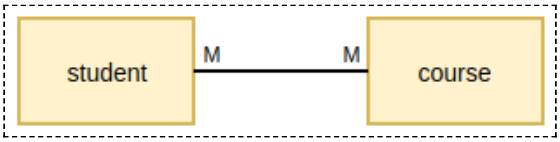
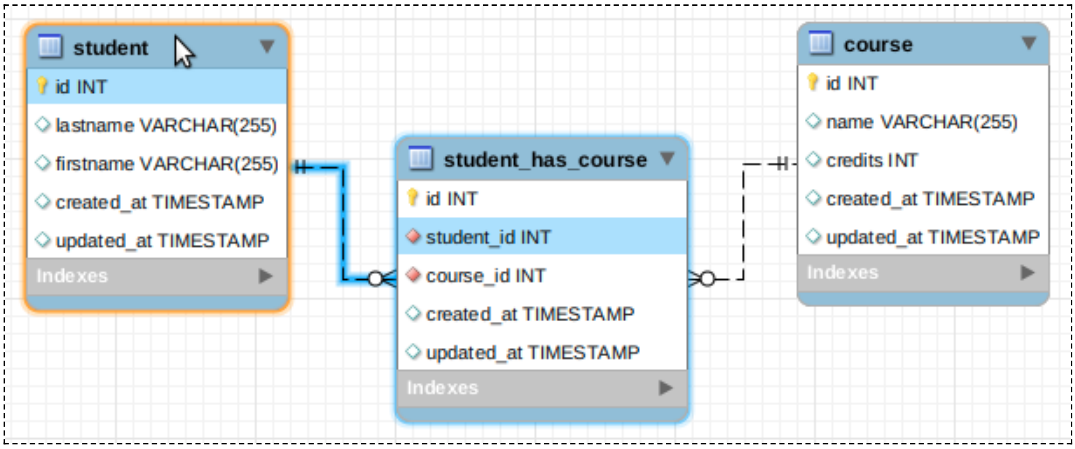
Laravel::DB - yhteydet, liitoskyselyt ja Query Builder

> Tässä luvussa esitetään, kuinka Laravelillä perustetaan ja käsitellään keskenään erilaisia lukumääräsuhteita sisältäviä relaatiotietokannan tauluja

Relaatiotietokannan rakenne on seurausta käsitemallinnuksessa määritellyistä käsitteiden välisistä yhteyksistä ja niiden lukumääräsuhteista. Opiskelija voi esimerkiksi suorittaa monta kurssia ja toisaalta saman kurssin voi suorittaa monta opiskelijaa. Tällöin ilmaistaan, että käsitteiden *opiskelija* ja *kurssi* lukumääräsuhde on monen suhde moneen.



Relaatiotietokannoissa monen-moneen-suhde on purettava. Näin saadaan *väli-* eli *junction-* eli *associative-* eli *pivot-taulu* `student_has_course`, joka sisältää viiteavaimiin merkittynä tiedon siitä kuka on suorittanut minkäkin kurssin. Oheisessa MySQL WorkBench -luonnoksessa esitetään tässä luvussa luonnin ja käsittelyn kohteena olevat taulut suhteineen ja kenttineen.



Luodaan tietokanta yhteyksineen

35.01 Luodaan migraatiotiedostojen pohjat

```
php artisan make:migration create_student_table
php artisan make:migration create_course_table
php artisan make:migration create_student_has_course_table
```

Avataan luodut migraatiotiedostot ja muokataan ne muotoihin, jotka johtavat alussa esitettyyn tietokannan rakenteeseen

database/migrations/2019_10_25_195728_create_student_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateStudentTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('student', function (Blueprint $table) {
            $table->increments('id');
            $table->string('lastname');
            $table->string('firstname');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('student');
    }
}
```

database/migrations/2019_10_25_195735_create_course_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCourseTable extends Migration
```

```
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('course', function (Blueprint $table) {

            $table->increments('id');
            $table->string('name');
            $table->integer('credits');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('course');
    }
}
```

database/migrations/2019_10_25_195740_create_student_has_course_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateStudentHasCourseTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('student_has_course', function (Blueprint $table
            $table->increments('id');
            $table->integer('student_id')->unsigned();
            $table->foreign('student_id')->references('id')->on('studen
```

```
20              $table->integer('course_id')->unsigned();
21              $table->foreign('course_id')->references('id')->on('course'
22              $table->timestamps();
23          });
24      }
25
26      /**
27       * Reverse the migrations.
28       *
29       * @return void
30       */
31      public function down()
32      {
33          Schema::dropIfExists('student_has_course');
34      }
35  }
```

Huomaa tässä viimeisessä tiedostossa viiteavainten määritykset vyörytyssääntöineen riveillä 18-21.

35.03 Aja migraatiot tietokantaan

Tallenna edellä muokkaamasi tiedostot ja aja migraatiot tietokantaan

```
php artisan migrate
```

35.04 Tutki luomiasi tauluja

Ota yhteys Laravel-sovelluksesi käyttämään tietokantaan

```
mysql -u root -psala laraveldb
```

Tutki luomiesi taulujen rakennetta ja asetuksia ja vertaa luotuja kenttiä Laravelin migraatiotiedostoissa määriteltyihin. Huomaa esim. että metodi `$table->timestamps()` luo kentät `created_at` ja `updated_at`.

