***VSoft***

***Система за обучения***

*Изготвена от*

*Валентин Маринов Колев*

*Спец. Софтуерно Инженерство II курс*

*Ф.номер 1909010411*

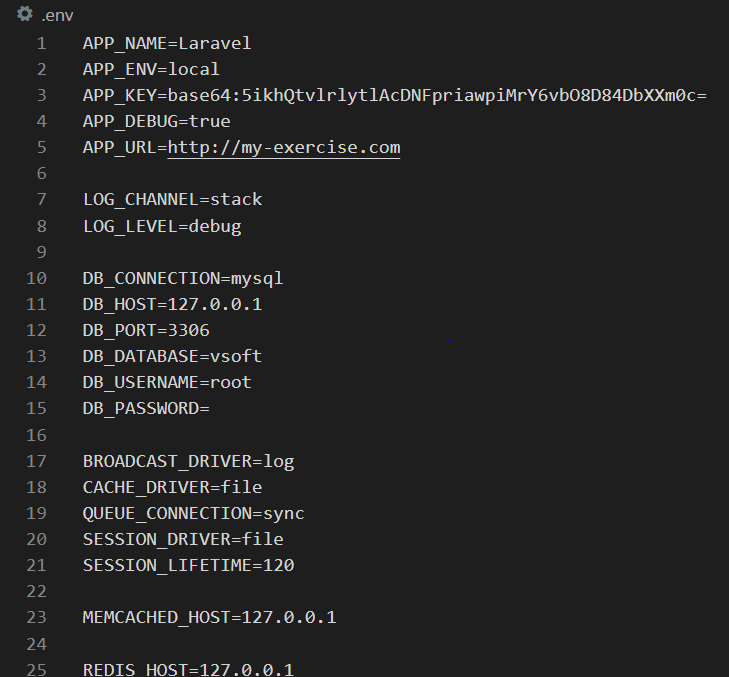
1. **Описание**

Проектът представлява система за обучения, в която може да се въвеждат данни , отговарящи за лектори, курсове, локации и организации, провеждащи съответните курсове.

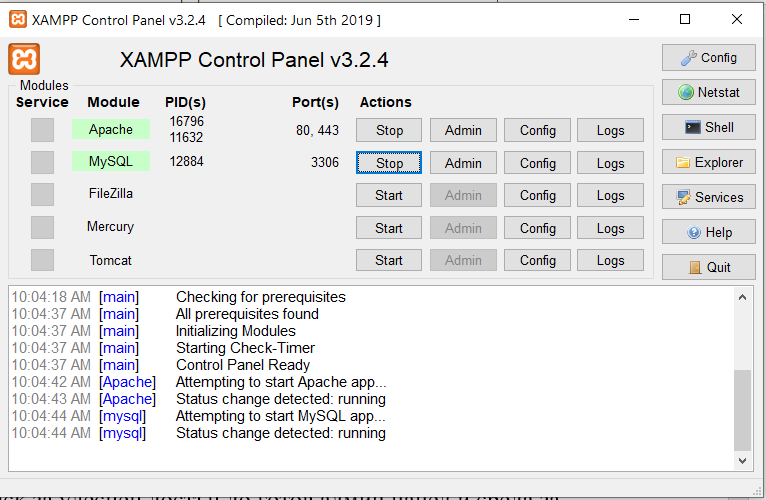
Създаден е , ползвайки функционалността на PHP Laravel и Backpack. Преди да искаме да работим с него или да го променим по наше желание, трябва да имаме инсталирани нужните изисквания за поддръжка на проекта. Toва са XAMPP , който ще служи за създаване и инстанциране на нашия виртуален сървър, върху който ще ползваме проекта, PHP Laravel работната рамка , Backpack за улеснен достъп до готов админ панел и среда за разработка ( за предпочитане PHP Storm / Visual Studio Code ).

Нещата трябва да са инсталиране в правилна подредба – първо среда за разработка, в която ще пишем нашия код, после XAMPP , на който ще настройм нашия virtual host, после ще създадем Laravel проект, и след това ще инсталираме допълнително върху нашия проект Backpack.

След като сме нагласили програмите и сме задали правилно виртуалните хостове и сме поставили проекта в папката ***htdocs*** на ***xampp***, трябва да се уверим , че правилно сме настроили ***.env***  файла в проекта да сочи към нашата база в phpMyAdmin и да сме сигурни ,че виртуалните хостове съвпадат.

