SELECT \* FROM users:

<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\DB;

class UserController extends Controller

{

/\*\*

\* Show a list of all of the application's users.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

$users = DB::table('users')->get();

return view('user.index', ['users' => $users]);

}

}

SELECT name FROM users:

use Illuminate\Support\Facades\DB;

$users = DB::table('users')->get();

foreach ($users as $user) {

echo $user->name;

}

SELECT email FROM users WHERE name = ‘John‘ LIMIT 1:

$user = DB::table('users')->where('name', 'John')->first();

return $user->email;

detto:

$email = DB::table('users')->where('name', 'John')->value('email');

SELECT \* FROM users WHERE id = 3:

$user = DB::table('users')->find(3);

SELECT title FROM users

use Illuminate\Support\Facades\DB;

$titles = DB::table('users')->pluck('title');

foreach ($titles as $title) {

echo $title;

}

SELECT \* FROM users ORDER BY id (spracúva výsledky po 100):

use Illuminate\Support\Facades\DB;

DB::table('users')->orderBy('id')->chunk(100, function ($users) {

foreach ($users as $user) {

//

}

});

SELECT \* FROM users ORDER BY id LIMIT 100:

DB::table('users')->orderBy('id')->chunk(100, function ($users) {

// Process...

return false;

});

UPDATE users SET active = true WHERE ID = ‘user\_id’

DB::table('users')->where('active', false)

->chunkById(100, function ($users) {

foreach ($users as $user) {

DB::table('users')

->where('id', $user->id)

->update(['active' => true]);

}

});

SELECT COUNT(\*) FROM users; SELECT MAX(price) FROM orders

use Illuminate\Support\Facades\DB;

$users = DB::table('users')->count();

$price = DB::table('orders')->max('price');

SELECT AVG(price) FROM orders WHERE finalized = 1

$price = DB::table('orders')

->where('finalized', 1)

->avg('price');

Pokiaľ existuje/neexistuje aspoň jeden taký rekord

if (DB::table('orders')->where('finalized', 1)->exists()) {

// ...

}

if (DB::table('orders')->where('finalized', 1)->doesntExist()) {

// ...

}

SELECT name, email as user\_email FROM users

use Illuminate\Support\Facades\DB;

$users = DB::table('users')

->select('name', 'email as user\_email')

->get();

SELECT DISTINCT \* FROM users

$users = DB::table('users')->distinct()->get();

Pridať stĺpec:

$query = DB::table('users')->select('name');

$users = $query->addSelect('age')->get();

Vkladá surový string

$users = DB::table('users')

->select(DB::raw('count(\*) as user\_count, status'))

->where('status', '<>', 1)

->groupBy('status')

->get();

Detto:

$orders = DB::table('orders')

->selectRaw('price \* ? as price\_with\_tax', [1.0825])

->get();

Surový WHERE:

$orders = DB::table('orders')

->whereRaw('price > IF(state = "TX", ?, 100)', [200])

->get();

Surový HAVING:

$orders = DB::table('orders')

->select('department', DB::raw('SUM(price) as total\_sales'))

->groupBy('department')

->havingRaw('SUM(price) > ?', [2500])

->get();

Surové ORDER BY:

$orders = DB::table('orders')

->orderByRaw('updated\_at - created\_at DESC')

->get();

Surový GROUP BY

$orders = DB::table('orders')

->select('city', 'state')

->groupByRaw('city, state')

->get();

SELECT \*, contacts.phone, orders.price FROM USERS INNER JOIN contacts on users.id = contasts.user\_id INNER JOIN orders ON users.id = orders.user\_id

use Illuminate\Support\Facades\DB;

$users = DB::table('users')

->join('contacts', 'users.id', '=', 'contacts.user\_id')

->join('orders', 'users.id', '=', 'orders.user\_id')

->select('users.\*', 'contacts.phone', 'orders.price')

->get();

SELECT \* FROM users LEFT JOIN posts ON users.id = posts.user\_id

SELECT \* FROM users RIGHT JOIN posts ON users.id = posts.user\_id

$users = DB::table('users')

->leftJoin('posts', 'users.id', '=', 'posts.user\_id')

->get();

$users = DB::table('users')

->rightJoin('posts', 'users.id', '=', 'posts.user\_id')

->get();

SELECT \* FROM sizes CROSS JOIN colors

$sizes = DB::table('sizes')

->crossJoin('colors')

->get();

SELECT \* FROM users JOIN contacts ON users.id = contacts.user\_id OR ON ...