```
mysql> show tables;
+---------------------+
| Tables_in_laraveldb |
+---------------------+
| course              |
| customers           |
| failed_jobs         |
| migrations          |
```

```
| password_resets     |
| student             |
| student_has_course  |
| users               |
+---------------------+
8 rows in set (0.00 sec)

mysql> desc student;
+------------+------------------+------+-----+---------+----------
| Field      | Type             | Null | Key | Default | Extra
+------------+------------------+------+-----+---------+----------
| id         | int(10) unsigned | NO   | PRI | NULL    | auto_incre
| lastname   | varchar(255)     | NO   |     | NULL    |
| firstname  | varchar(255)     | NO   |     | NULL    |
| created_at | timestamp        | YES  |     | NULL    |
| updated_at | timestamp        | YES  |     | NULL    |
+------------+------------------+------+-----+---------+----------
5 rows in set (0.00 sec)

mysql> desc course;
+------------+------------------+------+-----+---------+----------
| Field      | Type             | Null | Key | Default | Extra
+------------+------------------+------+-----+---------+----------
| id         | int(10) unsigned | NO   | PRI | NULL    | auto_incre
| name       | varchar(255)     | NO   |     | NULL    |
| credits    | int(11)          | NO   |     | NULL    |
| created_at | timestamp        | YES  |     | NULL    |
| updated_at | timestamp        | YES  |     | NULL    |
+------------+------------------+------+-----+---------+----------
5 rows in set (0.00 sec)

mysql> desc student_has_course;
+------------+------------------+------+-----+---------+----------
| Field      | Type             | Null | Key | Default | Extra
+------------+------------------+------+-----+---------+----------
| id         | int(10) unsigned | NO   | PRI | NULL    | auto_incre
| student_id | int(10) unsigned | NO   | MUL | NULL    |
| course_id  | int(10) unsigned | NO   | MUL | NULL    |
| created_at | timestamp        | YES  |     | NULL    |
| updated_at | timestamp        | YES  |     | NULL    |
+------------+------------------+------+-----+---------+----------
5 rows in set (0.00 sec)
```

Tarkistetaan vielä luodut viiteavaimet

```
mysql> show create table student_has_course;
...
| student_has_course | CREATE TABLE `student_has_course` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `student_id` int(10) unsigned NOT NULL,
  `course_id` int(10) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `student_has_course_student_id_foreign` (`student_id`),
  KEY `student_has_course_course_id_foreign` (`course_id`),
  CONSTRAINT `student_has_course_course_id_foreign` FOREIGN KEY (`c
  CONSTRAINT `student_has_course_student_id_foreign` FOREIGN KEY (`
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
...
```

35.05 Lisätään tietokantaan esimerkkidataa

LIsätään tietokantaan esimerkkidataa eli populoidaan tietokantaa esim. mysql-komentokehotteesta

Lisätään muutama opiskelija

```
INSERT INTO student (id, lastname, firstname, created_at, updated_at)
(1, 'Guru','Ken', (select now()), (select now())),
(2, 'Alainen','Kim', (select now()), (select now())),
(3, 'Leppä','Kerttu', (select now()), (select now()));
```

Lisätään muutama kurssi

```
INSERT INTO course (id, name, credits, created_at, updated_at)  VALUES
(1, 'Ohjelmointi', 5, (select now()), (select now())),
(2, 'Tietokannat',4, (select now()), (select now())),
(3, 'Ruotsi',3, (select now()), (select now()));
```

Lisätään muutamia kurssisuorituksia

```
INSERT INTO student_has_course (student_id, course_id, created_at, upda
(1, 1, (select now()), (select now())),
(1, 2, (select now()), (select now())),
(1, 3, (select now()), (select now())),
(2, 1, (select now()), (select now())),
(2, 2, (select now()), (select now())),
(3, 1, (select now()), (select now()));
```

Tarkastetaan tulokset

```
mysql> select * from student;
+----+----------+-----------+---------------------+---------------
| id | lastname | firstname | created_at          | updated_at
+----+----------+-----------+---------------------+---------------
|  1 | Guru     | Ken       | 2019-10-25 20:37:50 | 2019-10-25 20:3
|  2 | Alainen  | Kim       | 2019-10-25 20:37:50 | 2019-10-25 20:3
|  3 | Leppä    | Kerttu    | 2019-10-25 20:37:50 | 2019-10-25 20:3
+----+----------+-----------+---------------------+---------------
3 rows in set (0.01 sec)

mysql> select * from course;
+----+-------------+---------+---------------------+---------------
| id | name        | credits | created_at          | updated_at
+----+-------------+---------+---------------------+---------------
|  1 | Ohjelmointi |       5 | 2019-10-25 20:38:49 | 2019-10-25 20:
|  2 | Tietokannat |       4 | 2019-10-25 20:38:49 | 2019-10-25 20:
```

```
|  3 | Ruotsi      |       3 | 2019-10-25 20:38:49 | 2019-10-25 20:
+----+-------------+---------+---------------------+--------------
3 rows in set (0.00 sec)

mysql> select * from student_has_course;
+----+------------+-----------+---------------------+--------------
| id | student_id | course_id | created_at          | updated_at
+----+------------+-----------+---------------------+--------------
|  1 |          1 |         1 | 2019-10-25 20:39:38 | 2019-10-25 20
|  2 |          1 |         2 | 2019-10-25 20:39:38 | 2019-10-25 20
|  3 |          1 |         3 | 2019-10-25 20:39:38 | 2019-10-25 20
|  4 |          2 |         1 | 2019-10-25 20:39:38 | 2019-10-25 20
|  5 |          2 |         2 | 2019-10-25 20:39:38 | 2019-10-25 20
|  6 |          3 |         1 | 2019-10-25 20:39:38 | 2019-10-25 20
+----+------------+-----------+---------------------+--------------
6 rows in set (0.00 sec)
```

testataan vielä liitoksen toimivuus

```sql
SELECT student.id, student.lastname, student.firstname,
       course.name
FROM student
INNER JOIN student_has_course
  ON student.id = student_has_course.student_id
INNER JOIN course
  ON course.id = student_has_course.course_id;
```

tuloksena tulisi olla

