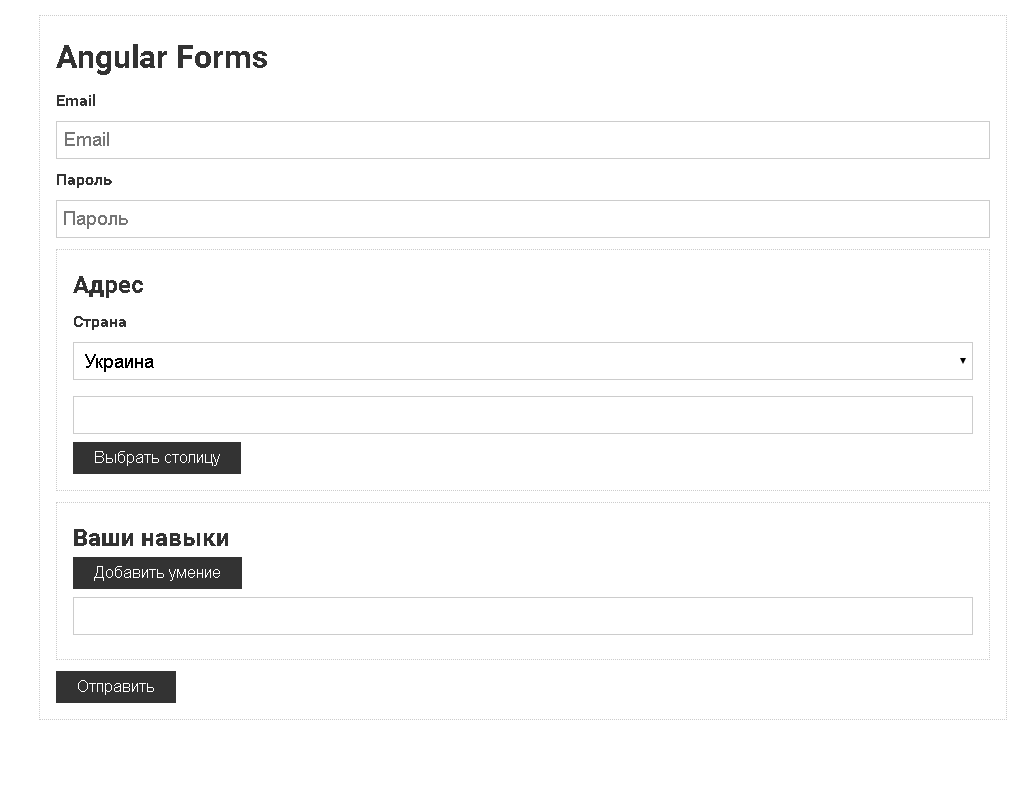
**Фреймворк Angular 9**

**Лабораторная работа № 5**

**Взаимодействие с формами в Angular**

**Задача. Создать форму как на рисунке. Для полей email и password сделать валидацию. Связать все поля с кнопкой «Отправить». Создать группу полей блока «Адрес». Предусмотреть динамическое добавление полей «умения». Создать свой валидатор, который будет запрещать почту из доменной зоны «.ru»**



**Ход работы**

1. Создайте новый проект на Angular. В файле styles.scss задайте стили из приведенного примера:

@import url('https://fonts.googleapis.com/css?family=Roboto');  
  
\* {  
 box-sizing: **border-box**;  
 margin: 0;  
 padding: 0;  
}  
  
body {  
 font-family: 'Roboto', **sans-serif**;  
 font-size: 1**rem**;  
 line-height: 1.6;  
 background-color: #fff;  
 color: #333;  
}  
  
.container {  
 max-width: 1000**px**;  
 margin: 0 **auto**;  
 padding: 1**rem**;  
}  
  
a {  
 text-decoration: **none**;  
}  
  
a:hover {  
 color: #666;  
}  
  
ul {  
 list-style: **none**;  
}  
  
img {  
 width: 100%;  
}  
  
.btn {  
 display: **inline-block**;  
 background: #333333;  
 color: #fff;  
 padding: 0.4**rem** 1.3**rem**;  
 font-size: 1**rem**;  
 border: **none**;  
 cursor: **pointer**;  
 margin-right: 0.5**rem**;  
 transition: **opacity** 0.2**s ease-in**;  
 outline: **none**;  
   
