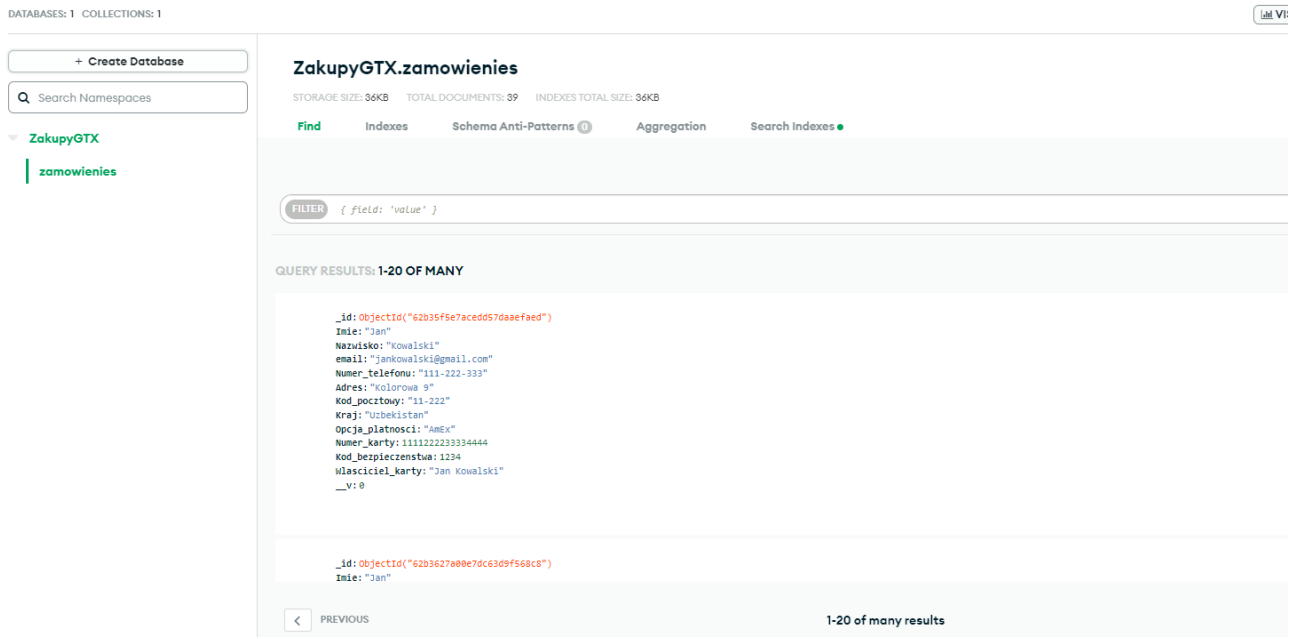
Dokumentacja Projektu		JPiTI
Autor	Sebastian Kaput, 106577	**Data oddania**
Kierunek, rok	Mechatronika, III rok, st. Stacjonarne (3,5-l)	15.07.2022
Specjalizacja	-	
Grupa	LAB 1	**Ocena**
Temat projektu	Formularz zakupowy wraz z panelem administracyjnym	

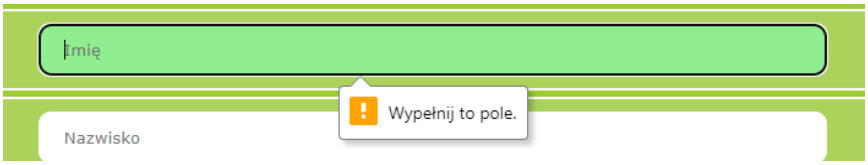
Opis tematyki projektu:

Projekt polega na wykonaniu w pełni działającego formularza zakupowego, który po odpowiednim jego uzupełnieniu i przejściu walidacji, zapisuje dane w chmurze na serwerze MongoDB.