```
+----+----------+-----------+-------------+
| id | lastname | firstname | name        |
+----+----------+-----------+-------------+
|  1 | Guru     | Ken       | Ohjelmointi |
|  1 | Guru     | Ken       | Tietokannat |
|  1 | Guru     | Ken       | Ruotsi      |
|  2 | Alainen  | Kim       | Ohjelmointi |
|  2 | Alainen  | Kim       | Tietokannat |
|  3 | Leppä    | Kerttu    | Ohjelmointi |
+----+----------+-----------+-------------+
6 rows in set (0.01 sec
```

Luodaan mallit ja määritellään moni-moneen-yhteydet

- https://laravel.com/docs/5.7/eloquent-relationships#many-to-many
- https://laraveldaily.com/pivot-tables-and-many-to-many-relationships/
- https://hackernoon.com/eloquent-relationships-cheat-sheet-5155498c209

35.06 Luodaan mallit yhteyksineen

Luo tiedostot `app/Student.php` ja `app/Course.php`

```
php artisan make:model Student
php artisan make:model Course
```

ja editoi mallitiedostot seuraavasti

app/Student.php

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Student extends Model
8  {
9      protected $table = 'student';
10
11     public function courses()
12     {
13         return $this->belongsToMany('App\Course', 'student_has_course',
14     }
15 }
```

- Rivillä 9 määritellään, että Student-malli käsittelee student-nimistä taulua (Laravelin oletus olisi käsitellä students-nimistä taulua)
- Student-mallin yhteyteen on riveille 11-14 kirjoitettu uusi metodi `courses()`, jonka avulla voidaan palauttaa opiskelijan suorittamat kurssit

app/Course.php

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Course extends Model
8  {
9      protected $table = 'course';
10
11     public function students()
12     {
13         return $this->belongsToMany('App\Student', 'student_has_course'
14     }
15 }
```

routes/web.php

```
...
Route::get('studentjson', 'StudentController@studentjson');
Route::get('coursejson', 'StudentController@coursejson');

Route::get('student', 'StudentController@studentlist');
Route::get('course', 'StudentController@courselist');
```

35.08 Luodaan StudentController opiskelijoiden ja kurssien listaamiseksi

```
php artisan make:controller StudentController
```

ja editoidaan kontrolleria seuraavasti. Huomaa riveillä 5 ja 6 mallien mukaanotot

app/Http/Controllers/StudentController

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Student;
6  use App\Course;
7
8
9  use Illuminate\Http\Request;
10
11 class StudentController extends Controller
12 {
13     public function studentjson() {
14         $students = Student::all();
15     return $students;
16     }
17
18     public function coursejson() {
19         $courses = Course::all();
20     return $courses;
21     }
22 }
```

ja testaan palautukset:

Osoitteesta http://192.168.1.126/~testi/projekti01/public/studentjson

pitäisi tulostua

```
[
{
"id": 1,
"lastname": "Guru",
"firstname": "Ken",
"created_at": "2019-10-25 20:37:50",
"updated_at": "2019-10-25 20:37:50"
},
{
"id": 2,
"lastname": "Alainen",
"firstname": "Kim",
"created_at": "2019-10-25 20:37:50",
"updated_at": "2019-10-25 20:37:50"
},
{
"id": 3,
"lastname": "Leppä",
"firstname": "Kerttu",
"created_at": "2019-10-25 20:37:50",
"updated_at": "2019-10-25 20:37:50"
}
]
```


Ja osoitteesta http://192.168.1.126/~testi/projekti01/public/coursejson

pitäisi tulostua


```
[
{
"id": 1,
"name": "Ohjelmointi",
"credits": 5,
"created_at": "2019-10-25 20:38:49",
"updated_at": "2019-10-25 20:38:49"
},
{
"id": 2,
"name": "Tietokannat",
"credits": 4,
"created_at": "2019-10-25 20:38:49",
"updated_at": "2019-10-25 20:38:49"
},
{
"id": 3,
"name": "Ruotsi",
"credits": 3,
"created_at": "2019-10-25 20:38:49",
"updated_at": "2019-10-25 20:38:49"
```

```
}
]
```

Katsetta kestävä käyttöliittymä Bootstrap-kirjastolla

Twitter Bootstrap -kirjaston käyttöönotolla saavutetaan astetta ammattimaisempi ja kaikille päätelaitteille automaattisesti mukautava käyttöliittymä kopiomalla yksi css- ja js-tiedosto sovelluksen käyttöön. Halutessasi voit tutkia verkosta tarkemminkin Bootstrapilla tavoiteltavia asioita, mutta se ei ole oleellista tämän opintojakson opiskelun jatkon kannalta.

35.09 Bootstrap-kirjastojen käyttöönotto näkymissä

Kopioidaan aluksi seuraavat Bootstrap-tiedostot oheisiin sijainteihin Laravelin käyttöön

- app-studetronic.css --> public/css/app-studetronic.css
- app-studetronic.js --> public/js/app-studetronic.js

