Criando as conexões no ambiente Laravel

Definindo a conexão com o banco de dados que se encontra no contêiner que fora configurado no capítulo passado.

- Criando e Estabelecendo conexão com o Banco de Dados do Guavira

Criando e Estabelecendo conexão com o Banco de Dados do Guavira

6. Configuração da Conexão com o Banco de Dados

Após realizar o deploy do contêiner responsável pelo banco de dados, é necessário configurar uma conexão com ele na API Laravel. Esse passo é fundamental para garantir que as *migrations* e demais operações relacionadas ao banco de dados sejam executadas na base correta.

Essa configuração é especialmente importante porque a aplicação possui conexões com diversos bancos de dados institucionais, como os utilizados pelo RU, E-Campus, ambiente de testes e outros serviços. Portanto, antes de criar ou executar qualquer *migration*, é preciso garantir que a conexão do banco de dados da Seleção PRPPG esteja devidamente cadastrada.

Para realizar essa configuração, serão necessárias algumas variáveis de ambiente. Por motivos de segurança, os valores reais não serão apresentados nesta documentação, sendo substituídos por exemplos fictícios. As credenciais corretas devem ser solicitadas pelos canais oficiais da organização.

```
SELECAO_PRPPG_DB_CONNECTION=pgsql
SELECAO_PRPPG_DB_HOST=postgres
SELECAO_PRPPG_DB_PORT_FORWARD=5450
SELECAO_PRPPG_DB_PORT=5432
SELECAO_PRPPG_DB_USERNAME=admin
SELECAO_PRPPG_DB_PASSWORD=admin123
SELECAO_PRPPG_DB_DATABASE=prppg
```

7. Adicionando a Conexão no Laravel

Com as variáveis de ambiente configuradas, o próximo passo é acessar o arquivo:

```
api/config/database.php
```

Nesse arquivo encontra-se o array associativo `connections`, responsável por armazenar todas as conexões de banco de dados utilizadas pela aplicação. Cada conexão segue a estrutura:

```
nome_da_conexao => [
    // configurações
]
```

Dentro desse array, deve-se criar uma nova entrada correspondente ao banco de dados da Seleção PRPPG.

```
    env('SELECAO_PRPPG_DB_CONNECTION', 'pgsql_selecao_prppg') => [
            'driver' => 'pgsql',
            'url' => env('DB_URL'),
            'host' => env('SELECAO_PRPPG_DB_HOST', '127.0.0.1'),
            'port' => env('SELECAO_PRPPG_DB_PORT', '5432'),
            'database' => env('SELECAO_PRPPG_DB_DATABASE', 'laravel'),
            'username' => env('SELECAO_PRPPG_DB_USERNAME', 'root'),
            'password' => env('SELECAO_PRPPG_DB_PASSWORD', ''),
            'charset' => env('DB_CHARSET', 'utf8'),
            'prefix' => '',
            'prefix_indexes' => true,
            'search_path' => 'public',
            'sslmode' => 'prefer',
        ],
```

A configuração da conexão é composta por diversos parâmetros, cada um responsável por definir uma característica específica da comunicação com o banco de dados:

- O nome da conexão e o valor padrão utilizado caso a variável de ambiente correspondente não seja encontrada;
- O driver de banco de dados utilizado pela conexão;
- A URL de conexão, utilizada como configuração genérica dentro do projeto;
- O host do banco de dados e seu valor padrão;
- A porta em que o serviço está escutando e seu valor padrão;
- O nome do banco de dados e seu valor padrão;
- O usuário utilizado para autenticação e seu valor padrão;
- A senha da conexão e seu valor padrão;
- O conjunto de caracteres (*charset*) utilizado pelo banco de dados.

8. Utilizando a Conexão nas Migrations

Com a conexão devidamente cadastrada no Laravel, resta apenas informar às *migrations* qual conexão deverá ser utilizada durante sua execução.

Isso pode ser feito definindo o atributo `$connection` na classe da migration:

```
return new class extends Migration
{
    /**
     * The database connection used by the migration.
     *
     * @var string
     */
    protected $connection = 'pgsql_ru';

    /**
     * Class constructor.
     */
    public function __construct()
    {
        // if tests are running, then change the testing connection to pgsql_testing
        if (app()->environment('testing')) {
            $this->connection = 'pgsql_testing';
        }
```

```
    }
```

Ao especificar essa propriedade, o Laravel executará todas as operações daquela migration utilizando a conexão indicada, garantindo que as tabelas sejam criadas, alteradas ou removidas no banco de dados correto. Dessa forma, evita-se que alterações sejam aplicadas acidentalmente em outros bancos de dados configurados na aplicação.