La Alvaroteca website Documentation.

La Alvaroteca website is restaurant management system, allowing customers to place bookings and check menu, daily menus and tasting menus. This proyect was developed by Alfonso Planas Guerrero as his Higher Technician in Development of Web Applications final proyect, and even if its a real proyect from the restaurant **La Alvaroteca**, no one of the data showed here are sensitive and all the code is open source.

You can check this proyect fully working (as a user) at laalvaroteca.com.

Features

La Alvaroteca achive the following features:

Users

+Check weekly set menus. +Check season menu. +Get information about degustation menus. +Get informacion about the restaurant. +Request bookings. +Cancel bookings.

Employees

Set employees accounts. Set menu and menu images. Set daily menu. Manage bookings. Get costumer booking historial. Get bookings stadistics.

Technology used

All the code and libraries used in this project are open-source and it wouldn be posible to develop this project soo easy without then. Here es the list o all the tecnology used:

Backend

This the backend of this proyect has been developed with MVC architecture in **PHP**, **SQL** and **HTACCESS** languages with the following technology:

Apache

Probally the most famous open source server. I use it for serve the proyejct in a VPS server.
Web site: httpd.apache.org.

Mysql Server

The biggest relational database managment is used for store our data.
Website: www.mysql.com/

Phalcon

The fastes PHP framework! Phalcon is a high-performance PHP web framework based on the model–view–controller (MVC) pattern. Unlike most PHP frameworks, Phalcon is implemented as a web server extension written in C, aiming to boost execution speed, reduce resource usage, and handle more HTTP requests per second than comparable frameworks written primarily in PHP. One drawback of this approach is that root/administrative access is required on the server to install Phalcon by building a custom binary or using a precompiled one.
Website: phalconphp.com.

Frontend

All the front end is development in Xhtml, javascript and sccs langauge with teh following technology:

foundation

Foundation is a responsive front-end framework and I used it for al the UI layel. Foundation provides a responsive grid and HTML and CSS UI components, templates, and code snippets, including typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.
Website. foundation.zurb.com.

backbone.js

Backbone.js is a JavaScript library with a RESTful JSON interface and is based on the model–view–presenter application design paradigm, so the data is picked up from the server and is treated as MVP in the client too. Backbone.js gives structure to the web application, by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.
Website: backbonejs.org

underscore.js

Backbone's only hard dependency is Underscore.js for RESTful persistence. Underscore.js is a JavaScript library which provides utility functions for common programming tasks and a strong template sytem.
Website: underscorejs.org/.

jquery.js

Backbone dependend on jquery for DOM manipulation, and Who wouldn't use jquery for facilitate his javascript pogramming?
Website: jquery.com.

Slick.js

Since foundation js components doesn support carousel images anymore, I used this light librery for show some images in the website.
Website: kenwheeler.github.io/slick/.

Fontawesome

I use fontawesome for all the icons on the website. I didnt use foundation font becouse is allmost the same size and I need some icons what they didn't have, like tripadvisor icon.
Website: http://fontawesome.io/.

flatdoc

This Documentation is powered by flatdoc at laalvaroteca.com/documentation. Flatdoc is a small JavaScript file that fetches Markdown files and renders them as full pages.
Website: ricostacruz.com/flatdoc/.

Proyect development

La alvaroteca is development in a model view controller patern. When we open the project we can find the folowing structure:

```
laalvaroteca/
    papp/
    common/
    controllers/
    email/
    forms/
    logs/
    models/
    plugins/
    views/
    cache/
        volt/
    config/
    public/
        js/
        css/
        assets/
        assets/
        css/
        fonts/
        js/
```

```
      index.php
   schemas/
.htaccess
.htrouter.php
```

At the root folder we find the proyect fild which we will describe below and the .htaccess / .htrouter.php files which will redirect us to the public folder, where ir the website hosted. With this way all the controllers, model and logic is encapsulated under the data showed to the user, and we can get a better integrity for our website.

Config folder

In this folder we can find all our web configuration and resource. It will be defined the initial configuration, services and routes to load.
Importan files from **config folder**:

Config.ini

This is the configuration fle where is stored all information about the project. The database information.

```
[database]
adapter  = Mysql
host     = localhost
username = user
password = ***
dbname   = laalvaroteca
```

The and the application information where you can find al the information about controllers routers and the base uri from the proyect.

```
[application]
controllersDir = app/controllers/
modelsDir      = app/models/
viewsDir       = app/views/
pluginsDir     = app/plugins/
formsDir       = app/forms/
commonDir     = app/common/
emailDir     = app/email/
baseUri        = /laalvaroteca/
```

loader.php

In this file a set of directories are registered taken from the configuration file by the class *\Phalcon\Loader*.

```
$loader = new \Phalcon\Loader();
```

```
$loader->registerDirs(
    array(
        APP_PATH . $config->application->controllersDir,
        ...
        ...
    )
)->register();
```

services.php

This is probaly the most important configuration file, everithing used by application is configured here:

We use *Phalcon\DI\FactoryDefault* class. The FactoryDefault Dependency Injector automatically register the right services providing a full stack framework.

```
$di = new FactoryDefault();
```

With *Phalcon\Events\Manager* class We register the events manager, we send all the events produced to the security plugin and we handle exception and not found exception using NotFoundPluggin

```
$di->set('dispatcher', function () use ($di) {
    $eventsManager = new EventsManager;
    $eventsManager->attach('dispatch:beforeExecuteRoute', new SecurityPlugin);
    $eventsManager->attach('dispatch:beforeException', new NotFoundPlugin);
    $dispatcher = new Dispatcher;
    $dispatcher->setEventsManager($eventsManager);
    return $dispatcher;
});
```

With *Phalcon\Mvc\Url* class we set the base uri from the project (define in config.ini). This component is used to generate all kind of urls in the application.

```
$di->set('url', function () use ($config) {
    $url = new UrlProvider();
    $url->setBaseUri($config->application->baseUri);
    return $url;
});
```

With *Phalcon\Mvc\View;* class we registered the application views, using volt engine.

```
$di->set('view', function () use ($config) {
    $view = new View();
    $view->setViewsDir(APP_PATH . $config->application->viewsDir);
    $view->registerEngines(array(
        ".volt" => 'volt'
```

```
    ));
    return $view;
});
```

Then, we set up Volt.

```
$di->set('volt', function ($view, $di) {
    $volt = new VoltEngine($view, $di);
    $volt->setOptions(array(
        "compiledPath" => APP_PATH . "cache/volt/",
    ));
    $compiler = $volt->getCompiler();
    $compiler->addFunction('is_a', 'is_a');
    return $volt;
}, true);
```

We establish the database.

```
$di->set('db', function () use ($config) {
    $config = $config->get('database')->toArray();
    $dbClass = 'Phalcon\Db\Adapter\Pdo\\' . $config['adapter'];
    unset($config['adapter']);
    return new $dbClass($config);
});
```

With *Phalcon\Session\Adapter\Files* class we start the session used for some componenet (this is not a php server session but a cookie session managed by phalcon.

```
$di->set('session', function () {
    $session = new SessionAdapter();
    $session->start();
    return $session;
});
```

An with Phalcon\Flash\Session we registered the flash flash service with custom CSS classes

```
$di->set('flash', function () {
    return new FlashSession(array(
        'error'   => 'alert alert-danger',
        'success' => 'alert alert-success',
        'notice'  => 'alert alert-info',
        'warning' => 'alert alert-warning'
    ));
});
```

Public folder

Here is where we find al the data to showed to the user.At the root of this folder we find the *index.php* file of the website and a .htaccess file which rewrite the url showed. Videos, images and resources are stored at the **assets folder**, stylsheets at the **css folder** , javascript files at **js folder**.

Important files and folders from **public folder**:

index.php

This is the root of the website and where the "magic" happens and we implement all the website configuration.
First, we read and charge the configuration from config.ini .

```
 $config = new ConfigIni(APP_PATH . 'config/config.ini');
    if (is_readable(APP_PATH . 'config/config.ini.dev')) {
        $override = new ConfigIni(APP_PATH . 'config/config.ini.dev');
        $config->merge($override);
    }
```

We include the Autoloader configuration.

```
require APP_PATH . 'config/loader.php';
```

We load application services

```
require APP_PATH . 'config/services.php';
```

And we start the web application.

```
$application = new Application($di);

echo $application->handle()->getContent();
```

app folder

Here is where is stored all the logic from the application, we can find:

common

Its a folder we all the common contollers/libraries for use in the application, here we found:

-Uuid.php: It's a small libary which provides unique keys for store models in the BD.

controllers

Here we find each controller for every part of the website. This controllers show, create, update,

delete data and redirect when its necesary. Inside this folder we can also find the folder **Ajax** were are stored all the ajax controllers used for get data for js and show in the front-end and also for modify, delete and create some data and request some actions.

emails

Here are stored all the emails controllers used as response when a user make or cancel a reservation.

logs

Here is stored all the errors in diferents errors files. This folder is used mainly for debugging and development.

model

All the model logic is stored in the **model folder**. The restriction and relations are controller in its files by *Phalcon\Mvc\Model\Relation* class, intead of being declared in the database schema.

plugins

Here are stored the plugings used by the event manager. We can find:

SecurityPlugging: With this plugging we check if there is an user logger before execute a path and we restinge some path (by causing a 404 exception) if there is not logged user and we try to ge them. This is how we protect the controll panel controllers from unwanted users.

NotFoundPluging: When an exception happens, this pluggings identify what kind of exception was, and it redirects you to a 404 or 500 page error.

views

All the volts file are stored here. There is 2 templates on the root (one for the website and anotherone for the control panel) and a volt file for each view showed in the website,

Cache

Here is where the cache data is stored, we can find:

volt

Volt is a template system that consume some recurses, so when a template is recuested a cache view is stored in the **volt folder** when the view is compiled and the next time we request the same view we will serve with the cache if the template hasn't changed, so we wont have to compile the template again.

schemas

The database schemas is stored in this folder.