Backend

- Verzija Laravel-a 8.4, trenutno aktuelna 9.5.1
- Nepotrebno dodati npm paketi u backend projekat
- Dosta nepotrebnih fajlova i koda poput SolarOptimizationService.php

ColorsController:
- nisu korisceni return type-ovi
- na pocetku fajla nije definisan `declare(strict_types = 1);`
- Bilo bi dobro kreirati custom request (CreateColorRequest) koji ce u sebi imati validaciona pravila
- Sav ovaj kod bi trebao biti prebacen u novi servis, koji bi bio pozvan iz kontrolera

```
$data = $request->validate([
    'name' => 'required|string',
    'hex_value' => 'required|string|min:6|max:6',
    'status' => 'required|boolean'
]);

$newColor = new Color();
$newColor->fill($data);
$newColor->hex_value = "#" . $data['hex_value'];
$newColor->save();
```

- Sta ce se desiti u slucaju da `$request->wantsJson()`ne bude `true`? Trebao bi da se pokrije i taj slucaj

```
if ($request->wantsJson()) {
    return JsonResource::make($newColor)->withSuccess(__('New color has been saved!'));
}
```

- Umesto JsonResource-a moze da se koristi dafault-ni JsonResponse
- Kod u svakoj od metoda kontrolera bi trebao da bude okruzen sa try-catch blokom i greske bi trebalo da budu handle-ovane.
- Bilo bi pozeljno da su uradjeni Unit testovi, dokumentacija iznad svake metode i swagger dokumentacija

Frontend

- Nema potrebe stavljati prop u state

```
function Color({color, colors, setColors}) {
  const [colorName, setColorName] = useState(color.name);
  const [colorHexValue, setColorHexValue] = useState(color.hex_value);
  const [colorStatus, setColorStatus] = useState(color.status);
```

- Ukoliko bi zeleo da prikazes status u App.js, obzirom da promenu statusa cuvas u state-u Color.jsx, prikazao bi pogresan status. Najbolje bi bilo da se podaci koji se koriste na vise mesta cuvaju u globalnom state-u i samo na jednom mestu vrsi manipulacija podataka (koriscenje Redux-a npr.).
- Sadrzaj onClick funkcije izbaciti van HTML-a
- Treba pokriti i slucaj kada kod ne udje u if

```
if(result.data.status === "ok") {
    setColors(colors.filter(function(color){
      return color.id !== actionColor.id;
    }));
}
```

- Bilo bi pozeljno, iz ugla korisnika, da greske budu ispisane ispod svakog polja

```
  <div className="form-group">
      <label htmlFor="color-name">Name: </label>
      <input id="color-name" type="text" value={name} onChange={(element :ChangeEvent<HTMLInputElement> ) => {setName(element.target.value)}}/>
  </div>
  <div className="form-group">
      <label htmlFor="color-hex-value">Hex Value: </label>
      <input id="color-hex-value" type="text" value={hexValue} onChange={(element :ChangeEvent<HTMLInputElement> ) => {setHexValue(element.target.value)}}/>
  </div>
  <div className="form-group">
      <label htmlFor="color-status">Status: </label>
      <input id="color-status" type="checkbox" checked={status} onChange={(element :ChangeEvent<HTMLInputElement> ) => {setStatus(element.target.checked)}}/>
  </div>
<div className="error-msg">{errors}</div>
</div>
```

- Prilikom editovanja boje takodje se boja dodaje u postojeci niz boja pa ce biti duplirana

```
editColor(colorsCOU[0].id).then(result => {
    if(result.data.status === "ok") {
        setColors(prev => [...prev, result.data.data]);
        setErrors( value: "");
        setName( value: "");
        setHexValue( value: "");
        setStatus( value: false);
    }
```

- Na svaki API poziv za create/edit najlakse bi bilo pozvati `getColors` API koji bi nam vratio sve boje, tako bi uvek imali validan state

- Trebalo bi disable-ovati `Save` dugme pri create/edit i prikazati loader