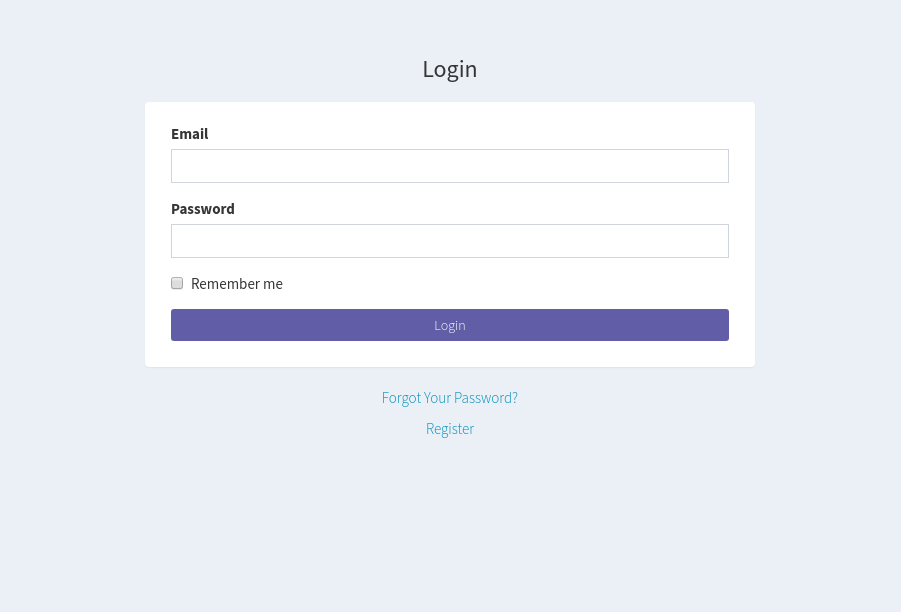


След като сме изпълнили тази стъпка , пускаме XAMPP и стартираме Apache и MySQL. При правилно нагласени настройки трябва да получим следния резултат:



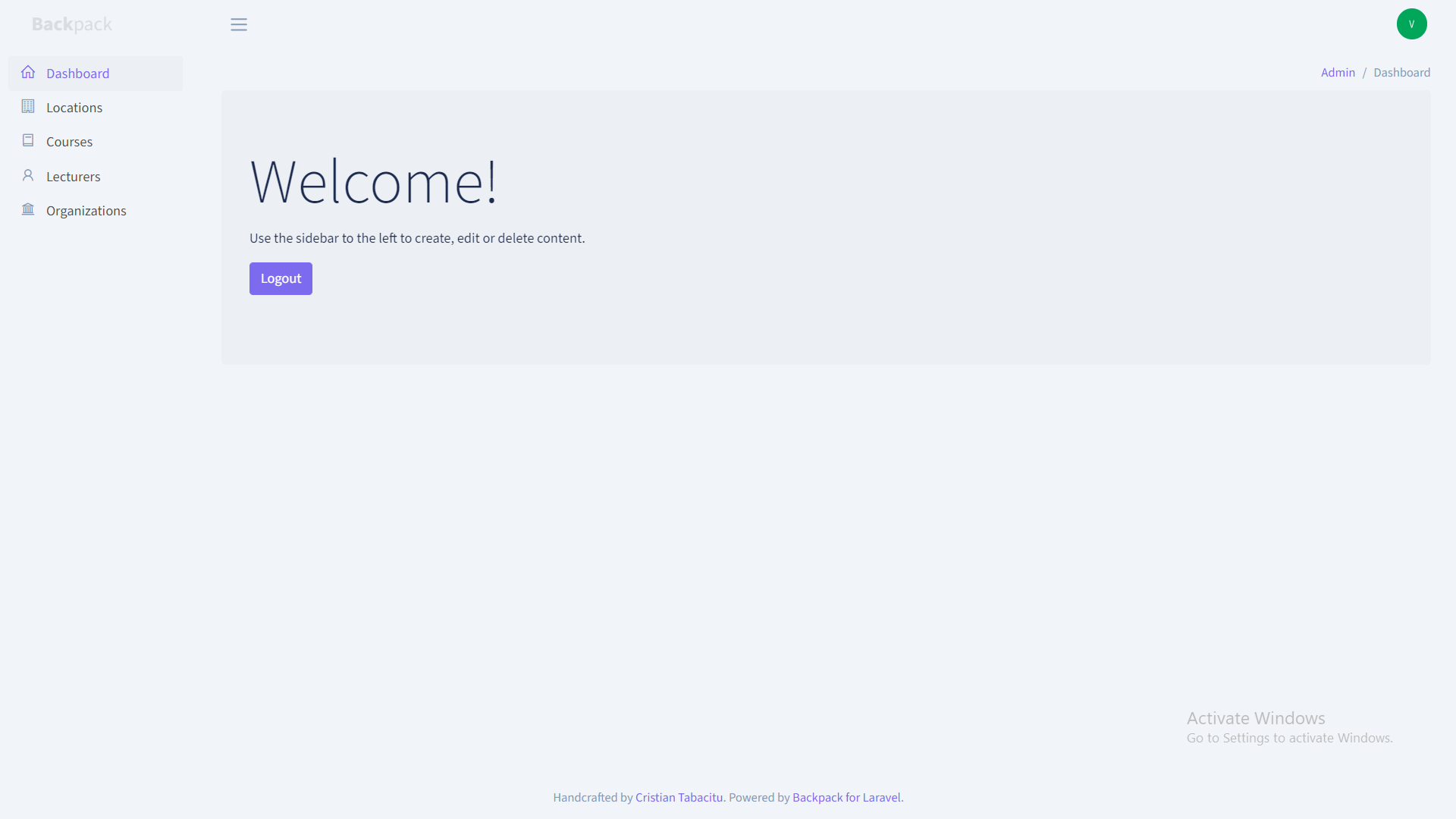
1. **Стартиране**

Направил съм линк-ът на достъпване през браузъра да бъде **my-exercise.com**. Ако тепърва започваме нашия проект, на този адрес ще се покаже изглед по подразбиране на Laravel , в който ще има кратко описание и линкове към самата документация. За да го сменим , трябва да имаме наш собствен изглед, който да смени този , по подразбиране.

Когато сме стартирали проекта , преминаваме към администраторския панел, където ще можем да регистрираме нашия потребител. Той се намира на съответния адрес: **my-exercise.com/admin.** Всеки , който е създал собтсвен url , може да го достъпи по аналогичен начин, като добави накрая наклонена черта и думичката admin. След като сме достъпили адресът за логин, получаваме следната формичка: 

Ако нямаме създаден профил, ще трябва да се регистрираме, ползвайки бутончето Register.

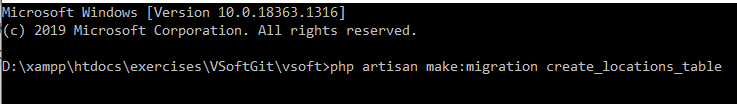
Преминаваме успешно регистрацията и вече се намираме в администраторския панел на Backpack:

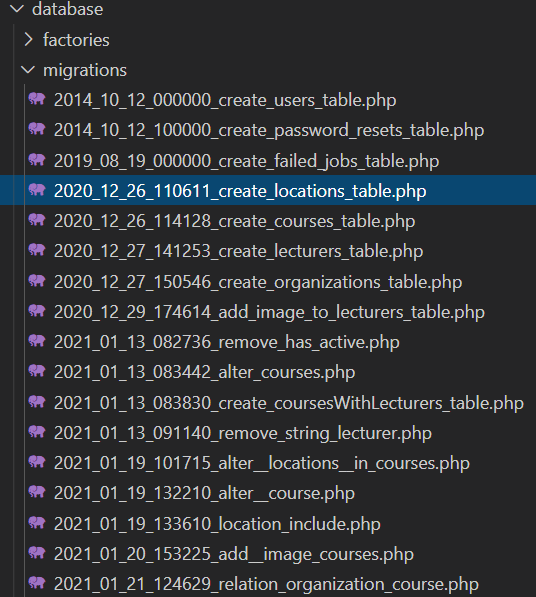


Това представлява завършения вид на един администраторски панел. В лявата част имаме меню, което съдържа различните CRUD секции на нашите модели. Там ние може да създаваме, редактираме, изтриваме и търсим записи от нашата база от данни.