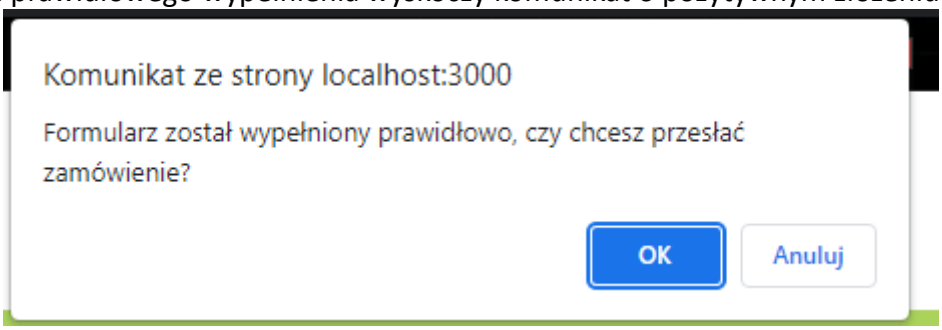


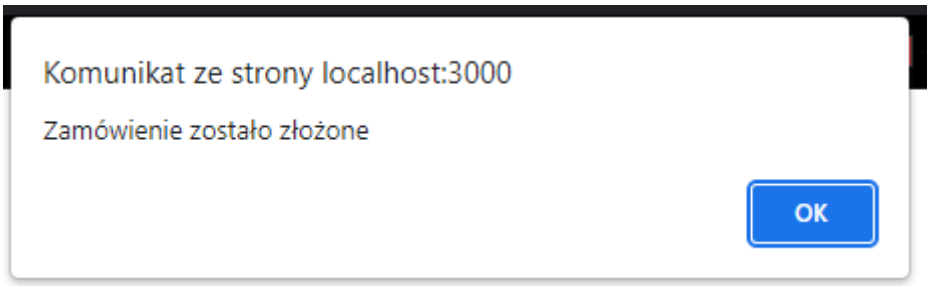
Użytkownik posiada możliwość uzupełnienia formularza oraz złożenia zamówienia przyciskiem „Kup Teraz”.

Po jego naciśnięciu zostanie on poinformowany o tym, czy dane zostały prawidłowo wpisane, gdy wystąpi błąd, formularz wskaże, gdzie został popełniony.

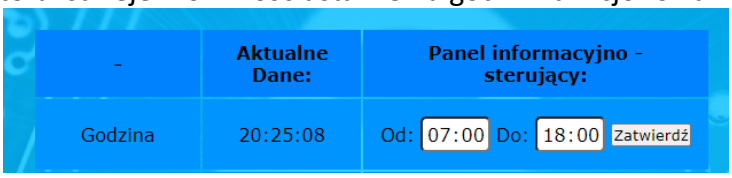


W przypadku prawidłowego wypełnienia wyskoczy komunikat o pozytywnym złożeniu zamówienia.



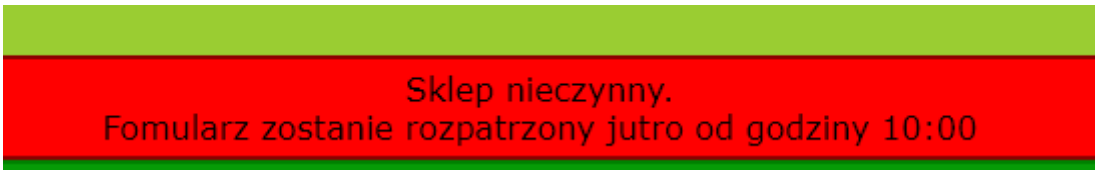


Po stronie administratora istnieje możliwość ustawienia godzin funkcjonowania sklepu.



