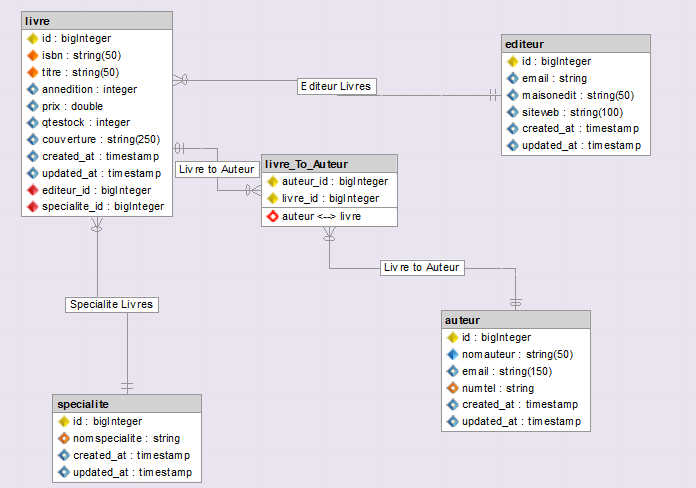
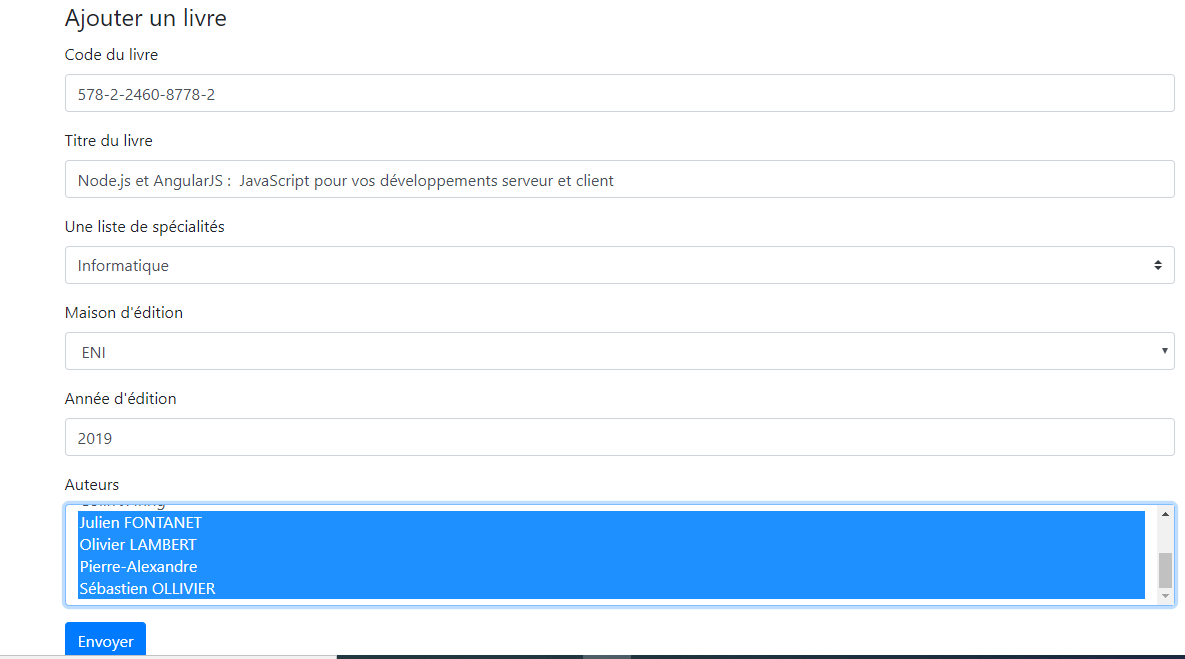
 Atelier : Mise en place d’une application Gestion Bibliothèque

