Para desenvolver uma aplicação web para gerenciar transações financeiras com Laravel no backend e Angular no frontend, seguiremos um fluxo dividido entre a criação das duas partes da aplicação. Abaixo, descrevo os principais passos.

Backend (Laravel)

1. Configuração do Projeto Laravel

1. Inicie um novo projeto Laravel para o backend:

```
bash
Copiar código
composer create-project --prefer-dist laravel/laravel financial-manager
```

2. Instale o pacote Laravel para CORS (permitir a comunicação entre o backend e o frontend):

```
bash
Copiar código
composer require fruitcake/laravel-cors
```

3. No arquivo `config/cors.php`, ajuste para permitir requisições do Angular:

```
php
Copiar código
'allowed_origins' => ['http://localhost:4200'], // Porta padrão do Angular
```

4. Rode o servidor de desenvolvimento Laravel:

```
bash
Copiar código
php artisan serve
```

2. Configuração do Banco de Dados

5. No arquivo `.env`, configure o banco de dados:

```
env
Copiar código
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=nome_do_banco
DB_USERNAME=usuario
DB_PASSWORD=senha
```

6. Crie o banco de dados no MySQL:

```
sql
Copiar código
CREATE DATABASE financial_manager;
```

3. Criar a Migration e a Model de Transação

7. Crie a migration para a tabela de transações:

```
bash
Copiar código
php artisan make:migration create_transactions_table --
create=transactions
```

8. No arquivo `database/migrations/*_create_transactions_table.php`, defina a estrutura da tabela:

```
php
Copiar código
public function up()
{
    Schema::create('transactions', function (Blueprint $table) {
        $table->id();
        $table->string('description');
        $table->decimal('amount', 10, 2);
        $table->enum('type', ['income', 'expense']); // Tipo:
receita ou despesa
        $table->timestamps();
    });
}
```

9. Execute a migration:

```
bash
Copiar código
php artisan migrate
```

10. Crie o model `Transaction`:

```
bash
Copiar código
php artisan make:model Transaction
```

11. No arquivo `app/Models/Transaction.php`, defina os campos:

```
php
Copiar código
protected $fillable = ['description', 'amount', 'type'];
```

4. Criar o Controller de Transações

12. Crie um controller para gerenciar as transações:

```
bash
Copiar código
php artisan make:controller TransactionController --resource
```

13. No arquivo `app/Http/Controllers/TransactionController.php`, implemente as funcionalidades CRUD:

```
php
Copiar código
use App\Models\Transaction;
use Illuminate\Http\Request;

class TransactionController extends Controller
{
    public function index()
    {
        return Transaction::all();
    }
```

```
    public function store(Request $request)
    {
        $request->validate([
            'description' => 'required|string',
            'amount' => 'required|numeric',
            'type' => 'required|in:income,expense',
        ]);

        return Transaction::create($request->all());
    }

    public function show($id)
    {
        return Transaction::findOrFail($id);
    }

    public function update(Request $request, $id)
    {
        $transaction = Transaction::findOrFail($id);

        $request->validate([
            'description' => 'required|string',
            'amount' => 'required|numeric',
            'type' => 'required|in:income,expense',
        ]);

        $transaction->update($request->all());

        return $transaction;
    }

    public function destroy($id)
    {
        $transaction = Transaction::findOrFail($id);
        $transaction->delete();

        return response()->json(null, 204);
    }
}
```

5. Adicionar Rotas

14. No arquivo `routes/api.php`, defina as rotas para o controller:

```
php
Copiar código
use App\Http\Controllers\TransactionController;

Route::apiResource('transactions', TransactionController::class);
```

15. Teste o backend usando Postman ou outra ferramenta para certificar-se de que está funcionando corretamente.

Frontend (Angular)

1. Configuração do Projeto Angular

16. Instale o Angular CLI, se não tiver instalado:

```
bash
Copiar código
npm install -g @angular/cli
```

17. Crie o projeto Angular:

```
bash
Copiar código
ng new financial-manager-frontend
```

18. Entre na pasta do projeto e inicie o servidor:

```
bash
Copiar código
cd financial-manager-frontend
ng serve
```

2. Criar o Service para Comunicação com a API

19. Gere um service Angular para interagir com o backend:

```
bash
Copiar código
ng generate service services/transaction
```