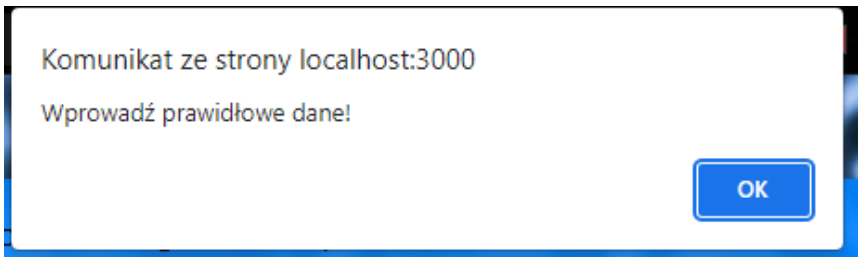
Po ustawieniu zakresu godzin, użytkownik będzie informowany o tym, czy sklep jest otwarty, czy zamknięty. W przypadku, gdy sklep będzie otwarty, będzie wyświetlana godzina, do której pozostanie on czynny.

W innym razie pojawi się informacja o jego zamknięciu oraz kiedy zostanie ponownie otworzony.

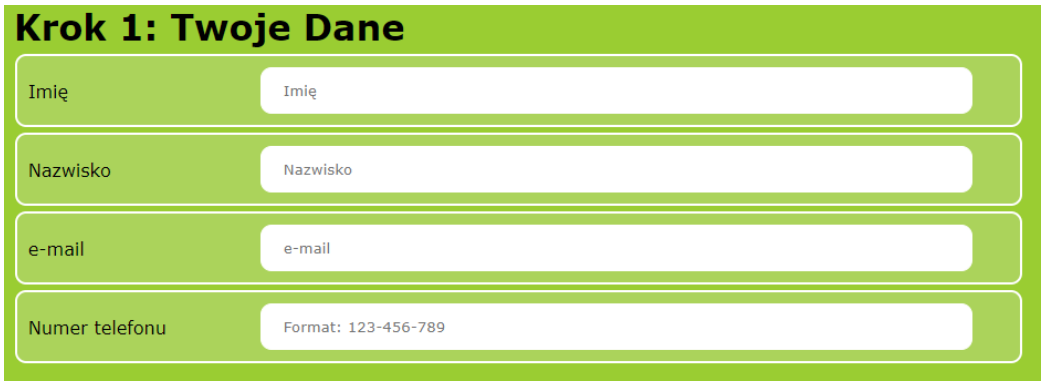


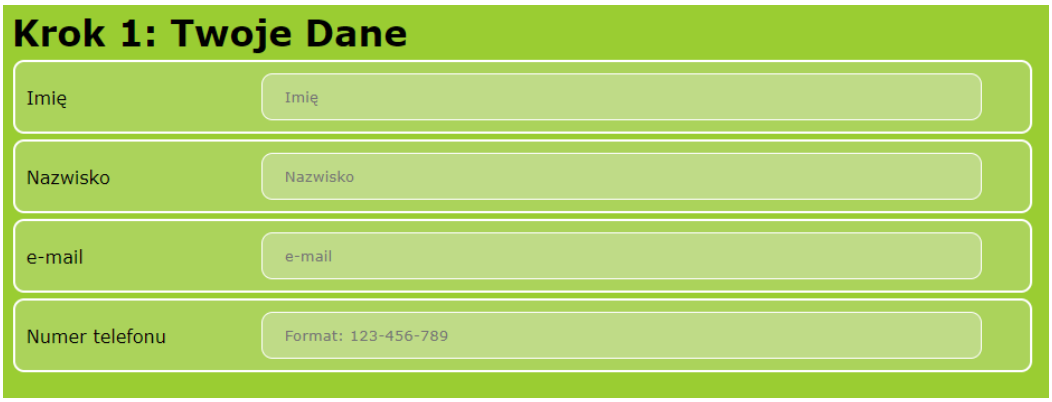
Następnie administrator ma możliwość ustawienia ilości sztuk produktu dostępnych na magazynie.

Może on wpisać zakres jedynie od 0 wzwyż. Próba wpisania innej wartości będzie skutkowała pojawieniem się komunikatu o nieprawidłowym działaniu.