Luodaan sitten tarvittaessa kansio `resources/views/layouts` ja luodaan Bootstrap-kirjastoon perustuva *masterlayout* seuraavasti:

resources/views/layouts/app-studetronic.blade.php

```
<!DOCTYPE html>
<html lang="{{ app()->getLocale() }}">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1"

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>StudeTronic</title>

    <!-- Styles -->
    <link href="{{ asset('css/app-studetronic.css') }}" rel="stylesheet"

</head>

<body>
    <div id="app">
        <nav class="navbar navbar-default navbar-static-top">
            <div class="container">
                <div class="navbar-header">

                    <!-- Collapsed Hamburger -->
                    <button type="button" class="navbar-toggle collapse
                        <span class="sr-only">Toggle Navigation</span>
```

```
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>

                    <!-- Branding Image -->
                    <a class="navbar-brand" href="{{ url('/student') }}"
                    StudeTronic
                    </a>
                </div>

                <div class="collapse navbar-collapse" id="app-navbar-co
                    <!-- Left Side Of Navbar -->
                    <ul class="nav navbar-nav">
                         
                    </ul>

            <ul class="nav navbar-nav">
                <li><a href="{{ url('/student') }}">Listaa opiskelijat</
              </ul>
                </div>
            </div>
        </nav>
        @yield('content')
    </div>

    <!-- Scripts -->
<script src="{{ asset('js/app-studetronic.js') }}"></script>

</body>
</html>
```

Tämän jälkeen luodaan masterlayout-tiedostoa käyttävä näkymä opiskelijoiden listaamiseksi

resources/views/studetronic/students.blade.php

```
@extends('layouts.app-studetronic')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Kaikki opiskelijat</div>


<div class="panel-body">

<table class="table table-striped">
```

```
  <thead>
    <tr>
      <th scope="col">id</th>
      <th scope="col">Sukunimi</th>
      <th scope="col">Etunimi</th>
      <th scope="col">Luontipvm</th>
    </tr>
  </thead>
  <tbody>

@foreach ($students as $student)

    <tr class="bg-success">
      <th scope="row">{{ $student->id }}</th>
      <td>{{ $student->lastname }}</td>
      <td>{{ $student->firstname }}</td>

      <td>{{ $student->created_at }}</td>
    </tr>

@endforeach

  </tbody>
</table>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

35.10 StudentControlleriin studentlist()-metodi

StudentControlleriin tulee sitten lisätä metodi opiskelijoiden hakemiseksi tietokannasta:

```
public function studentlist() {
        $students = Student::all();
return view('studetronic/students')->with('students', $students);
}
```

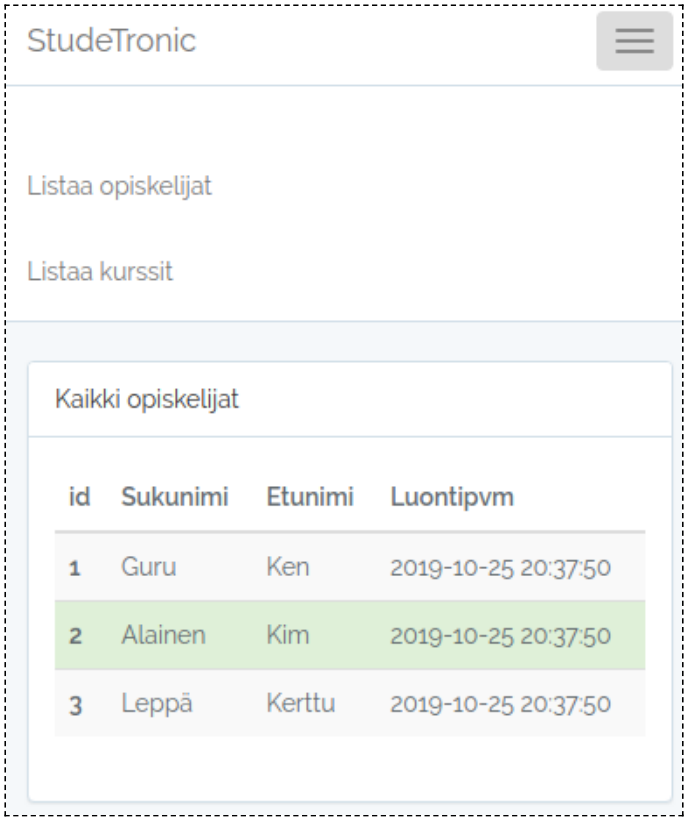
35.11 Testaus

Reittitiedostoon aiemmin lisätyn reitin `Route::get('student', 'StudentController@studentlist');` avulla opiskelijoiden listaaminen osoitteessa

http://192.168.1.126/~testi/projekti01/public/student

pitäisi tulostaa seuraavan näkymän

Mobiilinäkymäkin toimii valikko avattuna suoraan



Datan hakeminen kahdesta taulusta

Haetaan dataa kahdesta taulusta `student` ja `student_has_course` tietyn opiskelijan suoritettujen kurssien listaamiseksi.

35.12 Mallin muuttaminen

Esimerkin vuoksi pivot-taulusta `student_has_course` haetaan myös aikaleimakentät. Näin *Student*-mallin `courses()`-metodiin lisätään noudettavaksi nämäkin kentät seuraavasti

app/Student.php

```
...
public function courses()
{
      return $this->belongsToMany('App\Course', 'student_has_course', '
            ->withPivot('created_at', 'updated_at');
}
...
```

35.13 Lisätään tarvittava reitti

routes/web.php

```
Route::get('studentincourses', 'StudentController@studentincourses
```

35.14 Lisätään tarvittava metodi

Lisätään uusi metodi StudentControlleriin

app/Http/Controllers/StudentController.php

```
public function studentincourses() {
        $student = Student::find(1); // Haetaan opiskelijan id=1 tiedot
        // return $student->courses; // Kokeile ensin, että tämä palautu
        return view('studentincourses')->with('studentincourses', $stude
    }
```

35.15 Lisätään uusi näkymä

resources/views/studetronic/studentincourses.blade.php

```
@extends('layouts.app-studetronic')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Opiskelijan id=1 kurssit</div