 &:disabled {  
 background: #ccc;  
 color: #000;  
 cursor: **not-allowed**;  
 }  
}  
  
.btn:hover {  
 opacity: 0.8;  
}  
  
.form-control {  
 display: **block**;  
 margin-top: 0.3**rem**;  
 margin-bottom: .5**rem**;  
  
 label {  
 display: **block**;  
 font-weight: **bold**;  
 font-size: 1**rem**;  
 margin-bottom: .5**rem**;  
 }  
  
 input {  
 margin-bottom: 0;  
 }  
  
 .validation {  
 color: **red**;  
 }  
}  
  
.card {  
 padding: 1**rem**;  
 border: #ccc 1**px dotted**;  
 margin: 0.7**rem** 0;  
}  
  
input,  
select,  
textarea {  
 display: **block**;  
 width: 100%;  
 padding: 0.4**rem**;  
 font-size: 1.2**rem**;  
 border: 1**px solid** #ccc;  
 margin-bottom: 1**rem**;  
}  
  
hr {  
 margin: .5**rem** 0;  
}

В шаблоне компонента app.component.html разработаем скелет формы

<div class**="container"**>  
 <form class**="card"**>  
 <h1>**Angular Forms**</h1>  
  
 <div class**="form-control"**>  
 <label>**Email**</label>  
 <input type**="text"** placeholder**="Email"**>  
 <div class**="validation"**></div>  
 </div>  
  
 <div class**="form-control"**>  
 <label>**Пароль**</label>  
 <input type**="password"** placeholder**="Пароль"**>  
 <div class**="validation"**></div>  
 </div>  
  
 <div class**="card"**>  
 <h2>**Адрес**</h2>  
  
 <div class**="form-control"**>  
 <label>**Страна**</label>  
  
 <select>  
 <option value**="ua"**>**Украина**</option>  
 <option value**="pl"**>**Польша**</option>  
 <option value**="de"**>**Германия**</option>  
 </select>  
 </div>  
  
 <div class**="form-control"**>  
 <input type**="text"**>  
 </div>  
  
 <button class**="btn"**>**Выбрать столицу**</button>  
 </div>  
  
 <div class**="card"**>  
 <h2>**Ваши навыки**</h2>  
 <button class**="btn"**>**Добавить умение**</button>  
 <div class**="form-control"**>  
 <label></label>  
 <input type**="text"**>  
 </div>  
 </div>  
  
 <button class**="btn"**>**Отправить**</button>  
  
 </form>  
</div>

1. В файле app.module.ts импортруйте модуль FormsModule и ReactiveFormsModule

import {BrowserModule} from '@angular/platform-browser'  
import {*NgModule*} from '@angular/core'  
  
import {AppComponent} from './app.component'  
import {FormsModule, ReactiveFormsModule} from '@angular/forms'

@NgModule({  
 declarations: [  
 AppComponent,  
 ],  
 imports: [  
 BrowserModule,  
FormsModule,  
ReactiveFormsModule

],  
 providers: [  
  
 ],  
 bootstrap: [AppComponent]  
})  
export class AppModule {  
}

1. В компоненте app.component.ts подключим классы по работе с формами.

export class AppComponent implements OnInit{  
 form: FormGroup  
 ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl(''),  
 password: new FormControl(null)  
 })  
 }  
  
 submit() {  
 const formControl = {...this.form.value}  
 console.log(formControl)  
 }  
}

1. В шаблоне компонента свяжем форму html с классом

<form class**="card"** [formGroup]**="**form**"** (ngSubmit)**="**submit()**"**>

<input type**="text"** placeholder**="Email"** formControlName**="email"**>

<input type**="password"** placeholder**="Пароль"** formControlName**="password"**>

Если в шаблоне есть кнопки, то нужно обязательно указать тип.

