STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika pre prácu s databázou
DUtí(m)

Tím číslo: 18
Členovia tímu: Bende Tomáš, Búcsiová Veronika, Horniczká Veronika, Melicherčík Jozef, Pacher Marek, Pavkovček Filip, Slíž Boris, Sojka Michal
Akademický rok: 2018/2019
Dátum vypracovania: 18.11.2018
Dátum poslednej úpravy: -
Vypracoval: Jozef Melicherčík

1. Úvod

Metodika pre prácu s databázou opisuje postup pre prvotné spustenie lokálnej databázy a spôsob práce, konvencie a štandardy pre vytváranie tzv. *migrations* a *seeds*. Metodika je určená pre všetkých členov tímu, ktorí sa aktívne podieľajú na vývoji projektu.

Nadväzujúce a súvisiace metodiky a dokumenty:

- Inštalačná príručka
- Metodika verziovania kódu

2. Prvé spustenie databázy

Bližšie je celý tento postup opísaný v Inštalačnej príručke.

Inštalácia Postgres 10.5-2

Pri inštalácie je možné zvoliť možnosť automatického spúšťania Postgres serveru pri štarte počítača. Rovnako tak je možné zvoliť komunikačný port (deafult: 5432).
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

Inštalácia Composer

Inštalácia Composer https://getcomposer.org/

Nastavenie konfiguračného súboru .env

DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=dilema
DB_USERNAME=postgres
DB_PASSWORD=postgres

Príkazy po poradí v cmd (cmd je stále nastavené do priečinku projektu):

a. ***composer update --no-scripts***

b. ***php artisan migrate:reset***

Spustiť ešte pred *pull* príkazom z *master vetvy* alebo svojej vetvy, *rollbackne* to všetky tabuľky. Ak bude *pull* príkaz spustený skôr, je potrebné tabuľky odstrániť ručne.

c. V *php.ini* (C:xampp\php\php.ini) odkomentovať:

- *extension=pdo_pgsql*
- *extension=pgsql*

d. ***composer dump-autoload***

Tento príkaz upraví nekonzistentnosť lokálnej databázy a dát v priečinkoch projektu.

*e. **php artisan migrate:refresh --seed***

V *Pgadmin* sa po *migrate* príkaze vytvoria nové tabuľky a naplnia dátami.

f. Po update databázy je potrebné v priečinku projektu zadať ešte príkazy:

composer update
php artisan migrate

Po úspešnej inštalácií a spustení je projekt dostupný na adrese:
http://localhost:localhost_port /nazov_projektu/public/dilema

Login:
mail: *fiittim10@gmail.com*
heslo: *dilema*

3. Vytvárania a práca s „migrations“

Sekcia opisuje spôsob vytvárania tzv. *migrations* a definuje základné konvencie pre písanie kódu a pomenovávanie jednotlivých častí kódu.

V projektovom priečinku, časť *database/migrations,* sa nachádzajú všetky súbory, ktoré vytvárajú tabuľky databázy pre lokálne vývojové prostredie. Najpodstatnejším faktom pri migráciách je to, že sa pri príkaze *migrate:refresh* vytvárajú v lokálnej databáze tabuľky v poradí od hora nadol. To znamená, že pri vytváraní *migrations* je nutné myslieť na vzájomné väzby medzi tabuľkami. Napríklad, pokiaľ existuje tabuľka *users* a na ňu napojená väzobná tabuľka *user_type*, je nutné ako prvú vytvoriť migráciu pre tabuľku *users.* Je to z toho dôvodu, že väzobná tabuľka *user_type* bude s určitosťou obsahovať, ako cudzí kľúč, primárny kľúč z tabuľky *users*. Pokiaľ by boli migrácie vytvorené v opačnom poradí, príkaz *migrate:refresh* skončí chybovým hlásením.

Nasledujúce kroky opisujú postup vytvárania *migrations* a základné konvencie ich písania.

3.1. Vytvorenie novej migrácie

Pre vytváranie nových migrácií je vhodné využívať základnú funkcionalitu Laravel frameworku, ktorá umožňuje prostredníctvom príkazového riadku vytvoriť novú migráciu. Na začiatok je potrebné otvoriť príkazový riadok a nastaviť ho do priečinku projektu. Následne prostredníctvom *artisan* príkazu *php artisan make:migration nazov_migracie* Laravel vytvorí nový súbor pre migráciu aj s časovým označením na začiatku názvu projektu. To zaručí správne poradie vytvárania tabuliek pri príkaze *migrate:refresh*.

