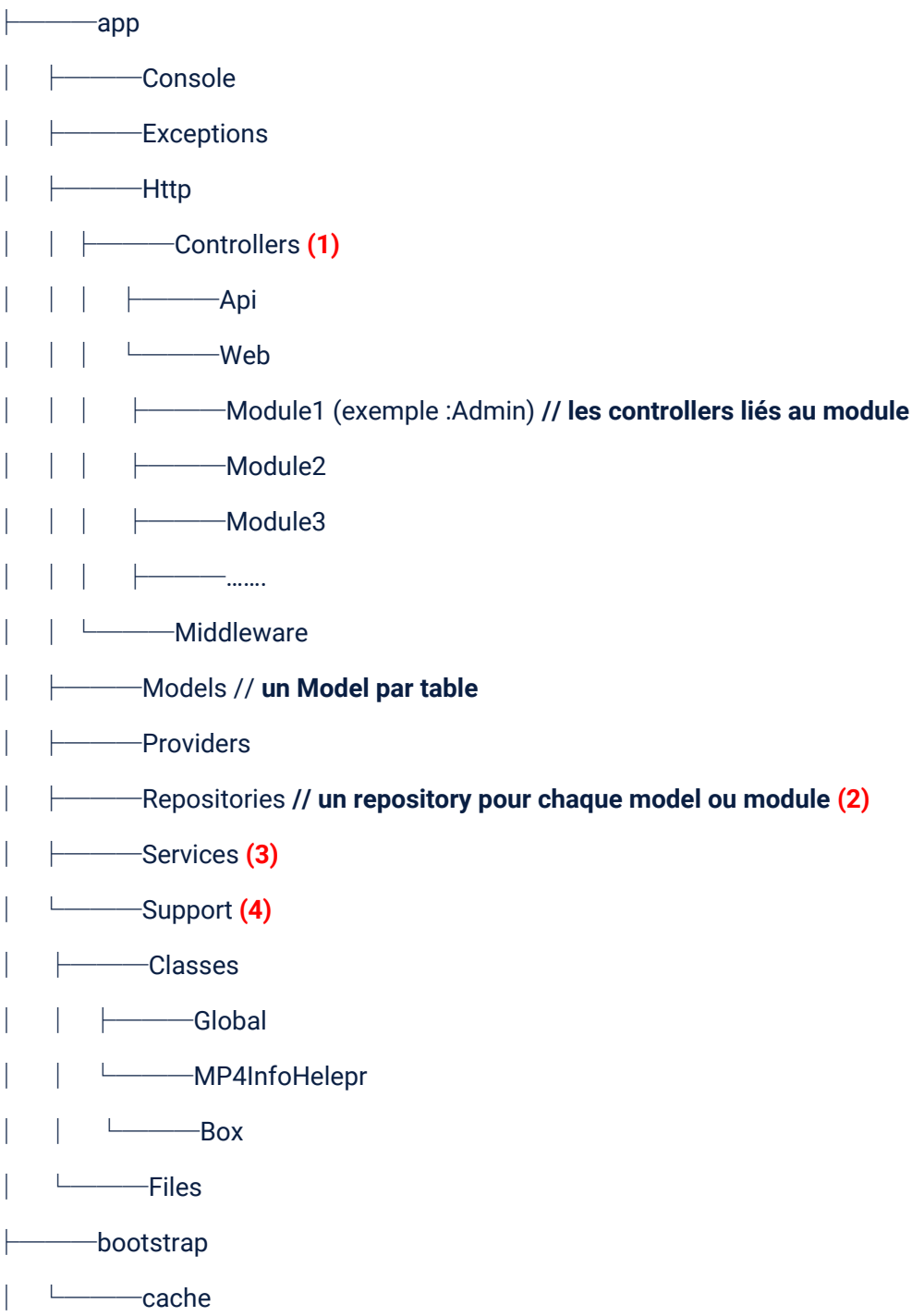
Architecture d’un projet Laravel



```
├──────config
│    ├──────constants // ce fichier doit contenir toutes les constantes utilisées dans le projet.
├──────database
│    ├──────factories
│    ├──────migrations
│    └──────seeds
├──────public
│    ├──────css  // fichiers css générés uniquement , ex : projectName-styles.css
│    └──────js // fichiers js générés uniquement,  ex: projectName-scripts.min.js
├──────resources
│    ├──────assets
│    │    ├──────css // les fichiers css créées
│    │    ├──────js // un fichier js par module  (5)
│    │    │    ├──────Module1
│    │    │    ├──────Module2
│    │    │    ├──────Module3
│    │    │    ├──────.....
│    │    ├──────sass //Contient la liste des fichiers scss liés à la template
│    │    └──────vendor //Contient la liste des librairies utilisées par l'app (bootstrap, jquery..)
│    │    ├──────bootstrap
│    │    ├──────bootstrap-sass-official
│    │    ├──────bootstrap-social
│    │    ├──────datatables
│    │    ├──────datatables-plugins
│    │    └──────datepicker
│    ├──────lang
│    │    └──────en
│    │    │    └──────messages // ce fichier doit contenir tous les messages du projet
```