On se propose d’automatiser la recherche des livres dans une bibliothèque. A cet effet, on considère la base de données appelée «**dbbiblio»** déduite d’une partie du diagramme de classes UML :

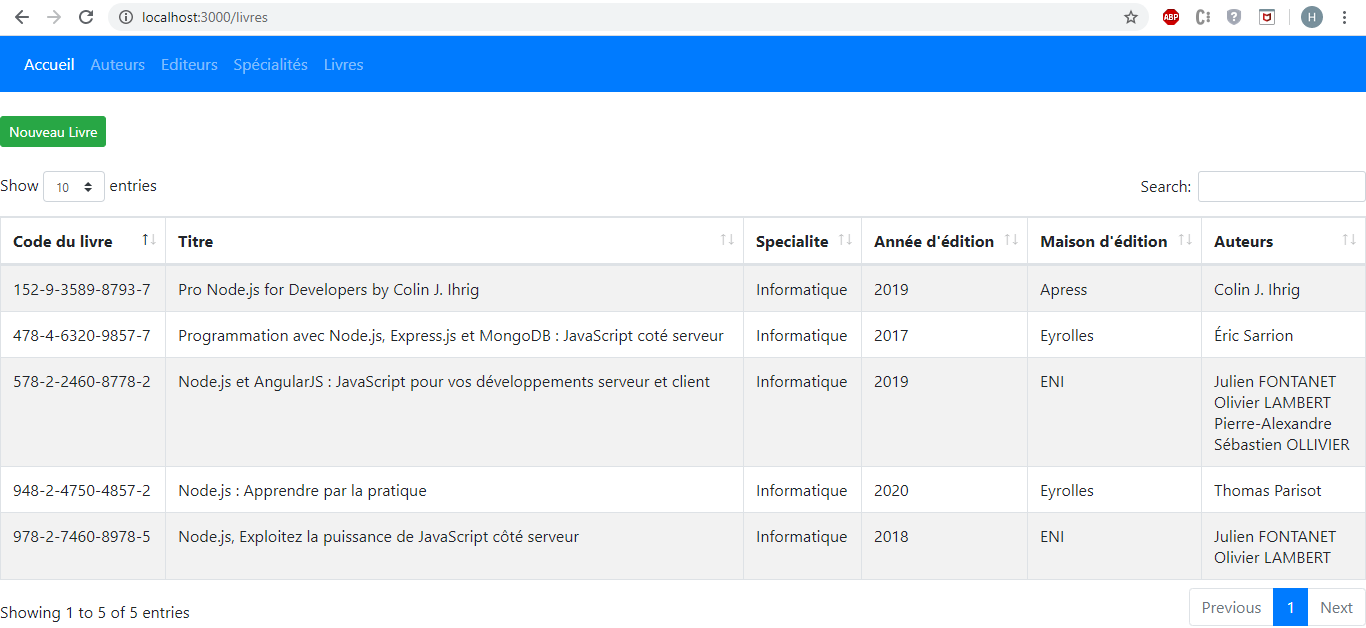


Le résultat du projet doit nous permettre de faire la recherche selon l’un des critères de recherche suivants : titre, auteur, éditeur, spécialité et l’intervalle d’année de publication.