DB::table('users')

->join('contacts', function ($join) {

$join->on('users.id', '=', 'contacts.user\_id')->orOn(/\* ... \*/);

})

->get();

SELECT \* FROM users JOIN contacts ON users.id = contacts.user\_id WHERE contacts.user\_id > 5

DB::table('users')

->join('contacts', function ($join) {

$join->on('users.id', '=', 'contacts.user\_id')

->where('contacts.user\_id', '>', 5);

})

->get();

SELECT \* FROM users JOIN (SELECT user\_id, MAX(created\_at) AS last\_post\_created\_at WHERE is\_published = true GROUP BY user\_id) AS latest\_posts ON users.id = latest\_posts.user\_id

$latestPosts = DB::table('posts')

->select('user\_id', DB::raw('MAX(created\_at) as last\_post\_created\_at'))

->where('is\_published', true)

->groupBy('user\_id');

$users = DB::table('users')

->joinSub($latestPosts, 'latest\_posts', function ($join) {

$join->on('users.id', '=', 'latest\_posts.user\_id');

})->get();

SELECT \* FROM users WHERE last \_name = NULL UNION SELECT \* FROM users WHERE first\_name = NULL

use Illuminate\Support\Facades\DB;

$first = DB::table('users')

->whereNull('first\_name');

$users = DB::table('users')

->whereNull('last\_name')

->union($first)

->get();

SELECT \* FROM users WHERE votes = 100 AND age > 35

$users = DB::table('users')

->where('votes', '=', 100)

->where('age', '>', 35)

->get();

SELECT \* FROM users WHERE votes >= 100

SELECT \* FROM users WHERE votes <> 100

SELECT \* FROM users WHERE name like T%

$users = DB::table('users')

->where('votes', '>=', 100)

->get();

$users = DB::table('users')

->where('votes', '<>', 100)

->get();

$users = DB::table('users')

->where('name', 'like', 'T%')

->get();

SELECT \* FROM users WHERE status = 1 AND subscribed <> 1

$users = DB::table('users')->where([

['status', '=', '1'],

['subscribed', '<>', '1'],

])->get();

SELECT \* FROM users WHERE status = 1 OR name = John

$users = DB::table('users')

->where('votes', '>', 100)

->orWhere('name', 'John')

->get();

SELECT \* FROM users WHERE votes > 100 OR (name = Abigail AND votes > 50)

$users = DB::table('users')

->where('votes', '>', 100)

->orWhere(function($query) {

$query->where('name', 'Abigail')

->where('votes', '>', 50);

})

->get();

SELECT \* FROM products WHERE NOT (clearance = true OR price < 10)

$products = DB::table('products')

->whereNot(function ($query) {

$query->where('clearance', true)

->orWhere('price', '<', 10);

})

->get();

K nasledujúcim príkladom existujú aj ekvivalenty orwhere...

SELECT \* FROM users WHERE votes >= 1 AND votes <= 100

$users = DB::table('users')

->whereBetween('votes', [1, 100])

->get();

SELECT \* FROM users WHERE votes < 1 AND votes > 100

$users = DB::table('users')

->whereNotBetween('votes', [1, 100])

->get();

SELECT \* FROM patients WHERE weight >= minimum\_allowed\_weight AND weight <= maximum\_allowed\_weight

$patients = DB::table('patients')

->whereBetweenColumns('weight', ['minimum\_allowed\_weight', 'maximum\_allowed\_weight'])

->get();

SELECT \* FROM patients WHERE weight < minimum\_allowed\_weight AND weight > maximum\_allowed\_weight

$patients = DB::table('patients')

->whereNotBetweenColumns('weight', ['minimum\_allowed\_weight', 'maximum\_allowed\_weight'])

->get();

SELECT \* FROM users WHERE id IN (1, 2, 3)

$users = DB::table('users')

->whereIn('id', [1, 2, 3])

->get();

SELECT \* FROM users WHERE id NOT IN (1, 2, 3)

$users = DB::table('users')

->whereNotIn('id', [1, 2, 3])

->get();

SELECT \* FROM users WHERE updated\_at = NULL

$users = DB::table('users')

->whereNull('updated\_at')

->get();

SELECT \* FROM users WHERE NOT updated\_at = NULL

$users = DB::table('users')

->whereNotNull('updated\_at')

->get();

Dátum:

$users = DB::table('users')

->whereDate('created\_at', '2016-12-31')

->get();

SELECT \* FROM users WHERE MONTH(created\_at) = 12

$users = DB::table('users')

->whereMonth('created\_at', '12')

->get();

SELECT \* FROM users WHERE DAY(created\_at) = 31

$users = DB::table('users')

->whereDay('created\_at', '31')

->get();

SELECT \* FROM users WHERE YEAR(created\_at) = 2016

$users = DB::table('users')

->whereYear('created\_at', '2016')

->get();

Čas:

$users = DB::table('users')

->whereTime('created\_at', '=', '11:20:45')

->get();

SELECT \* FROM users WHERE first\_name = last\_name

$users = DB::table('users')

->whereColumn('first\_name', 'last\_name')

->get();

Select \* from users where exists ( Select 1 from orders where orders.user\_id = users.id)

