Alan Chachoł 3F Person

1.Plik PersonForm.jsx. Wykorzystując informacje zdobyte z poprzednich lekcji wykonuje następujący kod:
PearsonForm renderuje formularz edycji danych osobowych.Zawiera on pola do wprowadzenie peselu, imienia, nazwiska,adresu,numeru telefonu, adresu e-mail oraz płci.
Wartości które zostały wprowadzone są przekazywane do onInputChange za pomocą funkcji onChange.

```
"../PearsonForm.css";
const PearsonForm = (props) => {
  <div className='EdycjaLekcji_input-grupa'>
              <label htmlFor='pesel'>Podaj pesel: </label>
                  placeholder='Wpisz pesel'
                  type='number
                      props.onInputChange({ [e.target.name]: e.target.value })
           <div className='EdycjaLekcji_input-grupa'>
              <label htmlFor='name'>Podaj imie: </label>
                  placeholder='Wpisz imie'
                  type='name
                      props.onInputChange({ [e.target.name]: e.target.value })
           <div className='EdycjaLekcji_input-grupa'>
              <label htmlFor='lastname'>Podaj nazwisko: </label>
                  placeholder='Wpisz nazwisko'
                 type='text'
id='lastname
                      props.onInputChange({ [e.target.name]: e.target.value })
           <div className='EdycjaLekcji_input-grupa'>
              <label htmlFor='lastname'>Podaj pełny adres: </label>
```

Gdy formularz zostanie wypełniony użytkownik może użyc przycisku OK, który spowoduje wywowałanie funkcji 'onSave'.s

2.Plik Person

zarządamy tutaj danymi osobowymi. Używamy 'OutPust' do rendoerowania danych osobowych oraz 'PearsonFrom' do interakcji z użytkownikiem.

```
if(lastPearson.pesel == editPearson.pesel){
            editPearson.pesel = "Błąd powtarzający się pesel!";
       const newLesson = {
           pesel:editPearson.pesel || lastPearson.pesel,
           name: editPearson.name || lastPearson.name,
           lastname: \ edit Pearson.lastname \ || \ lastPearson.lastname,
           adres: editPearson.adres || lastPearson.adres,
           tel: editPearson.tel || lastPearson.tel,
           email: editPearson.email || lastPearson.email,
           plec: editPearson.plec || lastPearson.plec,
           Pearson: [...Pearson, newLesson],
           editPearson: {
render() {
   const Pearson = this.state.Pearson.map((elm) => {
           <OutPut
               key={elm.pesel}
               name={elm.name}
               pesel={elm.pesel}
                lastname={elm.lastname}
               adres = {elm.adres}
               tel = {elm.tel}
               email={elm.email}
               plec = {elm.plec}
```

```
});
      return (
         <div className='EdycjaLekcji'>
            <br></br>
             <h1 id='h1'>Dane Osobowe</h1>
             <PearsonForm
                onInputChange={(val) => this.addPearson(val)}
                onSave={() => this.savePearson()}
             ></PearsonForm>
                   Pesel
                   Name
                   Lastname
                   Adres
                   Tel
                   Email
                   Płeć
                   {Pearson}
            export default Pearson;
```

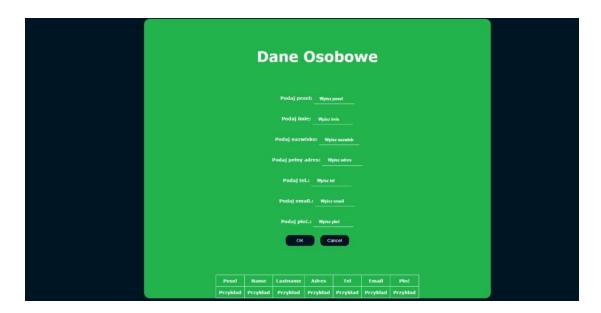
3.Plik Output.jsx

Na poczatku definiujemy 'BoxOutPut' i przyjmuje on właściwości'props'.W dalszej części kodu są definiowane oczekiwane typy właściwości.

```
import Proptypes from "prop-types";
function BoxOutPut(props) {
   return (
      {props.pesel}
             {props.name}
             {props.lastname}
             {props.adres}
             {props.tel}
             {props.email}
             {props.plec}
      );
BoxOutPut.propTypes = {
   pesel: Proptypes.string,
   name: Proptypes.string,
   lastname: Proptypes.string,
   adres: Proptypes.string,
   tel: Proptypes.string,
   email: Proptypes.string,
   plec: Proptypes.string,
};
export default BoxOutPut;
```

Ostatnia linia pozwala na wykorzystanie 'BoxOutPust' w innych częsciach aplikacji

4. Wygląd strony



5. Wynik po dodaniu do tabeli