1. **Backpack CRUD функционалност, миграции и достъпване на записи**

Всяка табличка със записи и модел към нея е направена по аналогичен начин. Първо зареждаме конзолата в директорията на нашия проект, след това правим миграция , която ще служи за създаване на нашия модел и CRUD операциите около него:



Всички миграции се намират на едно и също място – database>migrations: 

След като имаме вече създадена миграцията , трябва да опишем нейното тяло ( тоест какво трябва тя да изпълни при изпълняване). Избираме съответната миграция и имаме следния код:



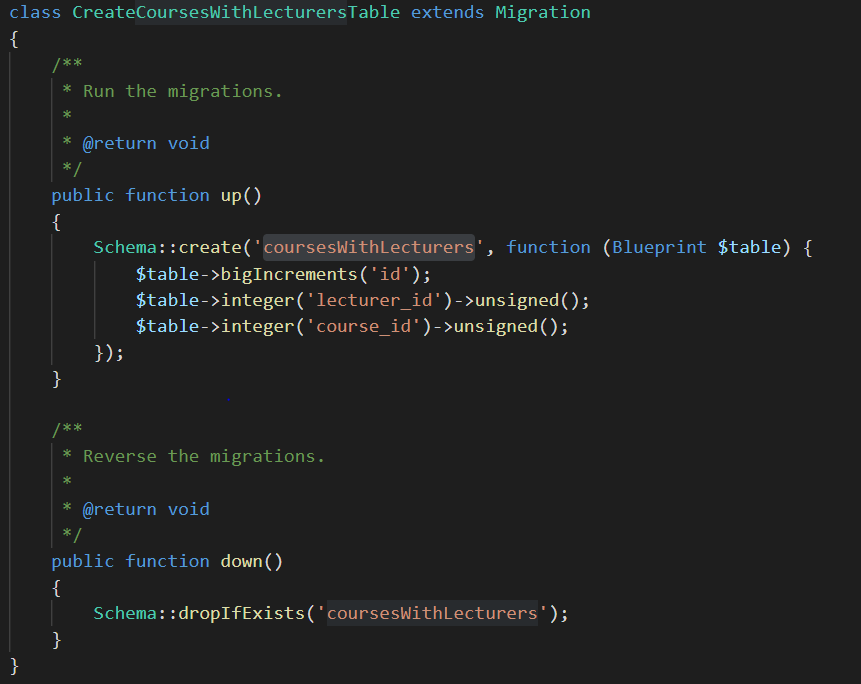
Във функцията function up() създаваме нашата таблица, която ще има полета – уникално ID , което се инкрементира автоматично, име от тип String , булева променлива и timestamps() метод , който ще отчита време на създаване и промяна. Трябва да имаме предвид ,че това е начален етап на нашите записи за локации и има допълнителни промени в следващите миграции ( като например са премахнати булевите стойности за проверка на активни курсове, добавени са връзки и външни ключове и още други допълни функционалности).

Създали сме вече и тялото на нашата миграция , остава само да я изпълним посредством командата *php artisan migrate.* След като е готова , ползваме вградената CRUD функционалност на Backpack и изпълняваме  *php artisan backpack:crud location* , като внимаваме за наименованието ( когато я създаваме е в множествено число, а когато ползваме backpack , достъпваме моделът , който е в единствено число.

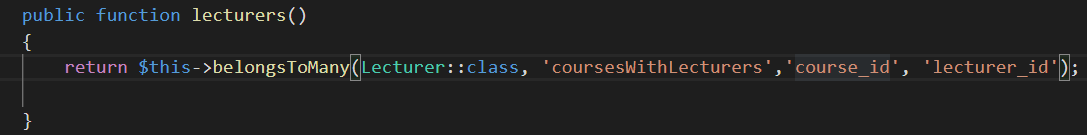
Следвайки тези стъпки, създаваме и останалите обекти в нашия проект.

1. **Връзки между отделните таблици**

Ползвам съм два основни начина за връзване на отделните модели. Единият е с междинна таблица (pivot table) между отделните курсове и лекторите , които ги водят ( които могат да бъдат повече от един ). Използвам съм миграция за създаване на съответната таблица:

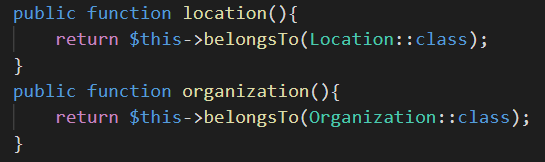


Тук освен ID колонка, имаме две колони , които сочат към двете отделни таблици , съдържащи записи за лекторите и за курсовете. Освен тук , трябва в отделните модели да си създадем функция, с която да посочим какъв тип връзка ще имат тези модели:



