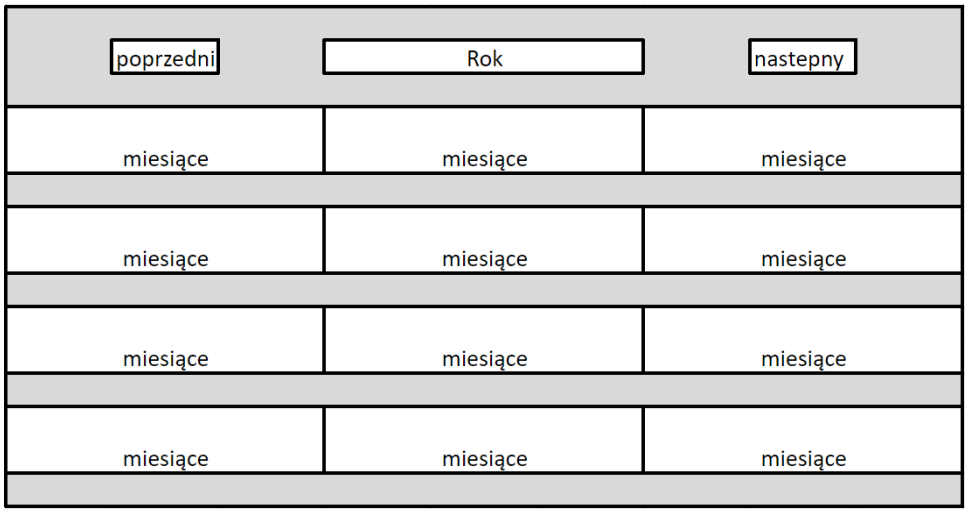
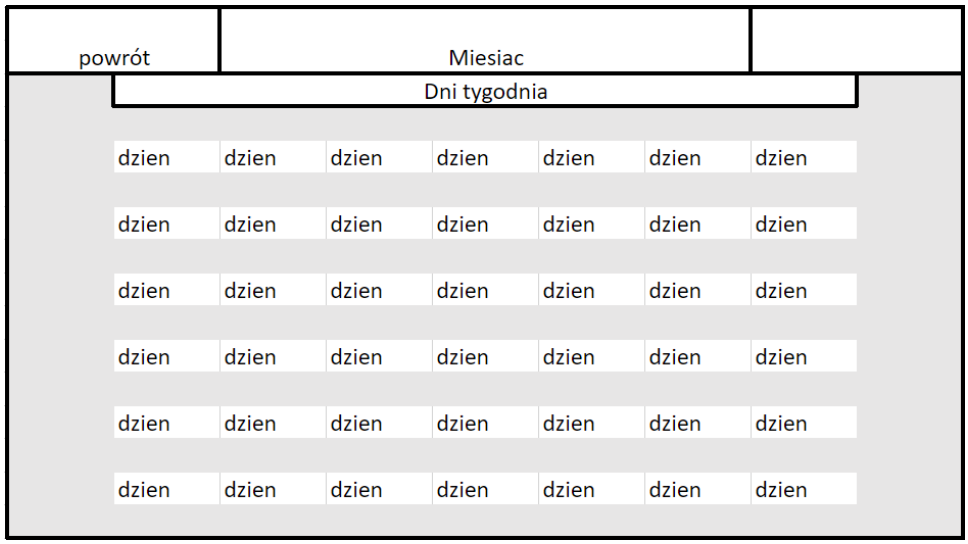
Podział strony