<button class**="btn"** type**="button"**>**Выбрать столицу**</button>

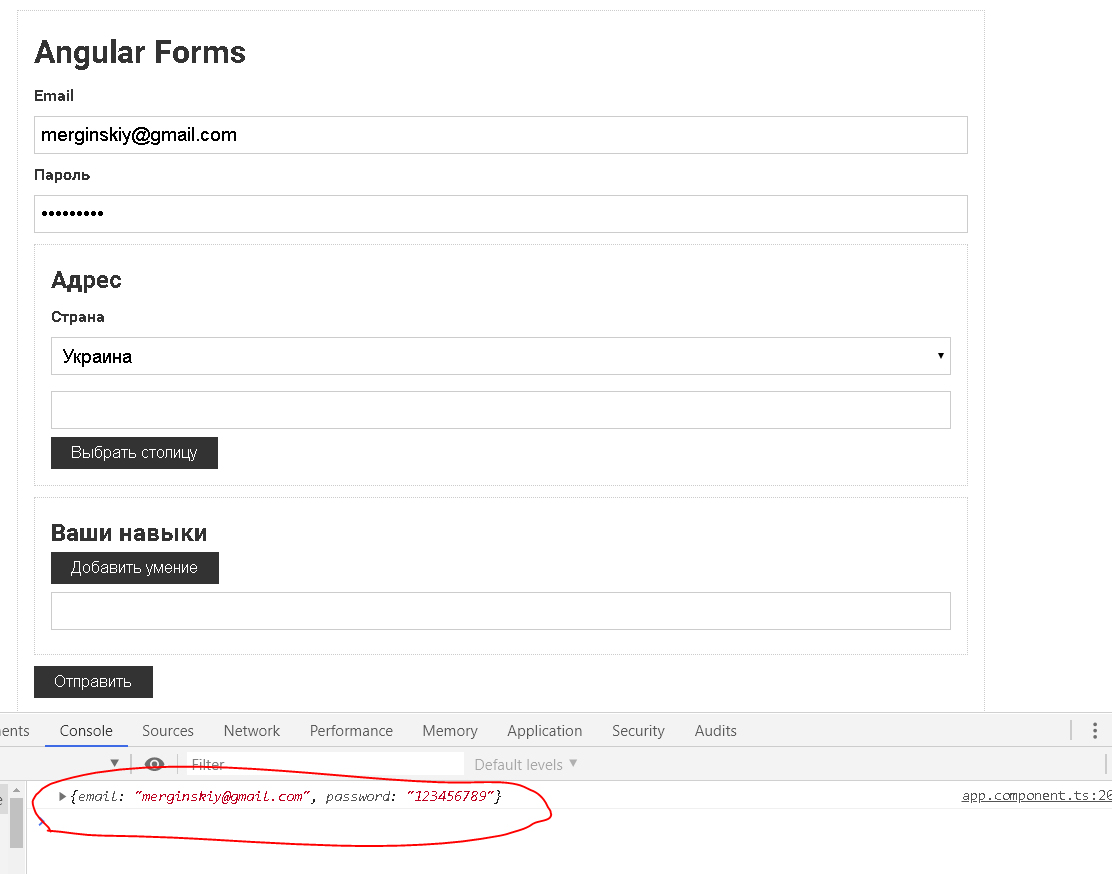
<button class**="btn"** type**="button"**>**Добавить умение**</button>

<button class**="btn"** type**="submit"**>**Отправить**</button>

1. Для проверки связи формы с компонентом реализуем метод submit()

export class AppComponent implements OnInit{  
 form: FormGroup  
 ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl(''),  
 password: new FormControl(null)  
 })  
 }  
  
 submit() {  
 const formControl = {...this.form.value}  
 console.log(formControl)  
 }  
}

Проверьте результат, запустив приложение.



1. Создадим валидацию формы средствами Angular. Для этого используется специальный класс Validators.

ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl('', [Validators.email, Validators.required]),  
 password: new FormControl(null, [Validators.required, Validators.minLength(6)])  
 })  
 }

**!!!Самостоятельно.**  **Измените валидацию пароля на проверку не менее 4 символов и не более 10.**

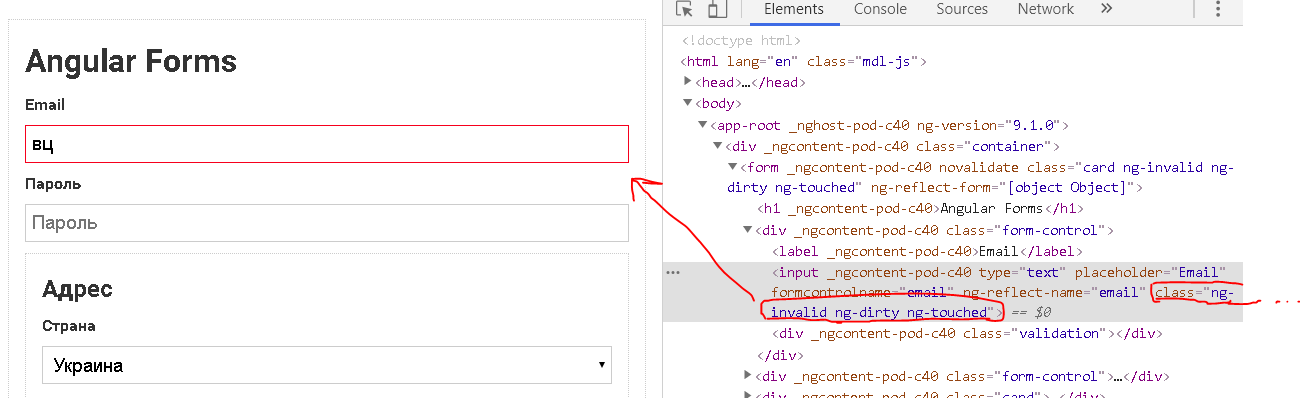
1. Подключим валидатор к форме для этого добавим директиву к кнопке «Отправить».

<button class**="btn"** type**="submit"** [disabled]**="**form.invalid**"**>**Отправить**</button>

Т.е. пока форма будет инвалидной кнопка «Отправить» будет не доступна.

1. Для визуализации состояния валидации пропишем стили для классов полей input. Такие классы Angular создает полям в зависимости от событий которые с ними происходит. В файле app.cpmponent.scss запишем:

input {  
 &.ng-invalid.ng-touched {  
 border: 1**px solid red**;  
 }  
 &.ng-valid.ng-touched {  
 border: 1**px solid green**;  
 }  
}



1. Для того чтобы пользователь понимал причину ошибки валидации пропишем в шаблоне компонента сообщения, которые будут указывать на предмет ошибки а также условия их появления через структурный декоратор ngIf.

<div class**="form-control"**>  
 <label>**Email**</label>  
 <input type**="text"** placeholder**="Email"** formControlName**="email"**>  
 <div class**="validation"** *\*ngIf***="**form.get('email').invalid && form.get('email').touched**"**>  
 <small *\*ngIf***="**form.get('email').errors.required**"**>**Поле email не может быть пустым.** </small>  
 <small *\*ngIf***="**form.get('email').errors.email**"** >**Введите корректный email.** </small>  
  
 </div>  
</div>  
  
<div class**="form-control"**>  
 <label>**Пароль**</label>  
 <input type**="password"** placeholder**="Пароль"** formControlName**="password"**>  
 <div class**="validation"** *\*ngIf***="**form.get('password').touched && form.get('password').invalid**"**>  
 <small *\*ngIf***="**form.get('password').errors.required**"**>**Поле password не может быть пустым.** </small>  
 <small *\*ngIf***="**form.get('password').errors.minlength.requiredLength**"**>**Неправильный password ввдеите еще** {{form.get('password').errors.minlength.requiredLength - form.get('password').errors.minlength.actualLength}} **символов**</small>  
  