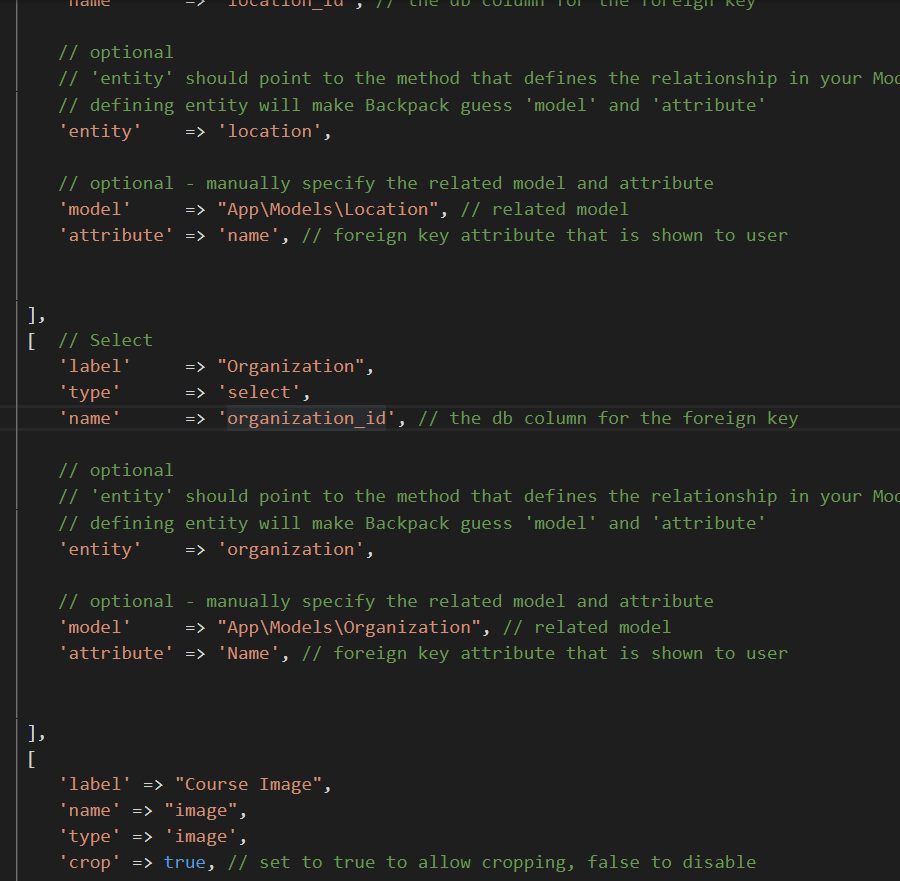
Функция в моделът Course.php ( отнасящ се за курсовете)

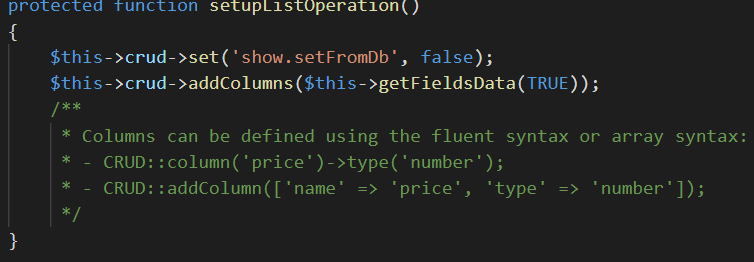
Другият начин е директно между моделите да се окаже връзката , без да се ползва междинна таблица:



Функции, отговарящи за връзката между курсовете и локации,организации

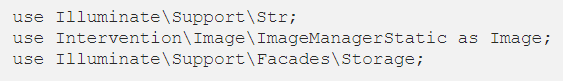
За да можем да ползваме тези връзки и допълните функционалности, като падащ списък, избиране от няколко записа и други екстри, в някой от контролерите има getFields() метод , който презаписва основния начин на изписване на записите от базата. В него можем да изберем по какъв начин да покажем нашие записи, да добавим опция за вкарване на изображение чрез файл, да избираме от вече съществуващи записи, които имат създадена връзка по между си, да пренапишем наименованията на отделните части на модела, като име , кога стартира даден курс и други допълнителни опции.



