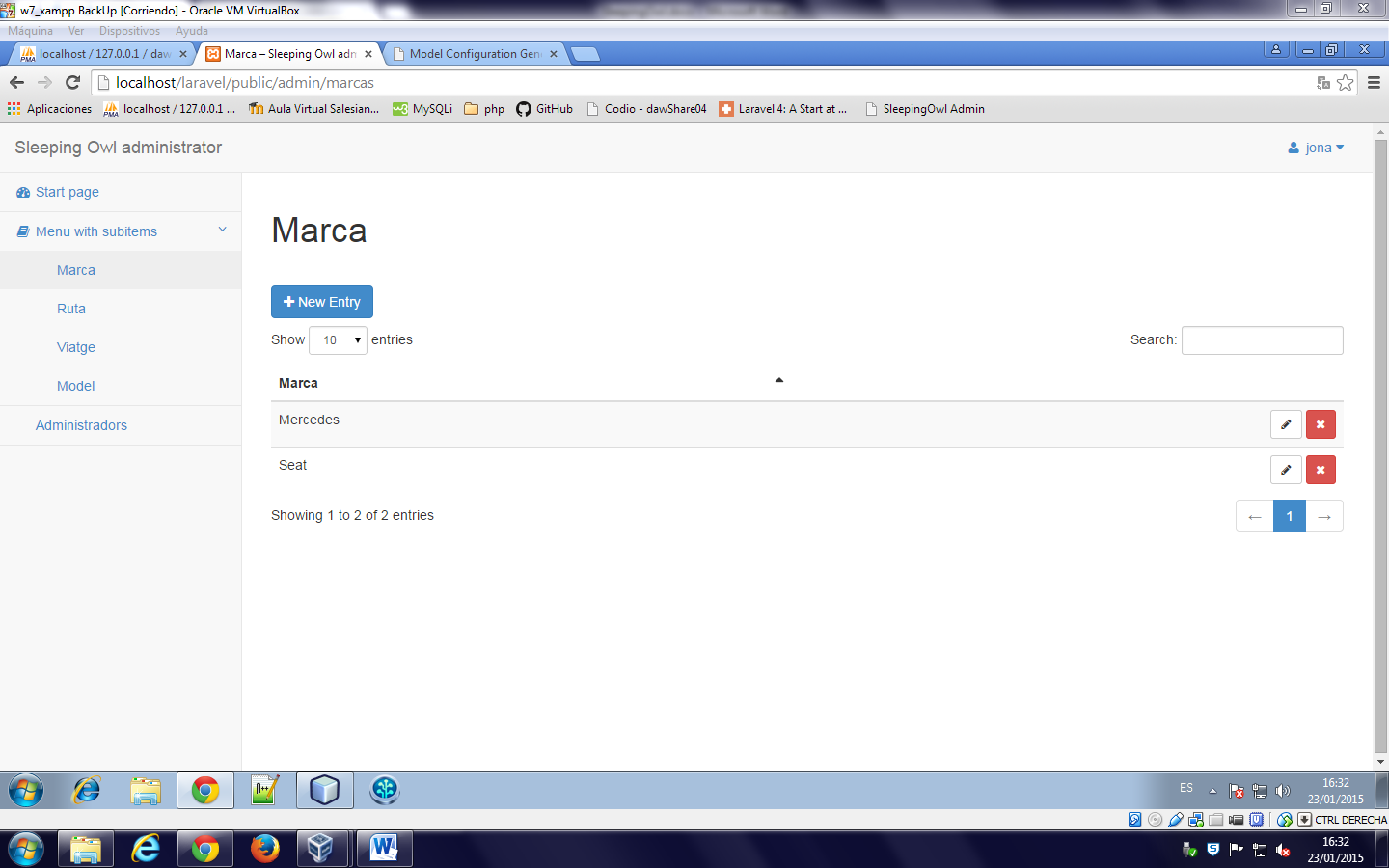
*Després, una vegada fet el model, creem un arxiu (admin/) per a posar les columnes a mostrar de la taula desitjada.*

**

*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.

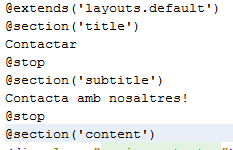
El nostre layout per defecte es el següent:



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.

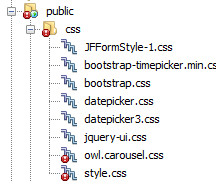
El nostre layout principal és tan optim gracies al haver creat uns includes en el apartat views->includes, com per exemple, el nostre footer-info.blade.php :



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.

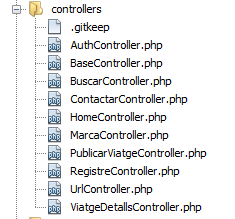
Aquest arxiu ens permetrà controlar totes les rutes de la nostre pagina web:



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 “@”.

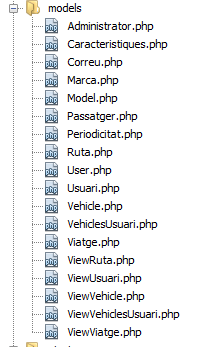
En aquest controlador, agafem una sèrie de variables per get que ens envia la pàgina de buscar ruta, aquestes variables les guardem en una array i ens envia a la pàgina “pages.buscarruta, $variables’.

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.