<div class="panel-body">

```

```
<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col">id</th>
      <th scope="col">Nimi</th>
      <th scope="col">ECTS-pisteet</th>
      <th scope="col">Luontipvm</th>
      <th scope="col">Pivot</th>
    </tr>
  </thead>
  <tbody>

@foreach ($studentincourses as $studentincourse)

    <tr class="bg-success">
      <th scope="row">{{ $studentincourse->id }}</th>
      <td>{{ $studentincourse->name }}</td>
      <td>{{ $studentincourse->credits }}</td>
      <td>{{ $studentincourse->created_at }}</td>
      <td>
      <!-- Tämä foreach on täällä vain esimerkin vuoksi, jotta
           nähdään pivotin sisältö. vaihda if(0) -> if(1)
           nähdäksesi tämän sisällön-->
      @if(0)
        @foreach ($studentincourse->pivot as $key => $value)
            {{ $key }}   : {{ $value }}
        @endforeach
      @endif
      </td>
    </tr>

@endforeach

  </tbody>
</table>


                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

35.16 Lisätään linkki navigointivalikkoon

Lisätään edellä luodulle toiminnallisuudelle linkki navigointivalikkoon eli tiedostoon `resources/views/layouts/app-studetronic.blade.php`

resources/views/layouts/app-studetronic.blade.php

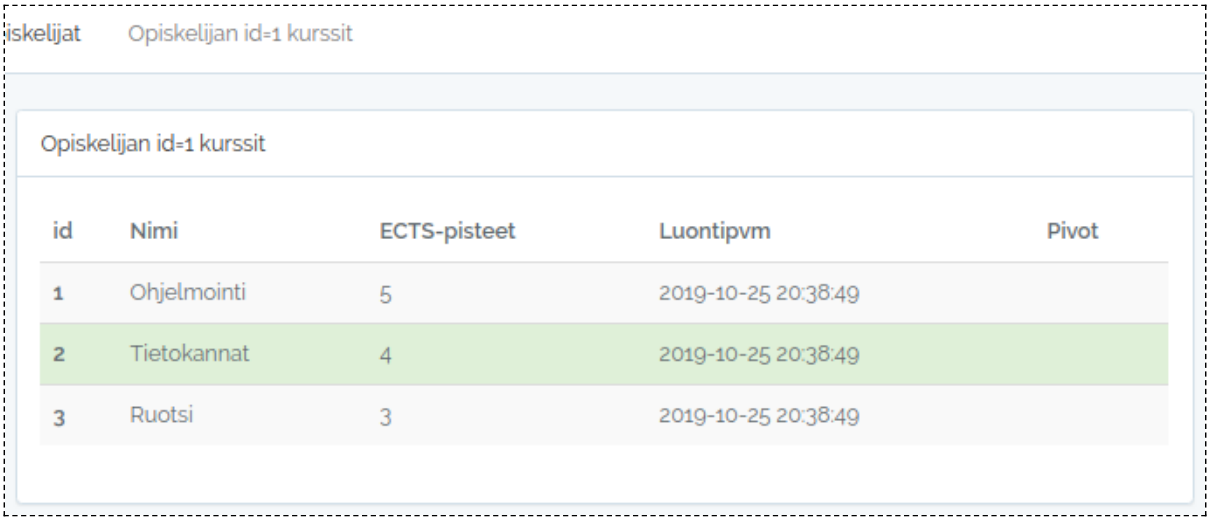
```
...
        <ul class="nav navbar-nav">
            <li><a href="{{ url('/studentincourses') }}">Opiskelija
        </ul>
...
```

35.17 Testaus

Testataan osoitteessa

http://192.168.1.126/~testi/projekti01/public/studentincourses

jolloin selaimeen pitäisi tulostua seuraava näkymä



PIvot-sarake on tyhjä koska näkymätiedostossa `studentincourses.blade.php` rivillä 36 on `@if(0)`. Kokeile arvolla `@if(1)`.

> Huomataan, että kiva on, mutta haluamme helpommin ja rikkaammin dataa useasta taulusta

Database: query builder

Laravelin tietokantakyselyjen rakentajalla query builder voi luoda lähes kaikkiin tietokannan käsittelyn tarpeisiin (CRUD) tietokantakyselyjä, jotka ilman eri toimenpiteitä ovat suojattuna SQL-injektion mahdollisuudelta.

Esimerkiksi SQL-kysely

```
SELECT  id, lastname AS Sukunimi, firstname AS Etunimi
FROM student
WHERE city = 'tampere'
ORDER BY id DESC
```

voidaan esittää query builderillä muodossa

```
$students = DB::table('student')
->select('id',
    DB::raw("lastname AS Sukunimi"),
    DB::raw("firstname AS Etunimi")
    )
->where('city','=','tampere')
->orderBy('id', 'desc')
->get();
```

Laajemmin query builderin tarjoamaan toiminnallisuuteen voit tutustua dokumentaation avulla

- Database: Query Builder

35.18 Datan hakeminen kolmesta taulusta - SQL

Haetaan opiskelijan id=1 kaikki kurssisuoritukset kolmesta taulusta kahdella liitoksella. Muokataan sarakeotsikkoja ja lajitellaan. SQL-tasolla se olisi näin:

```
SELECT student.id, student.lastname as Sukunimi, student.firstname as E
       course.name as Kurssi, course.credits, student_has_course.create
FROM student
INNER JOIN student_has_course
  ON student.id = student_has_course.student_id
INNER JOIN course
  ON course.id = student_has_course.course_id
WHERE student.id = 1
ORDER BY course.credits DESC;
```

ja tulos olisi tällöin

```
+----+----------+---------+-------------+---------+----------------
| id | Sukunimi | Etunimi | Kurssi      | credits | created_at
+----+----------+---------+-------------+---------+----------------
|  1 | Guru     | Ken     | Ohjelmointi |       5 | 2019-10-25 20:3
|  1 | Guru     | Ken     | Tietokannat |       4 | 2019-10-25 20:3
|  1 | Guru     | Ken     | Ruotsi      |       3 | 2019-10-25 20:3
+----+----------+---------+-------------+---------+----------------
```