За да можем да ползваме този метод , трябва допълнително да посочим на нашия проект , че искаме да не ползваме листването на обекти по подразбиране в метода setupListOperation():

1. **Вмъкване на изображение**

Вмъкването на изображение е една малко по-сложна операция. Тя изисква да имаме отделна колонка в нашата таблица, която да сочи към източника на това изображение , който ще представлява пътят му в файловата директория на проекта. За да ползваме изображения , трябва да имаме вградени следните готови неща в нашия модел, върху който ще работим:



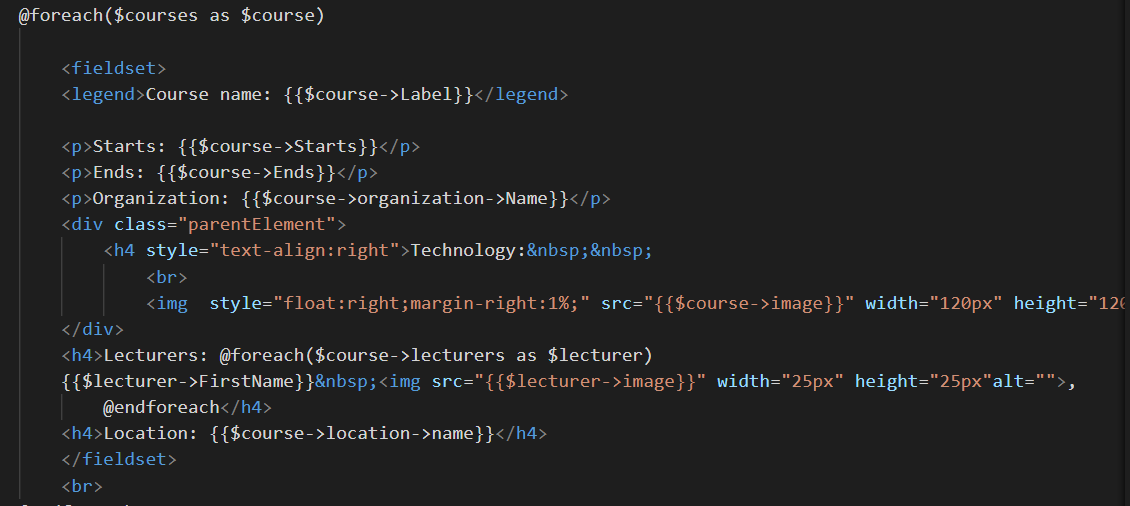
Без тях няма да можем да използваме възможностите за създаване на изображение. Трябва да допълним моделът с две допълнителни функции – *boot*  и *setImageAttribute,* които ще служат за изтриване и запазване на изображение. Не трябва да забравяме да си допълним в контролера визуализация за CRUD операции над това изображение и да имаме публичен достъп над нашата файлова директория, за да може да се достъпват изображенията, което се осъществява посредством командата *php artisan storage:link.*

1. **Изгледи**

Вече сме създали всички нужни записи на нашия проект, но още нямаме тяхна визуализация извън администраторския панел. За улеснена работа, ще ползваме готов template , от който ще си изберем и подредим нужните лицеви компоненти за частта , която виждат обикновените потребители. След като сме нагласили файловете , създаваме в папката ***index***  два изгледа – един за начална страница и един за филтрирани записи посредством зададен критерий. Създаваме и допълнителен IndexController , в който имаме нашите две функции – index() , която ще служи за началната ни страница и search() , която ще служи за филтриране на нашите записи , ползвайки резултатът от зададените заявки:



В изгледите , ползвайки цикъл , ще изваждаме всички нужни записи от базата:



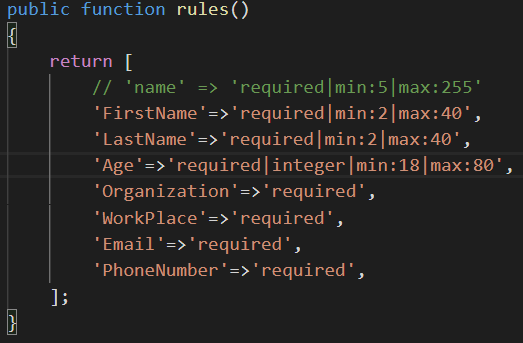
1. **Валидации**

Валидациите са важна част от всяко приложение. Ако потребителят. При неправилно въведени данни, ще получим SQL грешка , която ще прекрати нашият админ панел и би могла да повлияе негативно на User Experience-а.