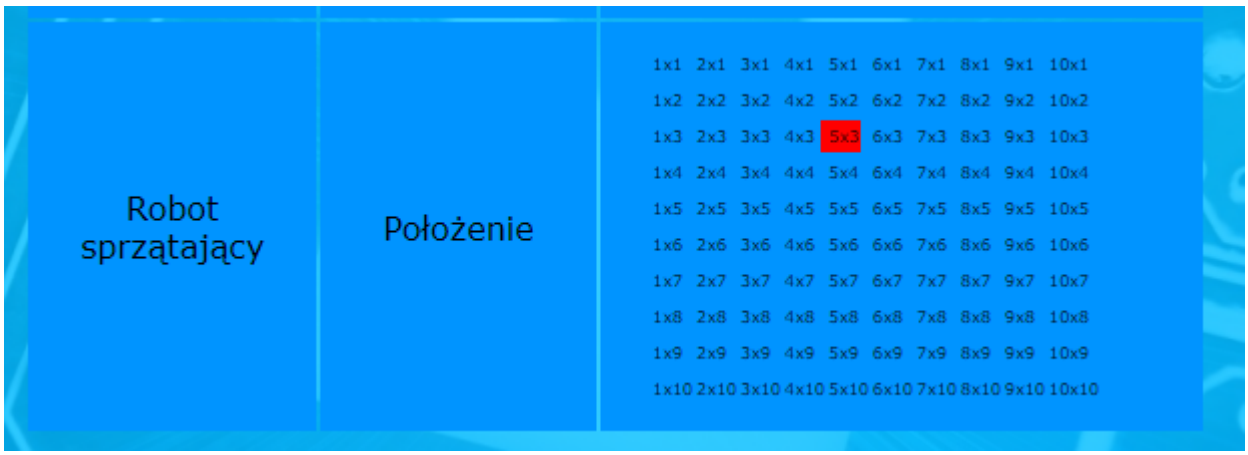


Gdy stan na magazynie spadnie do wartości „0”, formularz zostanie zablokowany i nie będzie można składać więcej zamówień, a pola do wpisywania danych zostaną wyłączone.
Poniżej przykład formularza aktywnego, gdy stan na magazynie >0 oraz gdy ilość sztuk spadła do 0.





Magazyn wyposażony jest w robota sprzątającego który udostępnia swoje położenie z pokoju o wymiarach 10 x 10 metrów. Dane te są przetwarzane i wyświetlane w tabeli 10 x 10 na panelu administratora z aktualnym jego położeniem, gdyż pracownicy często potykali się o niego nie wiedząc gdzie się znajduje.



Dodatkowo administrator może podejrzeć aktualne zamówienia wyświetlone w tabeli wraz ze wszystkimi informacjami wpisanymi przez użytkownika. Wygenerowanie tabeli następuje po kliknięciu przycisku „Zamówienia”

-	Imię	Nazwisko	E-Mail	Numer Telefonu	Adres	Kod Pocztowy	Kraj	Rodzaj Płatności	Numer Karty	Kod Bezpieczenstwa	Właściciel Karty
0	Jan	Kowalski	jankowalski@gmail.com	111-222-333	Kolorowa 9	11-222	Uzbekistan	AmEx	1111222233334444	1234	Jan Kowalski
1	Rafał	Miazmosowski	lubieskaLy@gmail.com	692-325-244	Radosna 69	01-100	Polska	Visa	1111111111112222	985	Jedenaście sajtów
2	Piotr	Pawłowski	nivea@gmail.com	123-654-234	Smutna 9/24	11-222	Francja	Mastercard	1111222277778888	456	Jan Brzydki
3	Bartaba	Kura	kot@interia.pl	321-345-643	Zielona 6	56-234	Ukraina	AmEx	6666777755554444	586	Barbara Wesoła
4	Michał	Smutny	michal@wp.pl	534-753-245	Hetmańska 120	24-467	Polska	Nerka	7777555599997777	763	Grzesiu Wielki

