

Laravel: Conceptes bàsics. Passos a seguir per la creació d'un projecte bàsic

Conceptes bàsics	Passos a seguir per un projecte bàsic
1- Namespace i MVC	1- Creació de base de dades, usuari i permisos amb un script sql
2- Migrations i Taules	2- Creació d'un projecte amb l'ordre laravel
3- Bastiments	3- Configuració del projecte modificant l'arxiu .env
4- Models i Controllers	4- Creació de taules del projecte:
5- Views i Controllers	• Creació dels fitxers de Migrations
6- Blades i Blade Templates	• Execució de les Migrations => Creació taules
7- Routing	5- [opcional] Afegiment de Middleware d'autenticació
8- Request & Responses	6- Desenvolupament dels Models
9- Middleware	7- Desenvolupament dels Controllers
10- Seeders i Factories	8- Creació de Routes
	9- Creació de Views (blades)

MIGRATIONS I TAULES. BACKUPS. INICIACIÓ A SEEDERS

PART 1 - Treballant amb migrations

a) Creant taules dins de la base de dades empresa

1- L'eina **artisan** s'inclou amb cada instal·lació del framework de **Laravel** i necessita l'interpret de **php** per ser interpretada i executada. Aquesta eina es troba sempre a l'arrel del projecte i permet fàcilment, entre altres opcions, crear **taules**, **models**, **controladors** i **vistes**. En el nostre cas, l'ordre **artisan** es trobarà dins del directori **empresa** que és on s'ha instal·lat el framework de **Laravel**.

2- Respecte de les **migrations** de **Laravel**:

- Són uns Scripts PHP que permeten crear, esborrar, reanomenar o canviar l'estructura de les taules d'una base de dades sense haver d'executar ordres MySQL manualment o utilitzant un script SQL.
- Les **migrations** es defineixen dins de fitxers PHP que es troben al subdirectori **database/migrations**. Per defecte ja existeixen unes fitxers de migrations dins del projecte, que són els següents: **failed_jobs**, **password_reset_tokens**, **personal_access_token** i **users**. Aquests scripts es van crear en el moment d'afegir al nostre projecte el framework de **laravel**.
- Hem d'utilitzar l'eina **artisan** per executar les **migrations** que s'aniran definint dins del projecte. Per exemple, per crear taules dins de la base de dades empresa amb les migration per defecte, hem d'anar al directori empresa i executar l'ordre:

```
php artisan migrate
```

i veurem (s'ha de comprovar) que es creen les taules: **failed_jobs**, **password_reset_tokens**, **personal_access_tokens** i **users** dins de la base de dades **empresa**.

NOTA IMPORTANT: Les migrations per defecte es van executar a la sessió 1 (veure el punt l de l'apartat 1.2 del document **laravel.pdf** que es trobava dins de **sessio01pj6f4a14doc1.tar.gz**)

3- En el moment de fer la primera **migration**, es crearà dins de la base de dades **empresa**, una taula amb el nom **migrations** a on deserà els **logs** de les tasques de **creació/modificació/esborrament** de taules que s'han dut a terme dins de la base de dades **empresa**.

4- De totes les taules creades, és especialment important parar atenció a la taula **users** perquè és a on definirem els nostres usuaris de l'aplicació. Dins d'aquesta taula podem trobar, els camps per l'identificador de l'usuari (que és primary key), el nom d'usuari, correu electrònic, contrasenya, etc..

5- Anem a crear una taula per desar les dades dels empleats de l'empresa. Ens caldrà per tant afegir un fitxer dins del directori de **migrations** per poder crear la taula que anomenarem **empleats**. En aquest cas, després d'analitzar les necessitats de l'empres i la nostra futura aplicació, hem arribat a la conclusió que ens calen els següent camps per la nova taula: **id** (identificador), **nom** de tipus **text**, **telefon** de tipus **text**, **email** de tipus **text** i els **timestamps** que ens informin dels accessos a la taula i els seus registres. Per tant, crearem un fitxer de **migrations** i per fer-ho, només ens cal executar dins del directori **empresa**:

```
php artisan make:migration create_empleats_table --create=empleats
```

a on **create_empleats_table** serveix per indicar el nom del fitxer PHP a crear dins de la carpeta **database/migrations**, i **--create=empleats** el nom de la taula a crear dins de la base de dades **empresa**.