**Formulaire d’ajout d’un Livre**



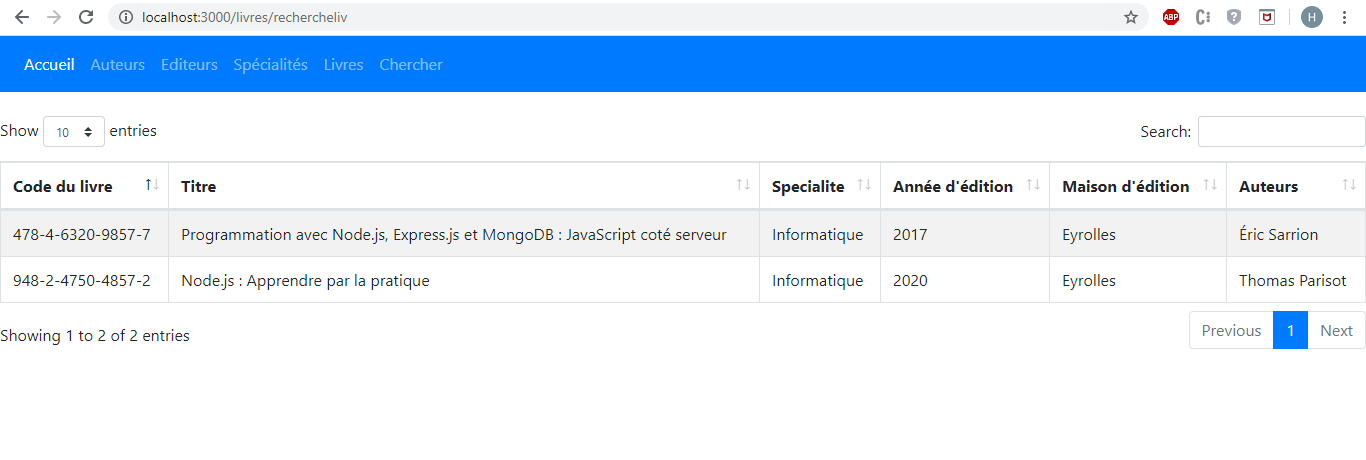
**Formulaire d’affichage de la liste des livres**



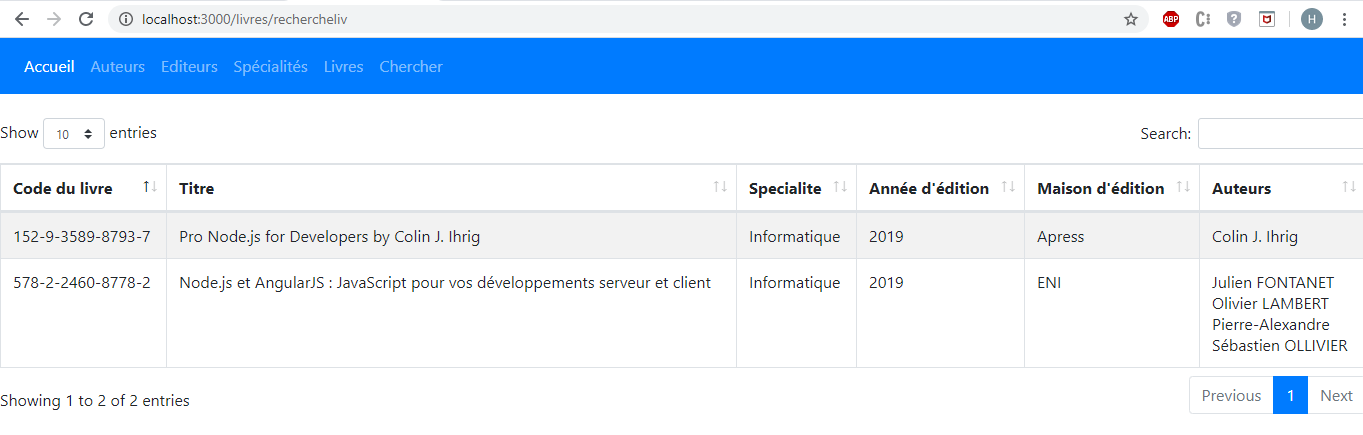
**Formulaire de Recherche Multicritères de livres (Titre, Spécialitée, Maison d’édition, Année d’édition)**



**Si on sélectionne Maison d’édition et on appuie sur le bouton Rechercher, l’écran suivant s’affiche :**



**Si on sélectionne Année d’édition et on appuie sur le bouton Rechercher, l’écran suivant s’affiche :**



Questions :

* Créer la base de données sous le SGBD MySQL.
* Créer une application Laravel VueJS, permettant :

1. De créer un nouveau projet Gestbiblio contenant la partie Backend et frontEnd
2. D’ajouter, modifier, supprimer et consulter des livres de la bibliotheque
3. Chercher un livre selon l’un des critères indiqués ci-dessus et afficher le résultat.