20. No arquivo `src/app/services/transaction.service.ts`, defina o serviço:

```
typescript
Copiar código
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TransactionService {
  private apiUrl = 'http://localhost:8000/api/transactions';

  constructor(private http: HttpClient) { }

  getTransactions(): Observable<any> {
    return this.http.get(this.apiUrl);
  }

  createTransaction(transaction: any): Observable<any> {
    return this.http.post(this.apiUrl, transaction);
  }

  updateTransaction(id: number, transaction: any): Observable<any> {
    return this.http.put(`${this.apiUrl}/${id}`, transaction);
  }

  deleteTransaction(id: number): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${id}`);
  }
}
```

3. Criar o Componente para o Formulário

21. Gere o componente para o formulário de transações:

```
bash
Copiar código
ng generate component components/transaction-form
```

22. No arquivo `src/app/components/transaction-form/transaction-form.component.ts`, defina o formulário:

```
typescript
Copiar código
import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { TransactionService } from
'src/app/services/transaction.service';

@Component({
  selector: 'app-transaction-form',
  templateUrl: './transaction-form.component.html',
})
export class TransactionFormComponent {
  transactionForm: FormGroup;

  constructor(private fb: FormBuilder, private transactionService:
TransactionService) {
    this.transactionForm = this.fb.group({
      description: ['', Validators.required],
      amount: [0, Validators.required],
      type: ['income', Validators.required],
    });
  }

  onSubmit() {

this.transactionService.createTransaction(this.transactionForm.value
).subscribe(response => {
      console.log('Transação criada:', response);
    });
  }
```

```
}
```

23. No arquivo `src/app/components/transaction-form/transaction-form.component.html`, defina o HTML do formulário:

```
html
Copiar código
<form [formGroup]="transactionForm" (ngSubmit)="onSubmit()">
  <label for="description">Descrição</label>
  <input formControlName="description" id="description" type="text">

  <label for="amount">Valor</label>
  <input formControlName="amount" id="amount" type="number">

  <label for="type">Tipo</label>
  <select formControlName="type" id="type">
    <option value="income">Receita</option>
    <option value="expense">Despesa</option>
  </select>

  <button type="submit">Salvar</button>
</form>
```

4. Listar as Transações

24. Gere o componente para listar as transações:

```
bash
Copiar código
ng generate component components/transaction-list
```

25. No arquivo `src/app/components/transaction-list/transaction-list.component.ts`, busque as transações:

```
typescript
Copiar código
import { Component, OnInit } from '@angular/core';
import { TransactionService } from
'src/app/services/transaction.service';
```

```
@Component({
  selector: 'app-transaction-list',
  templateUrl: './transaction-list.component.html',
})
export class TransactionListComponent implements OnInit {
  transactions: any[] = [];

  constructor(private transactionService: TransactionService) { }

  ngOnInit() {
    this.transactionService.getTransactions().subscribe(data => {
      this.transactions = data;
    });
  }

  deleteTransaction(id: number) {
    this.transactionService.deleteTransaction(id).subscribe(() => {
      this.transactions = this.transactions.filter(t => t.id !==
id);
    });
  }
}
```

26. No arquivo `src/app/components/transaction-list/transaction-list.component.html`, exiba as transações:

```
html
Copiar código
<table>
  <thead>
    <tr>
      <th>Descrição</th>
      <th>Valor</th>
      <th>Tipo</th>
      <th>Ações</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let transaction of transactions">
      <td>{{ transaction.description }}</td>
      <td>{{ transaction.amount }}</td>
      <td>{{ transaction.type }}</td>
```

```
      <td>
        <button
(click)="deleteTransaction(transaction.id)">Excluir</button>
      </td>
    </tr>
  </tbody>
</table>
```

Conclusão

Com essas etapas, você terá um sistema completo de gerenciamento de transações financeiras com Laravel no backend e Angular no frontend. O formulário permite cadastrar receitas e despesas, e a listagem permite visualizar, editar e excluir as transações.