│ │ │ └──validation // **ce fichier doit contenir tous les messages de validation du projet**

│ └──views

├──routes **(6)**

│ ├──api **// un seul fichier pour les routes api**

│ ├──api.php

│ └──web **// fichier par module et y mettre les routes front liés aux models (crud) et au module**

│ ├──module1.php

│ ├──module2.php

│ ├──module3.php

│ ├──.....

│ ├──web.php **// pour les routes en commun**

├──storage

│ ├──app **(7)**

│ │ ├──docs

│ │ ├──exports

│ │ ├──public

│ │ └──report

│ ├──framework

│ │ ├──cache

│ │ │ └──data

│ │ ├──sessions

│ │ ├──testing

│ │ └──views

│ └──logs

└──tests

├──Feature

└──Unit

(1) - L'objectif est de mettre :

- un dossier par module et y mettre les controllers par modules,
- un dossier Models et y mettre les controllers liés au models.

Ainsi, il sera possible de différencier les controllers liés aux models de ceux liés aux modules.

(2) - L'objectif est de :

- consacrer le Model pour les méthodes de relations, setters, getters…,
- faire un repository par Model pour y mettre les méthodes liées au autres traitements (save, edit , get …).

Dans certains cas on aura besoin de créer un module qui n'est pas lié à un modèle et dans ce cas il sera mieux de mettre les méthodes liées dans un Repository.

(3) - Services : Ce dossier doit contenir la liste des apis utilisées par l'app sous forme de classes.

En quelque sorte , c'est la liste des services externes .Il sera mieux de les séparer .

Ce qui facilite la réutilisation et la maintenance des services externes.

(4) **-** Le dossier Support : Ce dossier sera divisé en 2 sous-dossier 'Classes' et 'Files'

- "Classes" : doit être divisé en 2 sous-dossiers :

  -- Global : contient les classes 'helpers' de genre Mail...

  -- Un dossier par fonctionnalitée, exemple : MP4Info.

- "Files" : doit contenir ,généralement, un seul fichier , helpers.php .

(5) **-** Le module pattern sera utilisé, voir le lien suivant :
(https://toddmotto.com/mastering-the-module-pattern/).

Le but est de pouvoir charger uniquement les méthodes dont on a besoin pour chaque view et éviter ainsi les conflits entre les js.

(6) - L’objectif est de minimiser le nombre des fichiers routes et avoir 2 sous dossiers:

- api : contient un seul fichier pour les routes api et ça sera nommé api.php,
- web : contient:

-- un fichier par module qui va inclure les routes liées au models et au module concerné ,

-- et un fichier nommé web.php pour y mettre les routes en commun généralement liées uniquement au models et utilisées dans plusieurs modules .

(7) - Il sera mieux de mettre les documents du projet, les fichiers exportés sous le dossier storage/app.