 </div>  
</div>

1. Реализуем следующий блок формы адрес и создадим отдельную группу для него. Для этого в шаблоне компонента

<div class**="card"** formGroupName**="address"**>  
 <h2>**Адрес**</h2>  
  
 <div class**="form-control"**>  
 <label>**Страна**</label>  
  
 <select formControlName**="country"**>  
 <option value**="ua"**>**Украина**</option>  
 <option value**="pl"**>**Польша**</option>  
 <option value**="de"**>**Германия**</option>  
 </select>  
 </div>  
  
 <div class**="form-control"**>  
 <input type**="text"** formControlName**="city"**>  
 </div>  
  
 <button class**="btn"** type**="button"** (click)**="**setCapital()**"**>**Выбрать столицу**</button>  
</div>

Свяжем контролы формы с компонентом в файле app.component.ts

ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl('', [Validators.email, Validators.required]),  
 password: new FormControl(null, [Validators.required, Validators.minLength(6)]),  
 address: new FormGroup({  
 country: new FormControl('ua'),  
 city: new FormControl('Запорожье', Validators.required)  
 })  
 })  
 }

И реализуем метод setCapital() для связывания выпадающего списка с полем «Город»

setCapital() {  
 const mapCity = {  
 ua: "Киев",  
 pl: "Варшава",  
 de: "Берлин"  
 }  
 const getCity = this.form.get('address').value.country  
this.form.get('address').patchValue({city: mapCity[getCity]});  
 }

**!!!Самостоятельно.**  **Добавить валидацию на поле «Столица». Сделать его обязательным не менее 3 символов.**

1. Реализуем последний блок формы добавления умений. Необходимо разработать структуру которая позволит динамически создавать новые контролы и помещать их в массив.

ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl('', [Validators.email, Validators.required]),  
 password: new FormControl(null, [Validators.required, Validators.minLength(6)]),  
 address: new FormGroup({  
 country: new FormControl('ua'),  
 city: new FormControl('Запорожье', Validators.required)  
 }),  
 skills: new FormArray([])  
 })  
  
 }

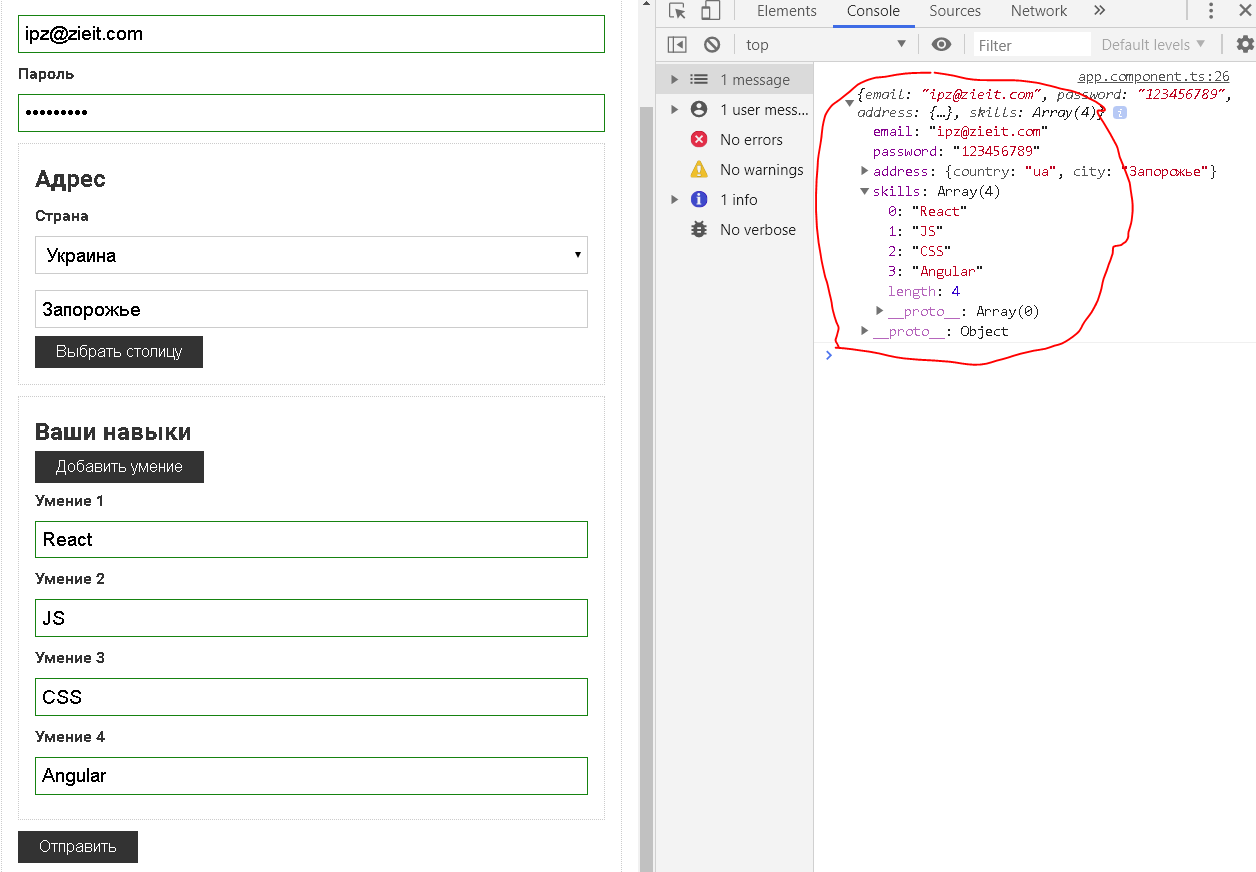
В шаблоне компонента сделаем следующие изменения:

<div class**="card"** formArrayName**="skills"**>  
 <h2>**Ваши навыки**</h2>  
 <button class**="btn"** type**="button"** (click)**="**addSkill()**"**>**Добавить умение**</button>  
 <div class**="form-control"** *\*ngFor***="**let *control* of this.form.get('skills').controls; let *idx*=index**"**>  
 <label> **Умение** {{*idx*+1}}</label>  
 <input type**="text"** [formControlName]**="***idx***"**>  
 </div>  
</div>

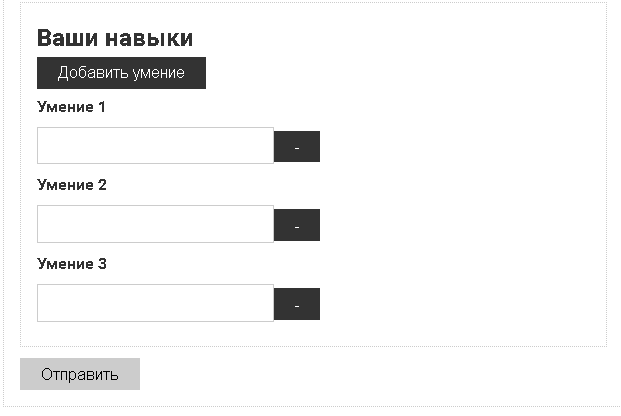
И реализуем метод addSkill() в компоненте:

addSkill() {  
 const control = new FormControl('', Validators.required);  
 (<FormArray>this.form.get('skills')).push(control);  
}

Проверьте данные которые передает форма:

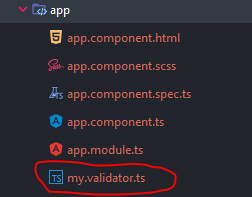


**!!!Самостоятельно.**  **Добавить к последнему блоку кнопку удаления лишних или случайно нажатых навыков. См. рис. Повесить на кнопку метод обработки и удаления элемента из DOM дерева. Для удаления использовать следующий метод (<FormArray>this.form.get('skills')).removeAt(index) где index это индекс элемента массива, который удаляется.**



1. Разработаем собственный валидатор, который будет запрещать почту из списка запрещенных доменных зон указом СНБО №133/2017.