A continuació, comprovarem que s'ha creat el fitxer **2026_02_26_103743_create_empleats_table.php** dins del directori **database/migrations**. Compte que **2026_02_36_103743** representa la data i hora de creació i per tant pot ser d'un valor diferent en funció del moment de la creació del fitxer.

6- Ara modificarem el contingut de fitxer **2026_02_26_103743_create_empleats_table.php** per adaptar-lo a les nostres necessitats. Fes que el codi sigui aquest:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateEmpleatsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('empleats', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('email');
            $table->string('telefon');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('empleats');
    }
}
```

NOTA: Les línies en vermell són les nostres modificacions.

Quan executem aquest codi, es crearan la taula **empleats** (que s'esborrarà si ja existeix) amb els camps indicats. El camp **id** ja serà de tipus **primary key**, **bigint(20)**, **unsigned** i **auto_increment**.

7- A continuació, procedirem finalment a la creació de la taula **empleats**. Executarem des del directori **empresa**:

```
php artisan migrate
```

i la taula es crearà.

8- Si vols assegurar-te que la taula s'ha creat, entra dins de **MySQL** i executa:

```
use empresa;  
show tables;  
describe empleats;
```

i comprova que efectivament s'ha creat la taula, amb els camps demanats. Ara, ja sabem crear taules amb Laravel.

9- Consulta també la taula **migrations** de la base de dades **empresa** executant:

```
select * from migrations;
```

i comprova que el resultat similar a:

```
MariaDB [empresa]> select * from migrations;  
+-----+-----+-----+  
| id | migration | batch |  
+-----+-----+-----+  
| 1 | 0001_01_01_000000_create_users_table | 1 |  
| 2 | 0001_01_01_000001_create_cache_table | 1 |  
| 3 | 0001_01_01_000002_create_jobs_table | 1 |  
| 4 | 2026_02_26_103743_create_empleats_table | 2 |  
+-----+-----+-----+  
4 rows in set (0.001 sec)
```

Això és un **log** de les **migrations** realitzades fins ara, que a més a més, ens permet controlar la manera de fer **rollbacks**. Un **rollback** ens permet [revertir la darrera/les N darreres/totes](#) les **migrations** realitzades.

b) Fent rollbacks (revertint) d'una migration

1- Per comprovar que podem fer una tasca extremadament útil com és la de fer **rollbacks**, surt del client **mysql** i reverteix la darrera **migration** realitzada executant des de la carpeta **empresa**:

```
php artisan migrate:rollback
```

i torna a accedir al client MySQL. Comprova que la taula **empleats** ha desaparegut i que dins de la taula **migrations** ha desaparegut l'entrada indicat associada.

A continuació, si no vols que es torni a crear la taula **empleats** en la propera **migration**, entra a **empresa/database/migrations** i esborra el fitxer PHP **2026_02_26_103743_create_empleats_table.php** (el valor inicial del nom del fitxer recorda que pot canviar en funció de l'hora de creació del fitxer).

Ara, ja sabem com revertir i i eliminar els efectes d'una **migration**.

c) Creació de taules i definint característiques dels camps

1- Anem a crear una taula més complexa i amb més camps que seran de diversos tipus. En comptes d'**empleats** la nova taula s'anomenarà **treballadors**. Executa des de la carpeta **empresa**:

```
php artisan make:migration crea_taula_treballadors --create=treballadors
```

de manera que a **empresa/database/migrations** es crearà el fitxer: **aaaa_mm_dd_hhmmss_crea_taula_treballadors.php** (a on **aaaa** és l'any, **mm** és el mes, **dd** és el dia i **hhmmss** és l'hora de creació del fitxer).

2- A continuació, obre el fitxer PHP creat i fes que la estructura de camps de la taula treballadors sigui aquesta:

```
$table->id('tid');
$table->string('nom');
$table->string('cognoms');
$table->string('nif')->unique();
$table->date('data_naixement');
$table->char('sexe',1);
$table->string('adresa');
$table->integer('tlf_fixe');
$table->integer('tlf_mobil');
$table->string('email');
$table->binary('fotografia')->nullable();
$table->boolean('treball_distancia');
$table->enum('tipus_contracte',['temporal','indefinit','en formació','en pràctiques']);
$table->date('data_contractacio');
$table->tinyinteger('categoria');
$table->string('nom_feina',50);
$table->float('sou');
$table->timestamps();
```

NOTA:

- Tipus de columnes: <https://laravel.com/docs/12.x/migrations#available-column-types>
- Modificadors de columnes: <https://laravel.com/docs/12.x/migrations#column-modifiers>
- Tipus d'index: <https://laravel.com/docs/12.x/migrations#available-index-types>

Ara, des del directori **empresa** executa:

```
php artisan migrate
```

A continuació, entra dins de MySQL i executa:

```
use empresa;
show tables;
describe treballadors;
```

i comprova que s'ha creat la taula correctament i que la seva creació s'ha enregistrat dins de la taula de logs.

d) Modificació d'una taula creada

1- Si ara volem modificar un camp de la taula, per exemple, fent que el camp 'nom_feina' tingui **30** caràcters en comptes de **50**, haurem de:

- Editar **aaa_mm_dd_hhmmss_crea_taula_treballadors.php**
- Modificar el camp 'nom_feina' ==> `$table->string('nom_feina',30);`

A continuació, executarem dins del directori **empresa**:

```
php artisan migrate:refresh
```

Després, amb l'ordre **describe treballadors;** dins de **MySQL** comprovarem que s'ha modificat el camp.

2- Així doncs, ara ja sabem:

- Fer migrations
- Revertir i eliminar migrations
- Treballar amb tipus de dades i els seus modificadors
- Modificar migrations

PART 2 – Creant roles d'usuaris dins de la taula users de la base de dades empresa

a) Modificació de la taula users de la base de dades empresa

1- Farem que la base de dades empresa treballi amb 2 tipus d'usuaris amb 2 roles diferents:

- Usuari tipus **basic**
- Usuari tipus **admin**

2- Els roles per ells mateixos no garanteixen cap permís o privilegi especial. Més endavant, s'aprofitaran aquest roles per mitjà de codi PHP per garantir que dins de l'aplicació cada tipus d'usuari tingui els privilegis que vulguem donar-li

3- En el nostre cas, dins de les properes sessions desenvoluparem un codi PHP que garanteixi els següent privilegis:

- Un usuari de tipus **basic** tindrà accés a visualitzar les seves pròpies dades personals.
- Usuari tipus **admin** tindrà accés a:
 - La secció de registre de nous usuaris.
 - Visualitzar/crear/esborrar/actualitzar usuaris.

4- Primer de tot, haurem d'afegir el camp **role** a la taula **users** de la base de dades **empresa**. Anirem a **empresa** i executarem:

```
php artisan make:migration afegeix_camp_role_a_users
```

i comprovem que es crea un fitxer **xxxx_xx_xx_XXXXXX_afegeix_camp_role_a_users.php** dins de **database/migrations**. Compte que **xxxx_xx_xx_XXXXXX** representa l'any, mes, dia i hora de creació del fitxer.

5- Ara obrim el fitxer que hem creat i farem que tingui aquest codi:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AfegeixCampRoleAUsers extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string('role')->after('name')->default('admin');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('role');
        });
    }
}
```

6.- Aquest codi ens permet:

- La creació de la nova columna **role** de tipus **string** després de la columna **name**.
- Els usuaris existents no s'esborraran i per defecte tindran el role **admin**.
- Podem esborrar fàcilment la nova columna si fem un **rollback**.

7.- A continuació executarem: **php artisan migrate**

8- Finalment, comprovarem que s'ha creat la nova columna **role** dins de la taula **users**. Aquest canvi es farà sense perdre usuaris existents en el cas que se n'haguessin creat prèviament.

PART 3 – Utilitzant "seeders" per crear múltiples usuaris dins de la taula users de la base de dades empresa

1- Laravel inclou la possibilitat d'omplir amb dades les taules d'una base de dades abans de començar a utilitzar l'aplicació utilitzant "seeders".