2) W dokumentacji mają być zawarte informacje w postaci opisów bądź wypunktowań: • opis tematyki projektu – opis słowny założeń projektu, jaki fragment rzeczywistości jest objęty działaniem aplikacji, co użytkownik może zrobić w aplikacji,

Opis wykorzystanych technologii

Do wykonania projektu wykorzystano technologie programowania takie jak:
- HTML
- CSS
- JS
- EJS

W celu działania aplikacji, wymagane są paczki takie jak:
-ejs (https://ejs.co/)
-espress (https://developer.mozilla.org/pl/docs/Learn/Server-side/Express_Nodejs/Introduction)
-mongoose (https://www.mongodb.com/)
-body-parser (https://www.npmjs.com/package/body-parser)

Do wykonania projektu wykorzystani środowisko takie jak Visual Studio Code (dostępne pod adresem https://code.visualstudio.com/)

Strona została wykonana w HTML / EJS, za ustawienia graficzne, responsywność, zachowanie elementów na stronie odpowiadał CSS, natomiast za funkcjonowanie i wykonywanie akcji np. po kliknięciu przycisku stał JavaScript. Całość została umieszczona na serwerze wykonanym w NodeJS.

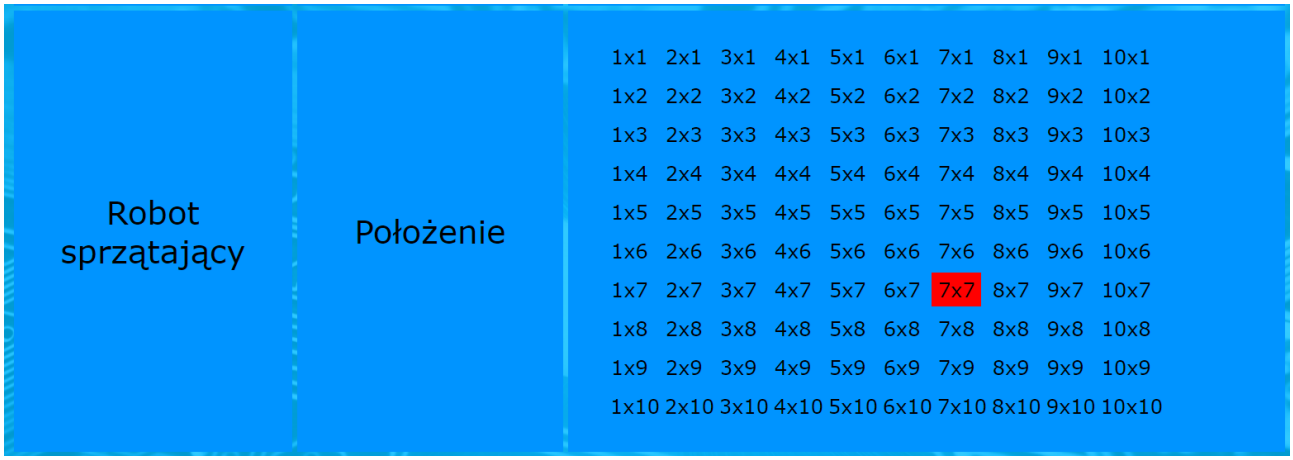
Instrukcja uruchomienia aplikacji & co trzeba mieć zainstalowane

Należy uruchomić główny folder z plikami w Visual Studio Code, dokonać instalacji wymaganych paczek „NPM Install", dokona się wtedy instalacja wszystkich wymaganych plików używanych w programie, a następnie uruchomić w terminalu „NPM Start"

- body-parser (wersja1.20.0)
- ejs (wersja 3.1.8)
- express (wersja 4.18.1)
- mongoose (wersja 6.4.0)
- nodemon (wersja 2.0.16)

Fragmenty kodu:

Tabela wyświetlająca położenie robota, kolorująca backgroundColor pola na czerwono:



Zdefiniowanie funkcji „ROBOT”, deklarujemy tablicę 2-wymiarową, ustawiamy pozycję startową 5,5
Ustawiamy const chanceToMove na 0.5 (czyli 50% szans na to, że robot wykona ruch)
Ustawiamy changeDirChance na 0.51 (mamy 51% szans na to, że robot zmieni kierunek)

```
// ============================================================================================================
// Robocik sprzątajacy
// ============================================================================================================

function ROBOT(){
// Tworzymy tablicę 2-wymiarową
let coords = [[],[]]
// X i Y, ustawiamy pozycje startowe
let positionX = 5
let positionY = 5

// Ustawienie szansy na to, że robocik się ruszy przy następnym ruchu
const chanceToMove = 0.5;
// Ustawienie szansy na to, czy zmieni kierunek
const changeDirChance = 0.51;

// Deklaracja zmiennych do ruszania się robota
let moving;
let moveRight;
let moveUp;

// Ustawienie granicy po której może się poruszać robocik
function Clamp(num, min, max){
    let  min_num = Math.min(max, num)
    let value = Math.max(min , min_num)
     return value
```

Ruch robota po osi X oraz Y

```
// Funkcja wykonująca ruch w zależności od tego co wylosuje, po osi X
function MoveX(){
    if(moveRight == true){
        positionX++
    } else {
        positionX--
    }
}

// Funkcja wykonująca ruch w zależności od tego co wylosuje, po osi Y
function MoveY(){
    if(moveUp == true){
        positionY++
    } else {
        positionY--
    }
}
```

Losowanie random liczby do wykonania ruchu, kolorowanie przebytej trasy.
Ustawienie ruchu co 1000ms (1s)

```
// Wszystko co znajduje się w funkcji wykonuje się po zadanym czasie 1s (1000ms)
let pozycja;
setInterval(() => {
    // Ustawiamy kolorek startowy takis sam jak kolor tła
     if(pozycja){
         pozycja.style.backgroundColor='rgba(0, 150, 255, 0.9)'
     }
let moveMoveMove = Math.random()
let directionx = Math.random()
let directiony = Math.random()
    if(directionx > changeDirChance){
        moveRight = false;
    }else{
        moveRight=true
    }
    if(directiony > changeDirChance){
        moveUp = false;
    }else{
        moveUp=true
    }
    if(moveMoveMove > chanceToMove) {
        moving = true;
        MoveX()
        MoveY()
    }else{
        moving=false;
    }
    coords = [[Clamp(positionX, 1, 10)], [Clamp(positionY,1 ,10)]]
    let kordy_stringX = Clamp(positionX.toString(),1,10)
    let kordy_stringY = Clamp(positionY.toString(),1,10)
    let kordy_string = `${kordy_stringX}x${kordy_stringY}`
    pozycja = document.getElementById(kordy_string)
    // Ustawiamy kolorek na czerwony, aby oznaczyć ruch robota sprzątającego
    pozycja.style.backgroundColor='red'
    console.log(kordy_string)
}, 1000);
```