Pri vytváraní migrácií je nutné mať na pamäti, že sa vždy vytvárajú dva základné typy pre každú tabuľku samostatne. Prvým typom je migrácia *create* a druhým migrácia *update*. Oba typy sa môžu využiť pri príkaze *migrate:refresh*, prvý v prípade prvého spustenia lokálnej databázy a druhý pre prípad aktualizácie lokálnej databázy (napr. po git príkaze *pull*).

Pomenovávanie súborov s migráciami sa musí riadiť základnými konvenciami, kde v názve súboru je definovaný čas jeho vytvorenie (zabezpečí Laravel framework), nasledovaný názvom súboru s koncovkou .php. Názov súboru má pritom nasledovný tvar:

create_nazov-tabulky (napr. 2014_10_12_000000_create_user_types_table.php)
update_nazov-tabulky (napr. 2017_11_13_000000_update_user_types_table.php)

Pozn: každé nové slovo v názve tabuľky pritom začína veľkým začiatočným písmenom.

3.2. Písanie kódu do migrácií

Pri písaní kódu v migračných súboroch je nutné vychádzať z nižšie popísaných štandardov a konvencií.

Laravel príkaz *make:migration* vytvorí základnú štruktúru súboru, s komentármi a funkciami pre prácu s databázou. V tomto prípade je teda nutné pôvodnú štruktúru zachovať a rozšíriť ju iba o potrebné funkcionalitu. To znamená pridať potrebné časti kódu do funkcií *up()* a *down(),* bez mazania komentárov, či iných častí základnej štruktúry.

Na začiatku súboru je nutné upraviť komentár obsahujúci informácie *user, date, time* a doplniť ich vlastnými údajmi.

```
/**
* Created by PhpStorm.
* User: meno_člena_tímu
* Date: 17.11.2018
* Time: 17:17
*/
```

4. Vytvárania a práca so „seeds“

Sekcia opisuje spôsob vytvárania tzv. seeds a definuje základné konvencie pre písanie kódu a pomenovávanie jednotlivých častí kódu.

V projektovom priečinku, časť *database/seeds,* sa nachádzajú všetky súbory, ktoré napĺňajú vytvorené tabuľky lokálnymi dátami pre vývojové prostredie.

4.1. Vytvorenie novej seed

Pri vytváraní nového súboru pre *seed* je nutné správne definovať názov súboru. Tento názov musí pozostávať s názvu tabuľky a kľúčového slova *Seeder* s príponou .php.

nazov-tabulkySeeder.php (napr. UserTypesSeeder.php)

Pozn: každé nové slovo v názve tabuľky pritom začína veľkým začiatočným písmenom.

4.2. Písanie kódu do seed

Pri písaní kódu v *seed* súboroch je nutné vychádzať z nižšie popísaných štandardov a konvencií.

Na začiatku súboru je nutné vytvoriť komentár obsahujúci informácie *user, date, time* a doplniť ich vlastnými údajmi, a to v ďalšom riadku za *<?php.*

```
/**
* Created by PhpStorm.
* User: meno_člena_tímu
* Date: 17.11.2018
* Time: 17:17
*/
```

Následne je nutné definovať rozširujúce súbory. Po komentári sa necháva jeden prázdny riadok.

```
use Carbon\Carbon;
use Illuminate\Database\Seeder;
```

Nasleduje samotná trieda *nazov-tabuľkySeeder*, ktorá rozširuje základnú triedu *Seeder*. Opäť je medzi predchádzajúcou časťou kódu a definovaním triedy nutné nechať jeden prázdny riadok.

```
class TestsTableSeeder extends Seeder
{
...
}
```

Do tela triedy sa s oddeleným začiatkom (tabulátor) píšu jednotlivé funkcie seed súboru. Základnou funkciou je funkcia *run(),* ktorá napĺňa jednotlivé stĺpce tabuľky preddefinovanými dátami. V tejto funkcií je nutné postupovať pri vytváraní *insert* podľa nižšie uvedeného príkladu.

```
public function run()
{
DB::table('tests')->insert([
'name' => "Skusobny test-uvodny",
'units_id' => 1,
'created_at' => Carbon::now()->format('Y-m-d H:i:s'),
'updated_at' => Carbon::now()->format('Y-m-d H:i:s'),
]);

DB::table('tests')->insert([
'name' => "Skusobny test-kvadraticke nerovnice",
'units_id' => 2,
'created_at' => Carbon::now()->format('Y-m-d H:i:s'),
'updated_at' => Carbon::now()->format('Y-m-d H:i:s'),
]);
```

```
DB::table('tests')->insert([
'name' => "Skusobny test-baroko",
'units_id' => 3,
'created_at' => Carbon::now()->format('Y-m-d H:i:s'),
'updated_at' => Carbon::now()->format('Y-m-d H:i:s'),
]);

}
```