Tehdään sama Laravelin **DB**-olion `select`-metodilla. Lisätään reitti

routes/web.php

```
Route::get('studentcredits', 'StudentController@studentcredits');
```

Muokataan StudenControlleria lisäämällä kontrollerin alkuun `use`-lause sekä uusi `studentcredits()`-metodi seuraavasti

app/Http/Controllers/StudentController.php

```
1  //...
2  use Illuminate\Support\Facades\DB;
3  //...
4
5      public function studentcredits()
6      {
7        $studentcredits = DB::table('student')
8        ->select('student.id',
9          DB::raw("student.lastname AS Sukunimi"),
10         DB::raw("student.firstname AS Etunimi"),
11         DB::raw("course.name as Kurssi"),
12         'course.credits',
13         'student_has_course.created_at'
14         )
15       ->join('student_has_course', 'student.id', '=', 'student_has_cours
16       ->join('course', 'course.id', '=', 'student_has_course.course_id'
17       ->where('student.id','=','1')
18       ->orderBy('course.credits', 'desc')
19       ->get();
20
21     // return $studentcredits;
22     return view('studetronic/studentcredits')->with('studentcredits', $
23     }
```

ja luodaan toiminnolle vielä näkymä

resources/views/studetronic/studentcredits.blade.php

```
1  @extends('layouts.app-studetronic')
2
3  @section('content')
4  <div class="container">
5      <div class="row">
6          <div class="col-md-8 col-md-offset-2">
```

```
            <div class="panel panel-default">
                <div class="panel-heading">Opintosuoriteote opiskelija :


<div class="panel-body">

<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col">id</th>
      <th scope="col">Sukunimi</th>
      <th scope="col">Etunimi</th>
      <th scope="col">Kurssi</th>
      <th scope="col">ECTS-pisteet</th>
      <th scope="col">Luontipvm</th>
    </tr>
  </thead>
  <tbody>

@foreach ($studentcredits as $studentcredit)

    <tr class="bg-success">
      <th scope="row">{{ $studentcredit->id }}</th>
      <td>{{ $studentcredit->Sukunimi }}</td>
      <td>{{ $studentcredit->Etunimi }}</td>
      <td>{{ $studentcredit->Kurssi }}</td>
      <td>{{ $studentcredit->credits }}</td>
      <td>{{ $studentcredit->created_at }}</td>
    </tr>

@endforeach

  </tbody>
</table>


                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

Lisätään edellä luodulle toiminnallisuudelle linkki navigointivalikkoontiedostoon

resources/views/layouts/app-studetronic.blade.php

```
...
<ul class="nav navbar-nav">
```

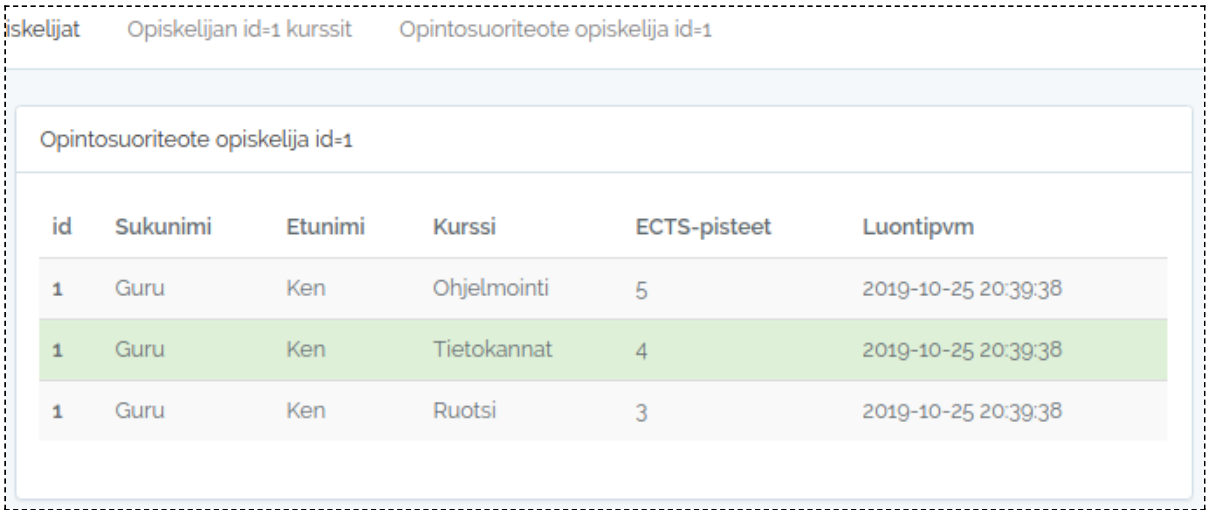
```
    <li><a href="{{ url('/studentcredits') }}">Opintosuoriteote opiskeli
</ul>
...
```

35.20 Testaus

Testataan osoitteessa

http://192.168.1.126/~testi/projekti01/public/studentcredits

jolloin selaimeen pitäisi tulostua seuraava näkymä



> Huomataan tämän olevan tosi kiva! Mutta haluamme vielä valita halutun opiskelijan opiskelijalistasta

---

Klikatun opiskelijan opintosuoriteote

Toteutetaan toiminnallisuus, jossa opiskelijalistasta hyperlinkkiä klikkaamalla saadaan näkyville opiskelijan opintosuoriteote

35.21 Näkymä

Muokataan opiskelijalistanäkymää hieman ja lisätään siihen hyperlinkki seuraavasti

resources/views/studetronic/students.blade.php

```
...