2- El "seeding" (o sembra) d'una base de dades en Laravel permet als desenvolupadors omplir les seves bases de dades amb dades de prova de manera ràpida i eficient. Aquesta funció és molt útil per fer proves de funcionament en fase de desenvolupament i en fase de producció abans d'alliberar l'aplicació.

3- Quan treballem amb "seeders":

- Cada taula de la base de dades està associada a un "seeder" que no és res més que una classe amb un mètode de nom **run** dins del qual inclourem el codi PHP amb les dades que volem introduir.
- El codi amb la classe i les seves dependències estarà a un fitxer que es trobarà dins de **database/seeders**.
- Per cada taula trobarem un fitxer dins de **database/seeders** amb la classe definida dins del fitxer.
- Tots els "seeders" s'hauran d'enregistrar per ser cridats dins de la classe **DatabaseSeeder** del fitxer **database/seeders/DatabaseSeeder.php**.
- Es poden crear seeders i cridar-los per omplir les taules de les bases de dades utilitzant l'ordre de Laravel **php artisan**.
- Una opció típica d'utilització de "seeders" és afegir usuaris a l'aplicació de manera que ja puguem començar a utilitzar-los sense que calgui utilitzar la pròpia aplicació per crear-los.

4- Per crear un nou **Seeder** per omplir la taula d'**usuaris** executa des del directori **empresa**:

```
php artisan make:seeder UsuarisAmbRolesSeeder
```

5- Si després de crear-lo, ens adonem que no volem el fitxer de Seeder pel motiu que sigui, únicament cal esborrar-lo del projecte.

6- Comprova que s'ha creat el fitxer **database/seeders/UsuarisAmbRolesSeeder.php** a on s'ha definit una nova classe de nom **UsuarisAmbRolesSeeder** amb els mètodes necessaris pel nostre objectiu.

7- Ara, desenvoluparem el codi d'**UsuarisAmbRolesSeeder.php**. Aquest codi aprofitarà dues de les múltiples classes i mètodes en que proporciona el framework de Laravel per fer-nos la vida més fàcil. En concret, aprofitarem aquestes dues:

- La classe **Hash** → Com que volem crear usuaris que tinguin la contrasenya emmagatzemada de manera segura per mitjà de funcions **hash**, indicarem a **UsuarisAmbRolesSeeder.php** que volem utilitzar la classe **Hash** que ens proporciona el framework de **Laravel** i que té el mètode **make** que pot crear un codi hash a partir d'una cadena d'entrada. Això és fàcil de fer afegint el codi:

```
use Illuminate\Support\Facades\Hash;
```

- La classe **DB** → El framework de **Laravel** proporciona dins de la classe **DB** el mètode **insert** que permet introduir dades dins d'una taula. Per poder utilitzar aquesta classe hem d'afegir el següent codi PHP:

```
use Illuminate\Support\Facades\DB;
```

8- Ara, ja podem desenvolupar el codi del mètode `run()` de la classe `UsuarisAmbRolesSeeder` aprofitant els mètodes `make` i `insert`:

```
public function run(): void
{
    $llista_usuaris = [
        [
            'name' => 'leniad',
            'role' => 'admin',
            'email' => 'leinad@fjeclot.net',
            'password' => Hash::make('fjeClot25#')
        ],
        [
            'name' => 'aletse',
            'role' => 'basic',
            'email' => 'aletse@fjeclot.net',
            'password' => Hash::make('clotFje25@')
        ],
        [
            'name' => 'igres',
            'role' => 'basic',
            'email' => 'igres@fjeclot.net',
            'password' => Hash::make('Clotfje26@')
        ]
    ];
    DB::table('users')->insert($llista_usuaris);
}
```

En aquest cas és important remarcar que:

- `DB::table('users')` → retorna un objecte que entre d'altres, té el mètode `insert` disponible per insertar dades dins de la taula `users`.
- `$llista_usuaris` és un array indexat de 3 posicions que a cada posició té un array associatiu amb 4 claus: `name`, `role`, `email` i `password` i les dades. Les claus han de correspondre amb el noms de les columnes de la taula `users`.
- El mètode `insert()` insertarà els dos registres definits a `$llista_usuaris` dins de la taula `users`.
- El mètode `make()` convertirà `fjeClot25#`, `fjeClot26@` i `Clotfje26@` en els seus codis equivalents `hash`.