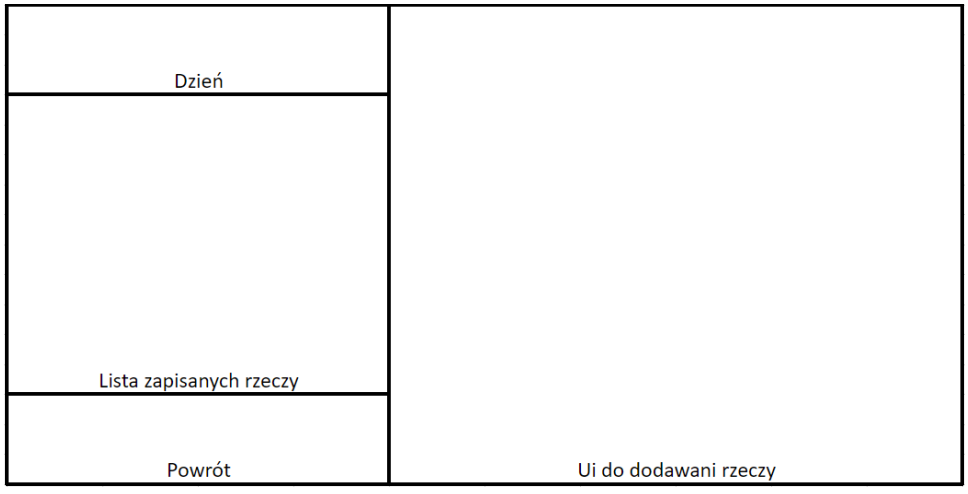
-Główny widok



-Widok Miesiąca



-Widok dnia



A) Widok główny:

-Buttony poprzedni i następny kontrolujące wyświetlany rok

-Nagłówek wyświetlający wybrany obecnie rok

-Zmniejszone Widoki miesiąca, po kliknięciu przechodzimy do widoku kliknięgo miesiąca

B) Widok Miesiąca

-Button pozwalający na powrót do widoku głównego

-Nagłówek wyświetlający wybrany miesiąc i rok

-Nagłówek wyświetlający listę dni tygodnia

-Tabelka zawierająca listę dni

C) Widok Dnia

-Nagłówek wyświetlający wybrany dzień

-Lista Zapisanych rzeczy pobrana z bazy

-Przycisk pozwalający na powrót do widoku miesiąca

-UI do dodawania rzeczy do bazy danych(formularz)

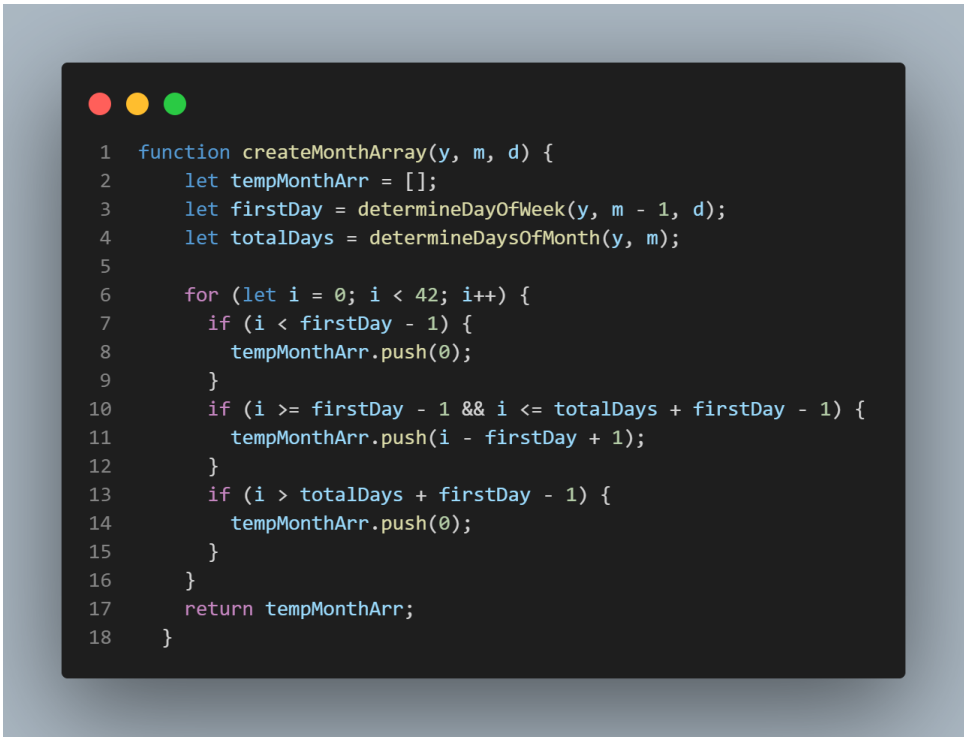
Strona statyczna:

A) Do stworzenia statycznej strony potrzebowałem dwóch funkcji określających od którego dnia tygodnia rozpoczyna się miesiąc a także ile w miesiącu znajduje się dni.

```
function determineDaysOfMonth(y, m) {
    return new Date(y, m, 0).getDate();
  }

  function determineDayOfWeek(y, m, d) {
    let dayOfWeek = new Date(y, m, d).getDay();
    dayOfWeek = dayOfWeek == 0 ? 6 : dayOfWeek - 1;
    return dayOfWeek;
  }
```

B) A także funkcji która na podstawie dwóch wyżej wymienionych tworzy tablice zawierająca dni miesiąca.



```
function createMonthArray(y, m, d) {
    let tempMonthArr = [];
    let firstDay = determineDayOfWeek(y, m - 1, d);
    let totalDays = determineDaysOfMonth(y, m);

    for (let i = 0; i < 42; i++) {
      if (i < firstDay - 1) {
        tempMonthArr.push(0);
      }
      if (i >= firstDay - 1 && i <= totalDays + firstDay - 1) {
        tempMonthArr.push(i - firstDay + 1);
      }
      if (i > totalDays + firstDay - 1) {
        tempMonthArr.push(0);
      }
    }
    return tempMonthArr;
  }
```

C) Następnie zmapowałem tablice dni miesiąca i dni tygodnia.

D) Propsy których użyłem: year, month.

Odwzorowanie stanu interfejsu użytkownika

-Stan określający co powinno się renderować.

-Data

-Notatki określający notatki zapisane przez użytkownika.

Określenie, gdzie powinien mieścić się stan.

-Główny widok:

-Data.

- Stan określający co powinno się renderować.

-Komponent miesiąca:

-Data (rok i miesiąc).

-Komponent widoku Dnia

-Notatki.

Dodanie przepływu danych w drugą stronę.

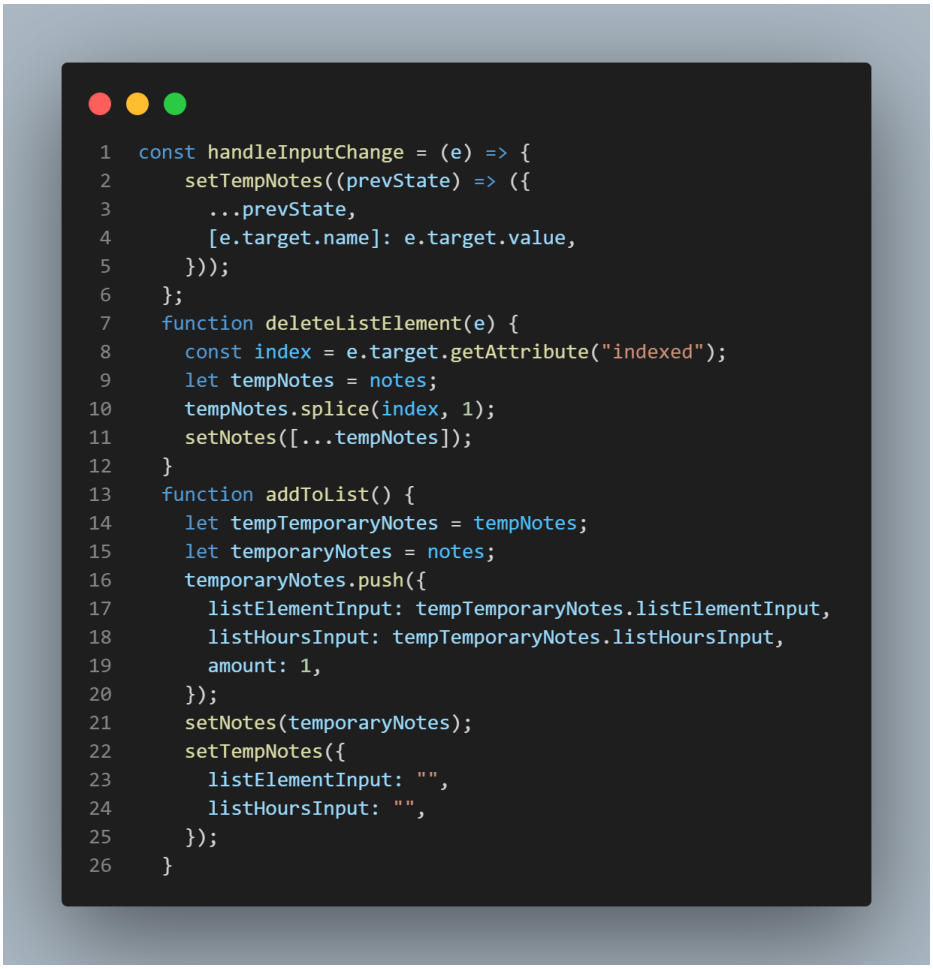
-Zmienianie widoku na podstawie klikania w kalendarz i dzień w kalendarzu (propsy: controlingFunctionCal, controlingFunctionDay) i powrót do poprzednich widoków.

```
function handleCalendarClickCal(e) {
    const monthVal = e.currentTarget.getAttribute("monthval");
    setWhatRender(1);
    setMonth(monthVal);
  }
  function handleCalendarClickDay(e) {
    const dayVal = e.currentTarget.getAttribute("dayval");
    setWhatRender(2);
    setDay(dayVal);
  }

  function revertHandleCalendarClickCal(e) {
    setWhatRender(0);
    setMonth(0);
  }
  function revertHandleCalendarClickDay(e) {
    setWhatRender(1);
    setDay(0);
  }
```