<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col">id</th>
```

```
      <th scope="col">Sukunimi Etunimi</th>
      <th scope="col">Luontipvm</th>
    </tr>
  </thead>
  <tbody>

@foreach ($students as $student)

    <tr class="bg-success">
      <th scope="row">{{ $student->id }}</th>
            <td><a href="{{ url('studentcredits') }}?id={{ $student->id
      <td>{{ $student->created_at }}</td>
    </tr>

@endforeach

  </tbody>
</table>
...
```

35.22 Muokataan kontrolleria

muokkaamalla studentcredits()-metodia siten, että pyynnön (request) mukana tuleva `id` vaikuttaa kyselyn **where**-osassa tulokseen

app/Http/Controllers/StudentController.php

```
public function studentcredits()
{

    // Luetaan linkin mukana tullut käyttäjän id
    $id = request('id');

    $studentcredits = DB::table('student')
    ->select('student.id',
        DB::raw("student.lastname AS Sukunimi"),
        DB::raw("student.firstname AS Etunimi"),
        DB::raw("course.name as Kurssi"),
        'course.credits',
        'student_has_course.created_at'
        )
    ->join('student_has_course', 'student.id', '=', 'student_has_course
    ->join('course', 'course.id', '=', 'student_has_course.course_id')
    ->where('student.id','=',"$id")
    ->orderBy('course.credits', 'desc')
    ->get();

// return $studentcredits;
```

```
22  return view('studetronic/studentcredits')->with('studentcredits', $stud
23  }
```

35.23 Navigointivalikoiden ja näkymien hienosäätäminen

Päävalikosta kannattaa sitten poistaa nämä aiemmin luodut tarpeettomat linkit

```
1  <ul class="nav navbar-nav">
2          <li><a href="{{ url('/studentincourses') }}">Opiskelijan id=1 ku
3  </ul>
4
5  <ul class="nav navbar-nav">
6      <li><a href="{{ url('/studentcredits') }}">Opintosuoriteote opiskel
7  </ul>
```

Harkintasi mukaan voit poistaa myös kontrollerista tarpeettomaksi käyneen `studentincourses()`-metodin ja siihen liittyvän `studentincourses.blade.php`-näkymän ja niihin liittyvän reitin. Toisaalta niiden jääminen sovellukseen ei haittaa mitään.

`studentcredits.blade.php`-näkymään voit halutessasi muuttaa opiskeijan nimen näyttävän otsikon:

resources/views/studetronic/studentcredits.blade.php

```
1  ...
2  <div class="panel-heading">Opintosuoriteote {{ $studentcredits[0]->Sukun
3  ...
```

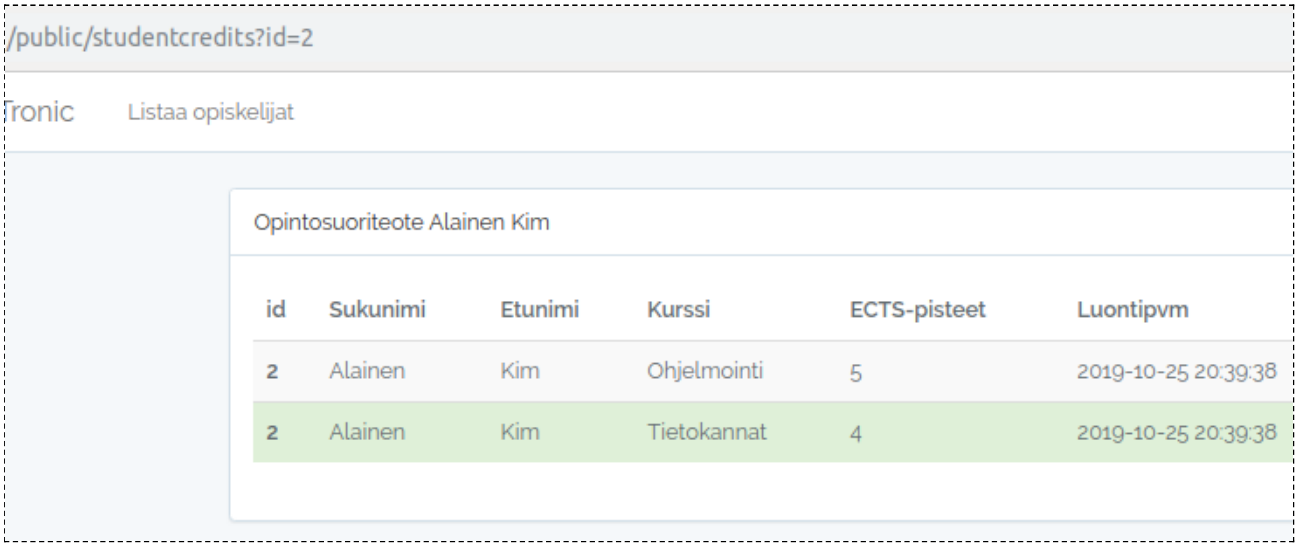
35.24 Testaus

Testataan osoitteessa

http://192.168.1.126/~testi/projekti01/public/student

jolloin selaimeen tulostuu seuraava näkymä

josta klikkaamalla Kim Alaista saadaan näkymä



Huomaa osoiteriviltä kuinka klikatun oiskelijan id välitetään

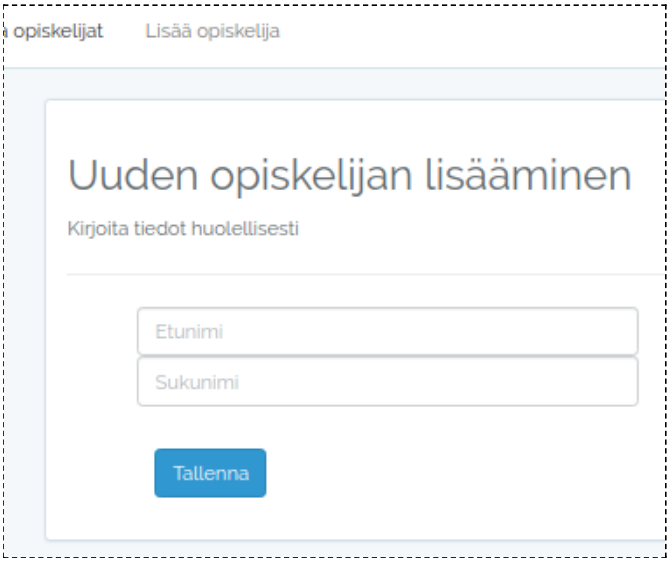
> Ai että voi ihanasti valita opiskelijalistasta halutun opiskelijan opintosuoritteen katsottavaksi. Tämä on liian hyvää ollakseen totta ja niin se onkin, koska vielä me haluamme ainakin kokeilla sen, kuinka uusi opiskelija lisätään lomakkeelta tietokantaan.

Opiskelijan lisääminen lomakkeelta tietokantaan

Toteutetaan toiminnallisuus, jossa tietokantaan voidaan lisätä HTML-lomakkeen avulla uusi opiskelija

35.25 Lisäyslomake linkkeineen

Luodaan aluksi uusi oheinen lisäyslomake



- `id`-, `created_at`-, `updated_at`-kentät täyttyvät oletusarvoilla, joten lomakkeessa kysytään ainoastaan suku- ja etunimeä.
- Lomakkeen kenttien sisäiset nimet ovat tietokannan taulun kenttien mukaiset lastname ja firstname

resources/views/studetronic/studentform.blade.php