Ustawienie ilości sztuk sprzętu dostępnego na magazynie

```
// ======================================== Ustawianie ilości sztuk ========================================

// Wyświetlenie ilości sztuk GTX na magazynie
const Ile = document.getElementById('Ile')
const Ile_Ma_Byc = document.getElementById('Ustaw_Ile')
const Zatwiedz_Ilosc = document.getElementById('Ilościomierz')

// Ustawiamy ręcznie ilość sztuk na magazynie, a następnie klikamy magiczny guziczek
Zatwiedz_Ilosc.addEventListener('click', (e)=>{
    console.log(Ile.value, Ile_Ma_Byc.value)

    // Jeśli wincyj lub równo 0, wtedy zapisuje
    if(Ile_Ma_Byc.value >= 0){
        Ile.innerHTML = Ile_Ma_Byc.value
        localStorage.setItem('Ile',Ile_Ma_Byc.value)
    }
    // Jeśli mniej jak 0, wtedy rzuca w nas alertem
    else if(Ile_Ma_Byc.value < 0){
        alert('Wartość nie może być ujemna!')
    }
    // Jeśli cokolwiek innego, również rzuca w nas alertem
    else  {
        alert('Wprowadź prawidłowe dane!')
    }
    })
```

```
function Tester_Zera(){
    let MAGAZYN = localStorage.getItem('Ile')
    if (MAGAZYN == 0){


        // Gdy 0 na magazynie, wyłączamy pole Input
        let x = document.querySelectorAll('Input')
        x.forEach((Wylacz)=>{
            Wylacz.setAttribute('disabled','')
        })

        // Gdy 0 na magazynie, wyłączamy pole TextArea
        let y = document.querySelectorAll('TextArea')
        y.forEach((Wylacz)=>{
            Wylacz.setAttribute('disabled','')
        })

        // Gdy 0 na magazynie, wyłączamy guziczek od zamówień
        let z = document.querySelectorAll('Button')
        z.forEach((Wylacz)=>{
            Wylacz.setAttribute('disabled','')
            document.getElementById('Guzik').style.backgroundColor='red'

        })
    }
```

Ustawienie zegarka zasysającego obecny czas z systemu. Następnie tworzymy pole do ręcznego ustawiania zakresu w jakim nasz sklep z internetowym formularzem jest aktywny.

```
// ==================================== Ustawianie godziny ========================================
// Skrypt tworzący zegarek
function startTime() {
    const today = new Date();
    let h = today.getHours();
    let m = today.getMinutes();
    let s = today.getSeconds();
    m = checkTime(m);
    s = checkTime(s);
    document.getElementById('Czas').innerHTML =  h + ":" + m + ":" + s;
    setTimeout(startTime, 1000);
  }

  // Funkcja dodająca "0" przed pojedyńczą cyferką (czyli mniejsza od 10)
  function checkTime(i) {
    if (i < 10) {i = "0" + i};
    return i;
  }

// Wyświetlenie godzin otwarcia sklepu
const Otwarcie = document.getElementById('Od')
const Zamkniecie = document.getElementById('Do')
const Zatwiedz_Czas = document.getElementById('Zegareczek')

// Ustawiamy ręcznie czas, a następnie klikamy magiczny guziczek
Zatwiedz_Czas.addEventListener('click', (e)=>{
    console.log(Otwarcie.value, Zamkniecie.value)
    localStorage.setItem('Otwarcie',Otwarcie.value)
    localStorage.setItem('Zamkniecie',Zamkniecie.value)
    })
```

Zapisujemy w pamięci naszą godzinę którą ustawiliśmy powyżej, następnie przyrównujemy ją do czasu systemowego i wyświetlamy odpowiednie dla danych godzin komunikaty.

```
// Ustawienie godzin otwarcia sklepu
function Sklep_Status(){
    const Godzina_Owtarcia =  localStorage.getItem('Otwarcie')
    const Godzina_Zamkniecia =  localStorage.getItem('Zamkniecie')
    // Gdy sklep nieczynny to:
    document.getElementById('ALARM').innerHTML = `Sklep nieczynny. <br> Fomularz zostanie rozpatrzony jutro od godziny ${Godzina_Owtarcia}`

    // Pobierany systemową datę (godziny i minut, a następnie oddzielamy je ":"")
    const today = new Date();
    let h = today.getHours();
    let m = today.getMinutes();
    const godzina =  h + ":" + m

    // Gdy sklep czynny to:
    if (godzina >= Godzina_Owtarcia && godzina <= Godzina_Zamkniecia){
        document.getElementById('Krzykacz').style.backgroundColor='rgb(6, 155, 6)'
        document.getElementById('ALARM').innerHTML = `Sklep czynny do godziny: ${Godzina_Zamkniecia}`
    }
}
```