$users = DB::table('users')

->whereExists(function ($query) {

$query->select(DB::raw(1))

->from('orders')

->whereColumn('orders.user\_id', 'users.id');

})

->get();

SELECT \* FROM users WHERE (SELECT type FROM membership WHERE membership.user\_id = users.id ORDER BY membership.start\_date DESC LIMIT 1) = ‚Pro’

use App\Models\User;

$users = User::where(function ($query) {

$query->select('type')

->from('membership')

->whereColumn('membership.user\_id', 'users.id')

->orderByDesc('membership.start\_date')

->limit(1);

}, 'Pro')->get();

SELECT \* FROM incomes WHERE amount < (SELECT AVG(i.amount) FROM incomes as i)

use App\Models\Income;

$incomes = Income::where('amount', '<', function ($query) {

$query->selectRaw('avg(i.amount)')->from('incomes as i');

})->get();

SELECT \* FROM users WHERE bio = ‚web developer’

$users = DB::table('users')

->whereFullText('bio', 'web developer')

->get()

SELECT \* FROM users ORDER BY name DESC

$users = DB::table('users')

->orderBy('name', 'desc')

->get();

SELECT \* FROM users ORDER BY name DESC OREDER BY email ASC

$users = DB::table('users')

->orderBy('name', 'desc')

->orderBy('email', 'asc')

->get();

SELECT \* FROM users ORDER BY created\_at DESC LIMIT 1 (latest/oldest)

$user = DB::table('users')

->latest()

->first();

Utriedenie v náhodnom poradí

$randomUser = DB::table('users')

->inRandomOrder()

->first();

Reorder odstraňuje všetky predošlé order,...

$query = DB::table('users')->orderBy('name');

$unorderedUsers = $query->reorder()->get();

Prípadne utriedi nanovo

$query = DB::table('users')->orderBy('name');

$usersOrderedByEmail = $query->reorder('email', 'desc')->get();

SELECT \* FROM users GROUP BY account\_id HAVING account\_id > 100

$users = DB::table('users')

->groupBy('account\_id')

->having('account\_id', '>', 100)

->get();

SELECT COUNT(id) AS number\_of\_orders, customer\_id FROM orders GROUP BY customer\_id HAVING number\_of\_orders >= 5 AND number\_of\_orders <= 15

$report = DB::table('orders')

->selectRaw('count(id) as number\_of\_orders, customer\_id')

->groupBy('customer\_id')

->havingBetween('number\_of\_orders', [5, 15])

->get();

SELECT \* FROM users GROUP BY firest\_name, status HAVING account\_id > 100

$users = DB::table('users')

->groupBy('first\_name', 'status')

->having('account\_id', '>', 100)

->get();

SELECT \* FROM users OFFSET 10 LIMIT 5

$users = DB::table('users')->skip(10)->take(5)->get();

Detto:

$users = DB::table('users')

->offset(10)

->limit(5)

->get();

INSERT INTO users (email, votes) VALUES (‚[kayla@example.com](mailto:kayla@example.com)’, 0)

DB::table('users')->insert([

'email' => 'kayla@example.com',

'votes' => 0

]);

Možno urobiť viac INSERT-ov naraz

DB::table('users')->insert([

['email' => 'picard@example.com', 'votes' => 0],

['email' => 'janeway@example.com', 'votes' => 0],

]);

insertOrIgnore ignoruje chyby, (napr. duplikáty id)

INSERT INTO pruned\_users (id, name, email, email\_verified\_at) VALUES (SELECT id, name, email, email\_verified\_at WHERE updated\_at <= MONTH(NOW()))

DB::table('pruned\_users')->insertUsing([

'id', 'name', 'email', 'email\_verified\_at'

], DB::table('users')->select(

'id', 'name', 'email', 'email\_verified\_at'

)->where('updated\_at', '<=', now()->subMonth()));

Pokiaľ má tabuľka auto-increment id, možno použiť:

$id = DB::table('users')->insertGetId(

['email' => 'john@example.com', 'votes' => 0]

);

UPDATE ak existuje (departure a destination), INSERT ak neexistuje (price)

DB::table('flights')->upsert(

[

['departure' => 'Oakland', 'destination' => 'San Diego', 'price' => 99],

['departure' => 'Chicago', 'destination' => 'New York', 'price' => 150]

],

['departure', 'destination'],

['price']

);

Detto:

DB::table('users')

->updateOrInsert(

['email' => 'john@example.com', 'name' => 'John'],

['votes' => '2']

);

UPDATE users SET votes = 1 WHERE id = 1

$affected = DB::table('users')

->where('id', 1)

->update(['votes' => 1]);

Zvýši/zníži číselnú hodnotu o 1 (alebo viac)

DB::table('users')->increment('votes');

DB::table('users')->increment('votes', 5);