9- Abans de poder executar l'ordre de **Laravel** per omplir de dades una taula amb el nou "seeder" **UsuarisAmbRolesSeeder**, s'haurà de registrar el nou "seeder" dins del mètode **run** la classe classe **DatabaseSeeder** que es troba a **database/seeder/DatabaseSeeder.php**. Això en assegura que el nou "seeder" serà cridat i el seu codi executat.

El codi complet de **DatabaseSeeder.php** en el cas particular d'aquest exemple es pot veure a continuació:

```
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    // use WithoutModelEvents;

    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // User::factory(10)->create();

        // User::factory([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);

        $this->call([
            UsuarisAmbRolesSeeder::class,
        ]);
    }
}
```

NOTA: He comentat una part del codi per defecte perquè no s'utilitza per poder ser-nos útil en algun moment.

10- Ara ha podem executar l'ordre que ens permet omplir la taula **users** amb les dades. Des del directori **empresa**, executa l'ordre d'**artisan**:

```
php artisan db:seed
```

11.- Finalment, comprova que s'han creat els nous registres dins de la taula **users** de la base de dades **empresa**.

12.- Desafortunadament no existeix una opció de rollback de seeders, de manera que si ens equivoquem tenim altres opcions:

- Esborrar manualment els usuaris amb codi SQL.
- Retornar les dades a partir d'un backup com veurem a continuació
- Utilitzar la proposta d'aquest projecte: <https://github.com/sonole/laravel-db-seed-rollback>
- Una mica radical i s'ha d'anar amb compte→ Fer un **refresh** de tots els **seeders**:
 - Commenta els seeders que no vols
 - Executa: **php artisan migrate:refresh --seed**

PART 4 – Backup de les taules de la base de dades empresa

1- Instal·la amb **composer** el paquet **Laravel-backup** executant dins de la carpeta del projecte (que en el nostre cas és **empresa**):

```
composer require spatie/laravel-backup
```

2- Afegiu una contrasenya d'enciptació del backup. Dins del fitxer **.env** afegiu el paràmetre

```
BACKUP_ARCHIVE_PASSWORD
```

i fes que valgui:

```
BACKUP_ARCHIVE_PASSWORD="FjeClot26#"
```

Un bon lloc per afegir aquest paràmetre és després del paràmetre **DB_PASSWORD** de la secció de configuració de **MySQL**.

3- Fes un **backup** de les taules del projecte **empresa**, executa dins del directori del projecte (que en el nostre cas és **empresa**):

```
php artisan backup:run
```

4- Comprova que s'ha creat un backup dins del directori **storage/app/private/Laravel** amb el nom **AAAA-MM-DD-HH-mm-SS.zip** a on **AAAA-MM-DD-HH-mm-SS** representa l'any, mes, dia, hora, minut i segon de creació del fitxer **.zip**.

5- Si descomprimeixis el fitxer **.zip** es crearan 2 carpetes:

- **db-dump** amb un script SQL que si s'executa regenera les bases de dades, les taules i totes les dades del projecte.
- **var** amb tots els directoris i tots els fitxers de l'aplicació.

6- Pàgina oficial: <https://spatie.be/docs/laravel-backup>

7- Aquest paquet no té una opció de **RESTORE** per defecte. Hauràs de fer aquesta acció:

- Manualment
- Utilitzant altres frameworks com per exemple el proporcionat pel següent projecte de Github: <https://github.com/stefanzweifel/laravel-backup-restore>. Aquest projecte no ha estat provat a l'hora d'escriure aquest document però té una ampla documentació a la següent adreça: <https://stefanzweifel.dev/posts/2023/06/15/introducing-laravel-backup-restore/>.

8- Restaurant amb **laravel-backup-restore**:

- Instal·la el framework. Executa: **composer require wnx/laravel-backup-restore** (dins del directori del projecte)
- Executa dins del directori del projecte: **php artisan backup:restore**