-Usuwanie i dodawanie do listy czynności a nastepnie usuwanie i dodawanie zapisywanie do localStorage.

-Lista czynności:



```
const handleInputChange = (e) => {
    setTempNotes((prevState) => ({
      ...prevState,
      [e.target.name]: e.target.value,
    }));
  };
  function deleteListElement(e) {
    const index = e.target.getAttribute("indexed");
    let tempNotes = notes;
    tempNotes.splice(index, 1);
    setNotes([...tempNotes]);
  }
  function addToList() {
    let tempTemporaryNotes = tempNotes;
    let temporaryNotes = notes;
    temporaryNotes.push({
      listElementInput: tempTemporaryNotes.listElementInput,
      listHoursInput: tempTemporaryNotes.listHoursInput,
      amount: 1,
    });
    setNotes(temporaryNotes);
    setTempNotes({
      listElementInput: "",
      listHoursInput: "",
    });
  }
```

-Lista LocalStorage:

```
function addToLocalStorage() {
    notes.forEach((element) => {
      let event = {
        name: element.listElementInput,
        time: element.listHoursInput,
        year: props.year,
        month: props.month,
        day: props.day,
      };
      localStorage.setItem(
        event.name + ": " + props.year + "-" + props.month + "-" + props.day,
        JSON.stringify(event)
      );
      setNotes([]);
    });
  }

  function returnArrayOfLocalStorage() {
    let notesFromLocalStorage = [];
    for (let i = 0; i < localStorage.length; i++) {
      let item = JSON.parse(localStorage.getItem(localStorage.key(i)));
      if (
        item.year == props.year &&
        item.month == props.month &&
        item.day == props.day
      ) {
        notesFromLocalStorage.push(item);
      }
    }
    return notesFromLocalStorage;
  }

  function returnLocalStorageList() {
    return memoizedLSArray.map((value, index) => {
      return (
        <li key={index}>
          <DayListElement
            del={deleteLocalStorageItem}
            number={index}
            index={index + 1}
            text={value.name}
            time={value.time}
          />
        </li>
      );
    });
  }
```

Elementy pobrane z localStorage po przycisnięciu usuń usuwają się ale komponent się nie rerenderuje (nie wiem dlaczego).