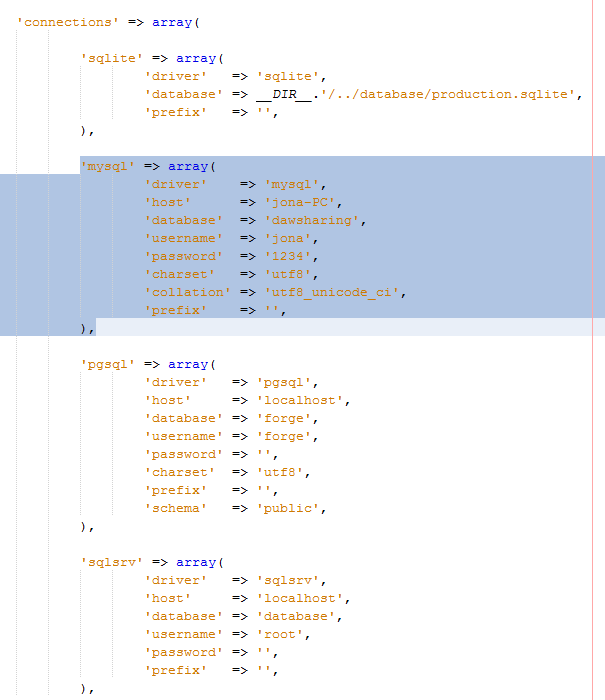
Una vegada tenim creada la classe, a la nostre pàgina amb extensió .blade.php hi podem accedir a aquestes taules amb el següent format:



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

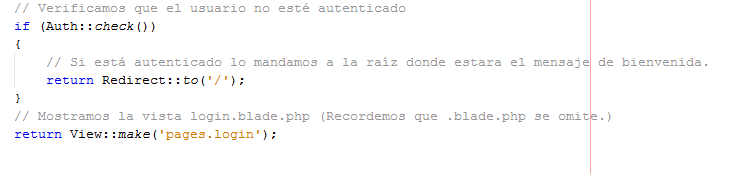
Dit això, per poder fer-ho, cal tenir configurada una base de dades en el framework, aquesta la podem editar en un arxiu anomenat database.php a app->config.

Aquest es el arxiu:

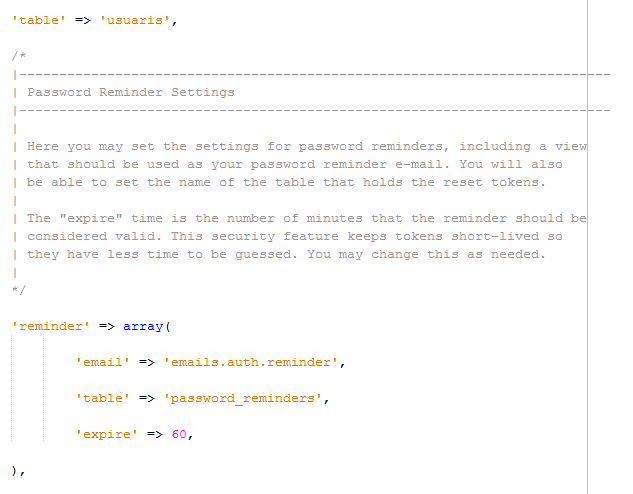


Subratllat amb color, tenim la nostre base de dades principals, on indiquem el nom que volguem utilitzar, direcció, usuari, contrasenya etc.

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:



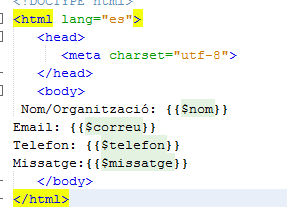
La nostre funció del controlador es la següent:



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.

## 

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Pressupost |  |  |  |
| **DawSharing grup4 S.L.** |  |  |  |
| ***Software Development*** |  |  |  |
| Avenida Diagonal 567 |  |  |  |
| Barcelona, 08029 |  |  |  |
| Teléfono (93) 2025689 Fax (93) 2025689 |  |  |  |
| B67890215 |  |  |  |
|  |  |  |  |
| **Nº DE FACTURA:** | 1462/07-F |  |  |
|  |  |  |  |
| Fecha: | 13/10/2009 |  |  |
|  |  |  |  |
|  |  |  |  |
| **Cliente** |  |  |  |
|  |  |  |  |
| Att.: |  | Dawsharing grup 4 S.L |  |
| Dirección: |  | C/Artesans, 60 |  |
| Localidad: |  | Valls, 43800 |  |
| Telf.: |  | 0034 977 609061 |  |
| N.I.F./C.I.F. |  | B689615001 |  |
|  |  |  |  |
|  |  |  |  |
| **Nº Trabajo:** | 1462/07-F |  |  |
|  |  |  |  |
| **Concepto** |  |  |  |
| **Descripcion del proyecto o servicio** |  |  |  |
|  |  |  |  |
| **DESCRIPCIÓN** | **TOTAL HORAS** | **COSTE HORA** | **TOTAL IMPORTE** |
| **Programación del software** | 140 | 25,00 € | 3500,00 € |
| **Dominio** |  |  | 7,95 € |
| **Hosting** |  |  | 79,00 € |
| **Manteniment mensual** |  |  | 10 € |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  | SUBTOTAL | **3596,95 €** |
|  |  | TIPO IMPOSITIVO | **21,0%** |
|  |  | IMPUESTO SOBRE VENTAS | **755,359 €** |
|  |  | **TOTAL** | **4352,3095 €** |