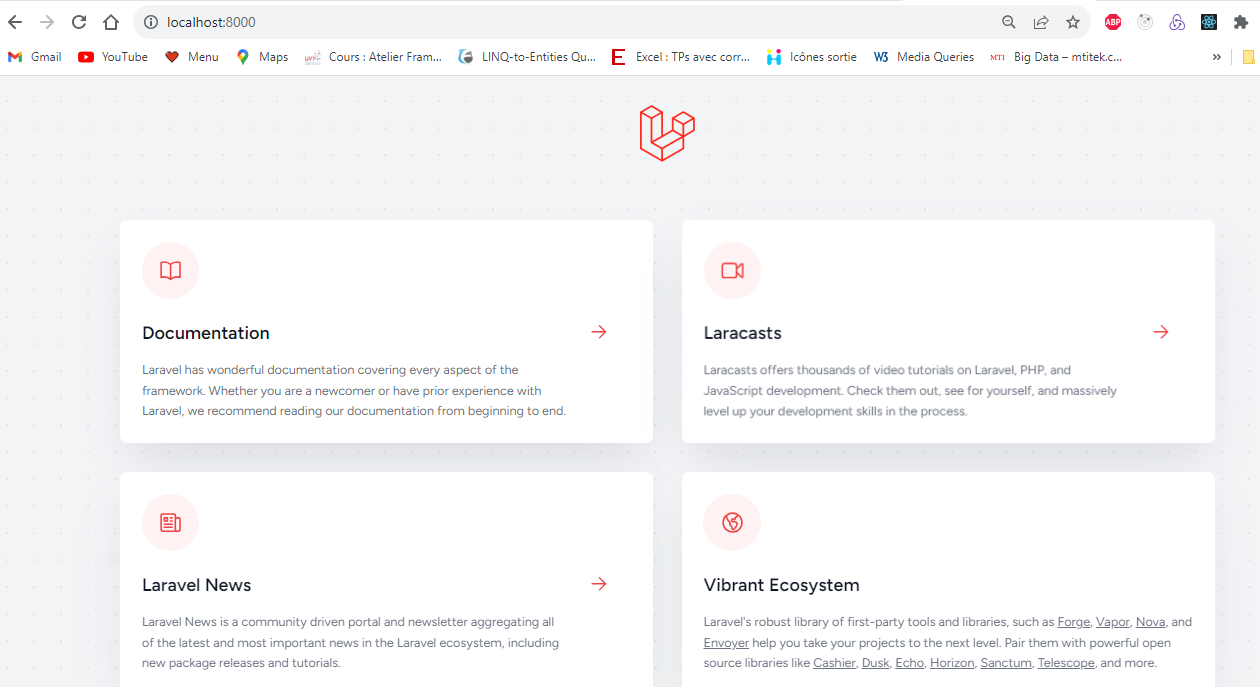
Composer create-project laravel/laravel Gestbiblio

Cd Gestbiblio

Code .

Terminal🡪new terminal

Php artisan serve



1/ fichier .env pour paramétrer la base de données

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=dbbiblio

DB\_USERNAME=root

DB\_PASSWORD=

2/ Créer le Model de chaque classe dans le modèle de classe

Model Specialite

php artisan make:model Specialite -mc --api

Model Editeur

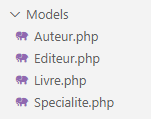
php artisan make:model Editeur -mc --api

