02

Migrations

Capítulo 6 - Migrations

Já estamos com nossa aplicação protegida. Mas ainda falta adicionar uma outra feature bem importante. Vamos adicionar categorias nos produtos, afinal nosso sistema começou a crescer e precisamos de uma organização.

Se agora teremos categorias, elas precisam ser salvas no banco de dados. Vamos acessá-lo pelo Terminal:

```
mysql -urot -p
```

Queremos usar a base de dados "estoque":

```
mysql> use estoque;
...
Database changed
```

Lembre-se que temos duas tabelas, a de usuários e a de produtos. As categorias serão salvas em uma terceira tabela, então podemos criá-la fazendo `mysql> create table categorias` .

Mas pensemos que em alguma situação precisaremos desfazer algum erro ou excluir uma categoria. Pior ainda seria se o sistema fosse grande e cada cliente possuísse uma versão diferente do banco de dados. Precisamos ter a capacidade de navegar entre essas versões, migrar de uma para outra. Existe, então, no Cake PHP, a ideia de **migrations**. São alguns arquivos que dizem como o nosso banco de dados cresce e evolui no decorrer da aplicação.

Não se preocupe, apesar de até agora não termos utilizado as migrations, não perderemos nada do que já construímos e não começaremos tudo do zero. Os desenvolvedores do cake PHP, já pensando nisso, criou uma maneira de pegarmos nossa base de dados e transformá-la em migrations, controlando, assim, suas versões.

Abrimos uma nova aba do Terminal e navegamos até a pasta do nosso projeto:

```
cd [diretório onde foi salvo]/estoque
```

Dentro dela, acessamos o diretório "bin/cake" criando uma foto do banco de dados. Esta primeira versão terá o nome de "Initial"

```
sudo bin/cake bake migration_snapshot Initial
```

```
aluras-Mac-mini:estoque alura$ sudo bin/cake bake migration_snapshot Initial
Password:

Welcome to CakePHP v3.0.11 Console
---------------------------------------------------------------
App : src
Path: /Users/alura/Documents/renan/estoque/src/
---------------------------------------------------------------

Creating file /Users/alura/Documents/renan/estoque/config/Migrations/20150820074
940_initial.php
Wrote `/Users/alura/Documents/renan/estoque/config/Migrations/20150820074940_ini
tial.php`
Marking the snapshot 20150820074940_initial as migrated...

Welcome to CakePHP v3.0.11 Console
---------------------------------------------------------------
App : src
Path: /Users/alura/Documents/renan/estoque/src/
---------------------------------------------------------------
using migration path /Users/alura/Documents/renan/estoque/config/Migrations
Migration successfully marked migrated !
```

A Migração foi criada para o nosso banco de dados e marcada, ou seja, a colocou como a versão atual.

Dentro da pasta config, outra foi criada com o nome "Migrations" e dentro desta o nosso arquivo "Initial". Ele possui uma Classe "Initial" que se estende outra com nome "AbstractMigration":

```
class Initial extends AbstractMigration
```

Ela é responsável por pegar todo seu código PHP, converter para SQL e mandar para nosso banco de dados. Agora queremos evoluí-lo, para isso criamos uma migração que, por sua vez, será responsável por criar a tabela de categorias:

```
sudo bin/cake migrations create CriaCategoriasTable
```

Dentro da pasta Migrations foi criado um novo arquivo chamado "CriaCategoriasTable". Dentro de sua Classe há o método `change()`, o qual serão passadas as alterações que precisam acontecer dentro do banco de dados. A primeira delas é criar uma tabela para as categorias:

```
public function change()
{
    $table = $this->table('categorias');
    $table->addColumn('nome','string');
    $table->create();
}
```

No Terminal onde o banco de dados está aberto, se tentarmos visualizar a tabela de categorias:

```
mysql> select * from categorias;
```

Retornará um erro dizendo que a tabela não existe. Precisamos migrar para uma versão atual, então fazemos na outra janela do Terminal:

```
sudo bin/cake migrations migrate
```

Se tentarmos novamente visualizar a tabela de categorias:

```
mysql> select * from categorias;
Empty set (0.00 sec)
```

De fato, a tabela existe, vazia. Podemos também listar as tabelas:

```
mysql> show tables;
+-------------------+
| Tables_in_estoque |
+-------------------+
| categorias        |
| phinxlog          |
| produtos          |
| users             |
+-------------------+
4 rows in set (0.00 sec)
```

A tabela "phinxlog" salva alguns dados, como a versão do banco de dados. Vamos ver a descrição da tabela "categorias":

```
mysql> desc categorias;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| nome  | varchar(225) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

O nome de uma categoria não precisa ter 255 caracteres. Vamos reduzir para 50, mas já criamos a tabela e teríamos de rodar outra migration. Se queremos voltar a versão do banco de dados:

```
sudo bin/cake migrations rollback
```

E assim as últimas alterações são desfeitas. Inclusive a tabela categorias foi excluída:

```
mysql> show tables;
+-------------------+
| Tables_in_estoque |
+-------------------+
| phinxlog          |
| produtos          |
| users             |
+-------------------+
3 rows in set (0.00 sec)
```

E agora sim, no código, passamos o parâmetro de limite de caracteres da tabela "categorias":

```
public function change()
{
    $table = $this->table('categorias');
    $table->addColumn('nome','string',[
        'limit' => 50
    ]);
    $table->create();
}
```

Fazemos novamente a migração, a tabela de categorias será criada e visualizemos seus dados:

```
mysql> desc categorias;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| nome  | varchar(50) | NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Perceba que o limite de caracteres é 50. Mas como o sistema sabia que, ao darmos o *rollback*, a tabela de categorias deveria ser excluída? É o que o método `change()` faz. Nele escrevemos os códigos para criação de, por exemplo, tabelas e, no momento do *rollback*, fica claro o que desfazer. Este método serve tanto para atualizar quanto desfazer.