Для этого создадим в приложении файл my.validator.ts



И добавим в него следующий код:

import {FormControl} from "@angular/forms";  
  
export class MyValidator {  
 static restrictedEmails(control: FormControl): {[key: string]: boolean } {  
 if ((control.value).includes('.ru') ){  
  
 return {restrictedRusEmail: true};  
 }  
 return null;  
 }  
  
}

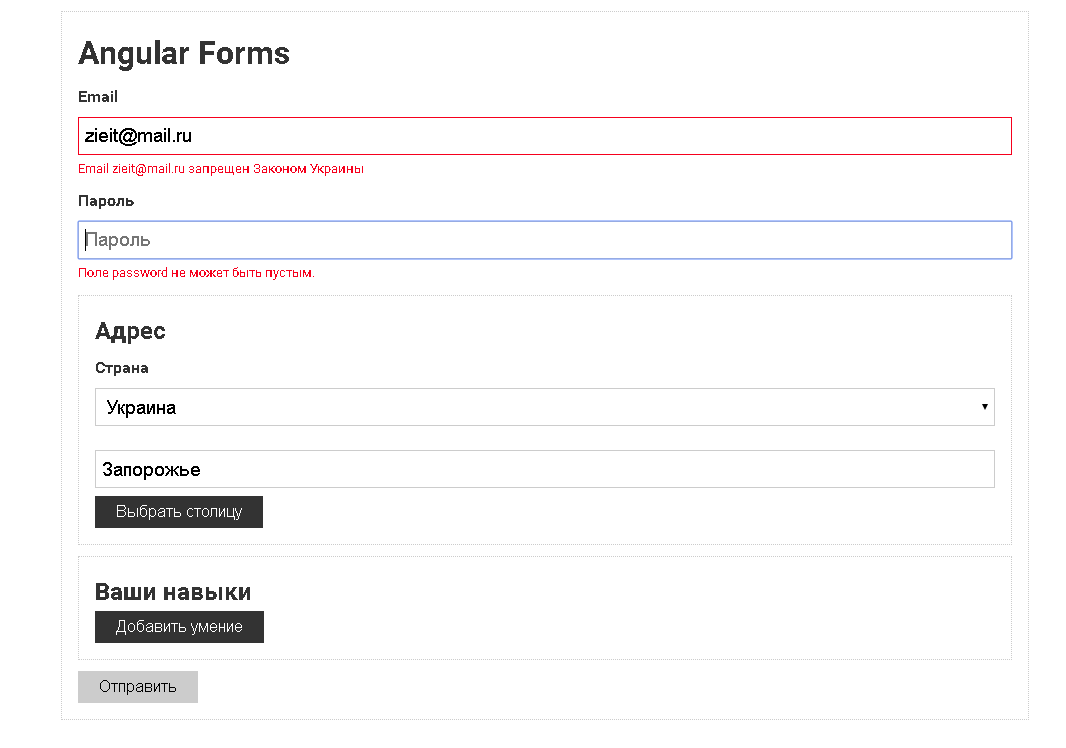
В компоненте включим наш валидатор для поля email не забывая импортировать его из файла my.validator.ts

ngOnInit() {  
 this.form = new FormGroup({  
email: new FormControl('', [Validators.email, Validators.required, MyValidator.restrictedEmails]),  
 password: new FormControl(null, [Validators.required, Validators.minLength(6)]),  
 address: new FormGroup({  
 country: new FormControl('ua'),  
 city: new FormControl('Запорожье', Validators.required)  
 }),  
 skills: new FormArray([])  
 })

Ну и для того чтобы пользователь мог видеть причину отказа записывать почту запишем в шаблоне компонента следующий код:

<small *\*ngIf***="**form.get('email').errors.restrictedRusEmail**"** >**Email** {{form.get('email').value}} **запрещен Законом Украины** </small>

Проверьте работу валидатора:



1. Сделаем очистку форму после добавления и отправки данных. Для этого в метод submit() добавляется для экземпляра класса form метод reset()

submit() {  
 const formControl = {...this.form.value}  
 console.log(formControl)  
 this.form.reset()  
 }