```
@extends('layouts.app-studetronic')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-body">
        <h2 id="uploadhead">Uuden opiskelijan lisääminen</h2>
        <p>Kirjoita tiedot huolellisesti</p>
        <hr/>
            <form role="form" data-toggle="validator" method="POST" act
                <div class="form-group row">
                    <div class="row">
                        <div class="form-group col-md-6 col-md-offset-1
                        <input id="firstname" type="text" class="form-c
                        <input id="lastname" type="text" class="form-co
                        </div>
                    </div>
                </div>

                <div class="form-group row">
                    <div class="col-md-5 col-md-offset-1">
                        <button type="submit" class="btn btn-primary">
                            Tallenna
                        </button>
                    </div>
                </div>
```

```
29
30                  <input type="hidden" name="_token" value="{{ csrf_token
31              </form>
32
33                  </div>
34              </div>
35          </div>
36      </div>
37  </div>
38  @endsection
```

LIsää myös päävalikkoon linkki opiskelijan lisäämiseksi

```
1  ...
2  <ul class="nav navbar-nav">
3          <li><a href="{{ url('/studentform') }}">Lisää opiskelija</a></li
4  </ul>
5  ...
```

35.26 Vaadittavat reitit

Lisätään reitit lomakkeelle ja tallentamista varten

app/web.php

```
Route::get('/studentform', 'StudentController@studentform');
Route::post('/storestudent','StudentController@store');
```

35.27 Muutokset kontrolleriin

Muokataan kontrolleria lisäämällä kaksi uutta metodia

app/Http/Controllers/StudentController.php

```
1  ...
2      public function studentform() {
3            return view('studetronic/studentform');
4      }
5
6      public function store() {
7          Student::create(request()->all());
8          return redirect(url('/student'));
9      }
10 ...
```

35.28 Muutokset malliin

Salli mallissa nimikenttien massatäyttö

app/Student.php

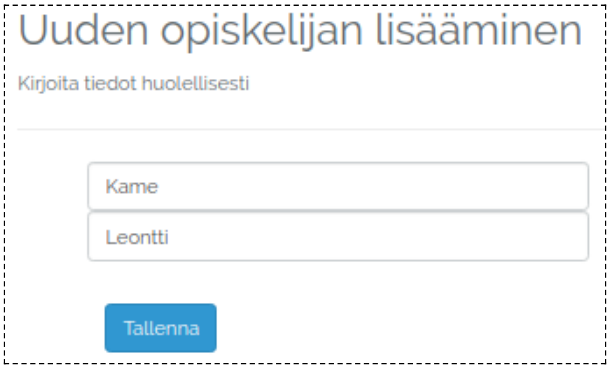
```
...
    protected $fillable = [
        'lastname',
        'firstname'
    ];
...
```

35.29 Testaus

Testataan osoitteessa

http://192.168.1.126/~testi/projekti01/public/studentform

täyttämällä lomakkeelle uuden opiskelijan tiedot



tallentamisen seurauksena ohjaudutaan automaattisesti opiskelilista-näkymään, jonka viimeisenä näkyvät uuden opiskelijan tiedot

Huomaat varmaan, että uuden *Kame Leontti* -opiskelijan opintosuoritusotteen näyttämisestä tulee ikävä virheilmoitus. Varo vaan, saatat joutua korjaamaan sen harjoitustehtävänä!

> Jos listasi näyttää opiskelijan lisäyksen jälkeen suurinpiirtein yllä olevalta, voit onnitella itseäsi ja hengähtää hetkeksi. Mutta. Vain hetkeksi. Koska sitten haluat vielä itsenäisesti lisätä tähän sovellukseen muutaman lisäominaisuuden tekemällä harjoitukset 8. Kreisiä!

```
© #AriRantala
```