DB::table('users')->decrement('votes');

DB::table('users')->decrement('votes', 5);

Zvýši hodnotu votes o 1, WHERE name = ‘John’

DB::table('users')->increment('votes', 1, ['name' => 'John']);

DELETE FROM users

DELETE FROM users WHERE votes > 100

$deleted = DB::table('users')->delete();

$deleted = DB::table('users')->where('votes', '>', 100)->delete();

DELETE FROM users, resetuje auto-increment na 0

DB::table('users')->truncate();

Uzamkne upravovanie tabuľky, kým nie je SELECT dokončený

DB::table('users')

->where('votes', '>', 100)

->sharedLock()

->get();

Uzamkne upravovanie tabuľky, kým nie je SELECT dokončený, aj pred inými sharedLock

DB::table('users')

->where('votes', '>', 100)

->lockForUpdate()

->get();

Príklad pre výpis tabuľky

<table class="table table-w">  
 <thead>  
 <tr>  
 <th scope="col">#id</th>  
 <th scope="col">Názov Firmy</th>  
 <th scope="col">Skratka</th>  
 <th scope="col">Adresa</th>  
 <th scope="col">Mesto</th>  
 <th scope="col"></th>  
 <th scope="col"></th>  
 </tr>  
 </thead>  
 <tbody>  
 <?php  
 $companies = DB::*table*('Firma')  
 ->get();  
 foreach ($companies as $comp) {  
 echo "<tr>";  
 echo "<th scope='row'>$comp->idFirma</th>";  
 echo "<td>$comp->Názov\_firmy</td>";  
 echo "<td>$comp->Skratka</td>";  
 echo "<td>$comp->Adresa</td>";  
 echo "<td>$comp->Mesto\_idMesto</td>";  
 echo "</tr>";  
 };  
 ?>  
 </tbody>  
</table>

Príklad pre INSERT do tabuľky:

<form action="stuContReportSave" method="POST">  
 @csrf  
<table class="table table-w">  
 <thead>  
 <tr>  
 <th scope="col">Pozícia</th>  
 <th scope="col">Názov Firmy</th>  
 <th scope="col">Typ Zmluvy</th>  
 <th scope="col">Dátum Začiatku</th>  
 <th scope="col">Dátum Konca</th>  
 <th scope="col">Kontaktná Osoba</th>  
 <th scope="col">Aktuálny stav</th>  
 <th scope="col">Predmety</th>  
 <th scope="col">Pracovník FPVaI</th>  
 <th scope="col"></th>  
 <th scope="col"></th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td><input type="text" name="Pozicia" size="8"></td>  
 <td><input type="text" name="Nazov\_firmy" size="8"></td>  
 <td><input type="text" name="Typ\_Zmluvy" size="8"></td>  
 <td><input type="text" name="Datum\_Zaciatku" size="8"></td>  
 <td><input type="text" name="Datum\_Konca" size="8"></td>  
 <td><input type="text" name="Kontaktna\_Osoba" size="8"></td>  
 <td><input type="text" name="Aktualny\_stav" size="8"></td>  
 <td><input type="text" name="Predmety" size="8"></td>  
 <td><input type="text" name="Pracovnik\_FPVaI" size="8"></td>  
 <td>  
 <div class="d-flex mb-3">  
 <button type="submit" class="btn btn-primary">Add</button>  
 </div>  
 </td>  
 <td>  
 <div class="d-flex mb-3">  
 <a class="btn btn-danger" href="stuContReportList">Cancel</a>  
 </div>  
 </td>  
 </tr>  
 </tbody>  
</table>  
</form>

Keď bude form obsahovať celú tabuľku, bude to fungovať.

Ešte tam nemám dropdowny, ale to sa ešte dorobí.

Toto je routa:

Route::*post*('/stuContReportSave', [TestController::class, 'stuContReportSave']);

A toto je funkcia v Controllery:

public function stuContReportSave (Request $req){  
 DB::*table*('Prax')->insert([  
 'Pozicia' => $req->Pozicia,  
 'Student\_idPouzivatel' => Auth::*user*()->id,  
 'Datum\_start' => $req->Datum\_Zaciatku,  
 'Datum\_end' => $req->Datum\_Konca,  
 'Firma\_idFirma' => $req->Nazov\_firmy,  
 'Kontaktna\_osoba\_idPouzivatel' => $req->Kontaktna\_Osoba,  
 'Aktuálny\_stav' => $req->Aktualny\_stav,  
 'Predmety\_idPredmety' => $req->Predmety,  
 'Pracovnik\_FPVaI\_idPouzivatel' => $req->Pracovnik\_FPVaI,  
 'Zmluva\_idZmluva' => $req->Typ\_Zmluvy  
 ]);  
 return redirect('stuContReportList');  
}

Request funguje tento:

use Illuminate\Http\Request;