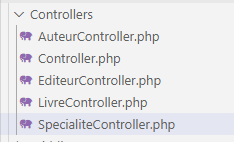
Model Auteur

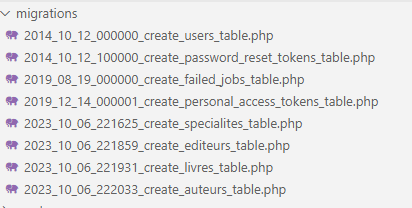
php artisan make:model Auteur -mc --api

Model Livre

php artisan make:model Livre -mc --api







create\_specialites\_table.php

public function up(): void

    {

        Schema::create('specialites', function (Blueprint $table) {

            $table->id();

            $table->string("nomspecialite",50)->unique();

            $table->timestamps();

        });

    }

create\_editeurs\_table

public function up(): void

    {

        Schema::create('editeurs', function (Blueprint $table) {

            $table->id();

            $table->string("maisonedit",50)->unique;

            $table->string("siteweb",100)->unique;

            $table->string("email",150)->unique;

            $table->timestamps();

        });

    }

create\_auteurs\_table

public function up(): void

    {

        Schema::create('auteurs', function (Blueprint $table) {

            $table->id();

            $table->string("nomauteur",50);

            $table->string("email",150)->unique;

            $table->string("numtel",15)->nullable;

            $table->timestamps();

        });

    }

create\_livres\_table

public function up(): void

    {

        Schema::create('livres', function (Blueprint $table) {

            $table->id();

            $table->string("isbn",100)->unique;

            $table->string("titre",100)->unique;

            $table->integer("annedition");

            $table->double("prix");

            $table->integer("qtestock");

            $table->string("couverture");

            $table->unsignedBigInteger("specialite");

            $table->foreign('specialite')

            ->references('id')

            ->on('specialites')

            ->onDelete('restrict');

            $table->unsignedBigInteger("maised");

            $table->foreign('maised')

            ->references('id')

            ->on('editeurs')

            ->onDelete('restrict');

            $table->timestamps();

        });

    }

Creation de la table non porteuse de données (table pivot)

* Les relations **n..n** sont légèrement plus compliquées à représenter en base que les relations **1..1** et **1..n**. En effet, elles nécessitent une table “pivot” qui contient une référence aux clés primaires des deux autres tables.

php artisan make:migration create\_livre\_auteur\_table

public function up(): void

    {

        Schema::create('livre\_auteur', function (Blueprint $table) {

            $table->bigInteger('auteur\_id')->unsigned();

            $table->bigInteger('livre\_id')->unsigned();

            $table->primary(['auteur\_id','livre\_id']);

            $table->foreign('auteur\_id')->references('id')->on('auteurs');

            $table->foreign('livre\_id')->references('id')->on('livres');

        });

    }

Models

Auteur.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Auteur extends Model

{

    use HasFactory;

    protected $fillable = [

        'nomauteur',

        'email',

        'numtel'

    ];

    public function livres()

    {

        return $this->belongsToMany(Livre::class);

    }

}

Editeur.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Editeur extends Model

{

    use HasFactory;

    protected $fillable = [

        'maisonedit',

        'siteweb',

        'email'

    ];

    public function livres()

    {

        return $this->hasMany(Livre::class);

    }

}

Livre.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Livre extends Model

{

    use HasFactory;

    protected $fillable = [

        'isbn',

        'titre',

        'annedition',

        'prix',

        'qtestock',

        'couverture',

        'specialite\_id',

        'editeur\_id'

    ];

    public function editeur()

    {

        return $this->belongsTo(Editeur::class,'editeur\_id');

    }

    public function specialite()

    {

        return $this->belongsTo(Specialite::class,'specialite\_id');

    }

    public function auteurs()

    {

return $this->belongsToMany(Auteur::class, **'livre\_auteur', 'livre\_id', 'auteur\_id'**);

    }

}

Specialite.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Specialite extends Model

{

    use HasFactory;

    protected $fillable = ['nomspecialite'];

    public function livres()

    {

        return $this->hasMany(Livre::class);

    }

}

Migration

php artisan migrate