За да се избегнат тези неща, ползваме вградена валидация и правила, които ако не бъдат изпълнени , заявката няма да се зареди и няма да стигнем до SQL грешка:



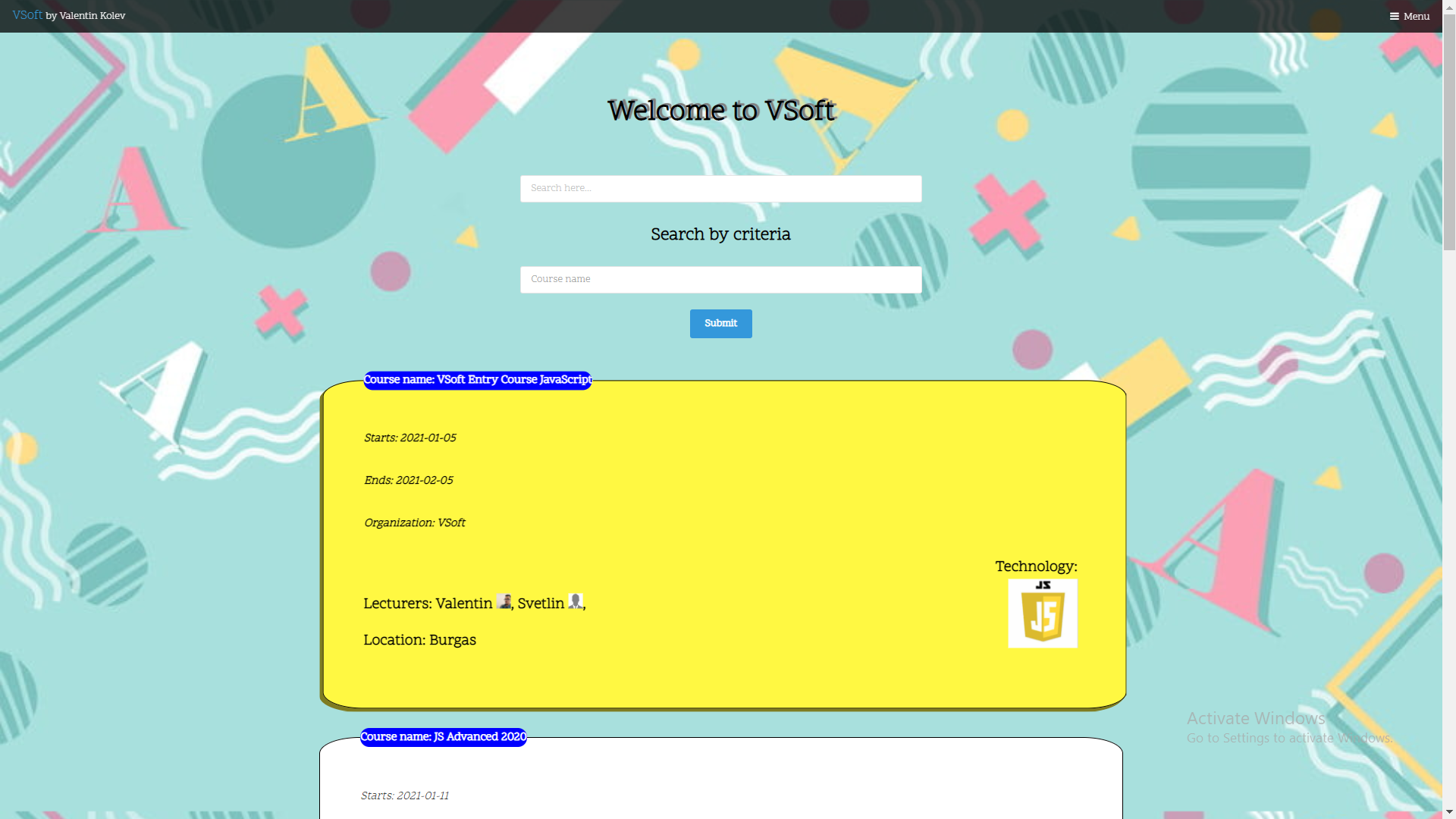
Осъществяването на тези валидации се намира в отделните Request файлове, отговорни за изпълнение на заявките за всеки наш запис от базата. В тях можем да зададем желаната от нас валидация, както и да напишем свои собствени съобщения за грешки, които ще презапишат тези по подразбиране:



Валидация на лектори

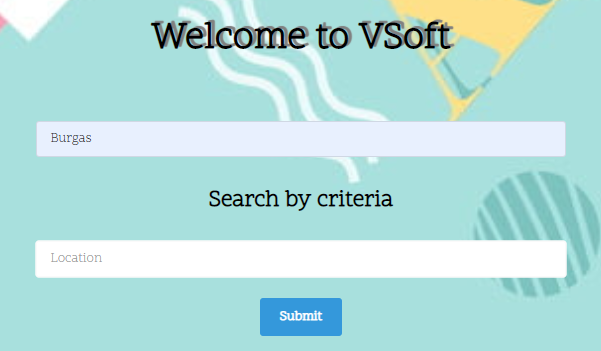
1. **Завършен вид**

Ако сме успели да преминем през всичките стъпки и правилно сме работили по проекта трябва да получим следния вид:

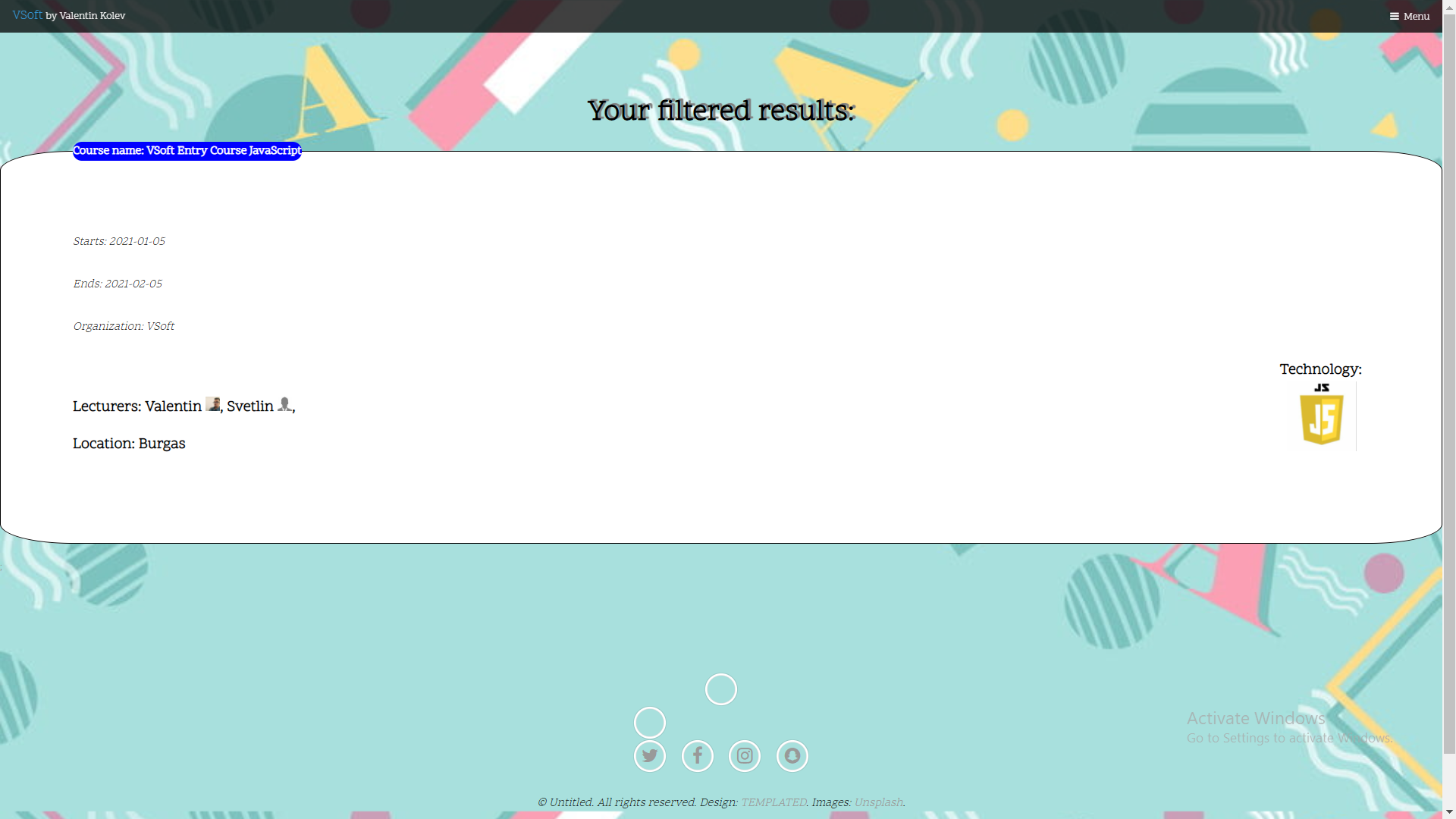


Включен е hover ефект към CSS частта за допълнително отличаване на избрания от нас курс. В търсачката въвеждаме желания от нас резултат, а от падащото меню избираме критерия , по който ще търсим, като всички те са запазени в search() метода, който ползва различни заявки, на база на избрания критерий.

Имаме лента в дясно, която може директно да ползваме, за да достъпим Admin Panel-a.



Търсим по локация, с избран град Бургас



Получен резултат на база локация Бургас