



Politechnika Wrocławska



W04ISA-SI0406P

---

## Projekt zespołowy - Time cube

---

Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Informatyczne systemy automatyki

Inteligentne systemy przemysłu 4.0

Mikołaj Oganiaczyk 264263

Mateusz Żygadło 264193

Grzegorz Hołuj 264233

Prowadzący dr inż. Radosław Idzikowski

Czwartek 13:15 - 16:15

14 czerwca 2024

# Spis treści

<b>1</b>	<b>Donkumentacja Esp</b>	<b>2</b>
1.1	Główny plik . . . . .	2
1.2	wifi_starter.c . . . . .	3
1.3	sender.c . . . . .	3
1.4	readAndDisplayTimerValue.c . . . . .	3
1.5	led.h . . . . .	4
1.6	i2cInitialize.h . . . . .	4
1.7	accelReadAndCheckPosition . . . . .	4
<b>2</b>	<b>Błędy po stronie esp</b>	<b>5</b>
<b>3</b>	<b>Zużycie baterii</b>	<b>5</b>
3.1	Start up . . . . .	5
3.2	Normalne działanie . . . . .	5
3.3	Deep Sleep . . . . .	6
<b>4</b>	<b>Dokumentacja Frontend</b>	<b>7</b>
4.1	plik settings.html . . . . .	7
4.1.1	'DOMContentLoaded' . . . . .	7
4.1.2	metoda populateSelectOptions . . . . .	7
4.2	plik index.html . . . . .	8
4.2.1	Opis skryptu z pliku index.html . . . . .	8
4.3	plik show_tasks_Manager.js . . . . .	9
4.3.1	constructor . . . . .	9
4.3.2	metoda renderTasks . . . . .	10
4.3.3	metoda initTheWebsite . . . . .	10
4.4	plik Task.js . . . . .	11
4.5	createTaskElement . . . . .	11
4.6	metoda showEditPanelMetoda . . . . .	11
4.7	plik config.js . . . . .	11
<b>5</b>	<b>Błędy po stronie frontendu</b>	<b>12</b>
<b>6</b>	<b>Donkumentacja Serwer</b>	<b>13</b>
6.1	Działanie serwera . . . . .	13
6.1.1	Plik app.js . . . . .	13
6.1.2	Plik authRoutes.js . . . . .	13
6.1.3	Plik config.js . . . . .	13
6.1.4	Plik connect.js . . . . .	13
6.1.5	Plik cube_class.js . . . . .	13
6.1.6	Plik database.js . . . . .	13
6.1.7	Plik index.js . . . . .	13
6.1.8	Plik initDB.js . . . . .	13
6.1.9	Foldery views i public . . . . .	13
6.1.10	Plik package.json . . . . .	13
6.1.11	Plik env_example . . . . .	13
6.1.12	Plik Caddyfile . . . . .	14
6.2	Tabele w bazie danych . . . . .	14

# 1 Donkumentacja Esp

## 1.1 Główny plik

**Funkcje NVS do zapisywania i odczytywania danych WiFi**

```
1 void save_wifi_credentials(const char *ssid, const char *password, const char *  
    cube_id);
```

**Kod do odczytywania danych WiFi z NVS**

```
1 bool read_wifi_credentials(char **ssid, char **password, char **cube_id);
```

**Obsługa żądania HTTP GET**

```
1 static esp_err_t hello_get_handler(httpd_req_t *req);
```

**Funkcja HTTP do pobierania danych z formularza i zapisywania ich do NVS**

```
1 static esp_err_t hello_post_handler(httpd_req_t *req);
```

**Kod do uruchamiania serwera WWW w trybie hosta WiFi**

```
1 static httpd_handle_t start_webserver(void);
```

**Zatrzymanie serwera httpd**

```
1 static esp_err_t stop_webserver(httpd_handle_t server);
```

**Kod do inicjalizacji WiFi w trybie SoftAP**

```
1 esp_err_t wifi_init_softap(void);
```

**Kod do uruchamiania serwera WWW w trybie hosta WiFi**

```
1 void start_server_wifi_host(void);
```

**Kod do inicjalizacji akcelerometru MMA8452Q**

```
1 void MMA8452Q_init(void);
```

**Kod do sprawdzania, czy pozycja ściany uległa zmianie na podstawie wartości timera**

```
1 void checkIfPositionWallHasChanged(u_int16_t timerValue);
```

**Kod do wysyłania danych pozycji i czasu na serwer**

```
1 void SendAndPositionTabToServer(char *cube_id);
```

**Funkcjado realizacji działania przycisku - wymazanie pamięci esp i restart urządzenia**

```
1 void button_press(void);
```

**ISR handler dla przycisku**

```
1 static void IRAM_ATTR button_isr_handler(void *arg);
```

**Zadanie obsługujące naciśnięcie przycisku**

```
1 static void button_task(void *arg);
```

**Funkcja obsługująca naciśnięcie przycisku do uruchomienia deep sleepu**

```
1 void switch_deep_sleep(void);
```

**ISR handler dla przycisku**

```
1 static void IRAM_ATTR switch_deep_sleep_isr_handler(void *arg);
```

## 1.2 wifi\_starter.c

**Restartowanie urządzenia**

```
1 void reboot_device(void) {  
2  
3 }
```

**Czyszczenie NVS i ponowne uruchomienie urządzenia**

```
1 void erase_nvs(void) {  
2  
3 }
```

**Handler zdarzeń dla zdarzeń Wi-Fi i IP**

```
1 static void event_handler(void *arg, esp_event_base_t event_base, int32_t event_id  
    , void *event_data) {  
2 }
```

**Funkcja do inicjalizacji Wi-Fi w trybie stacji**

```
1 void wifi_init_sta(char *user_ssid, char *user_password) {  
2  
3 }
```

**Funkcja do uruchamiania Wi-Fi z podanym SSID i hasłem**

```
1 void startWifi(char *user_ssid, char *user_password) {  
2 }
```

## 1.3 sender.c

**Handler zdarzeń klienta HTTP**

```
1  
2 esp_err_t client_event_post_handler(esp_http_client_event_handle_t evt) {  
3 }
```

**Pobranie adresu MAC i przekazanie go przez wskaźnik**

```
1 void get_mac_address(uint8_t *mac) {  
2  
3 }
```

**Funkcja do wysyłania żądania HTTP POST z bieżącą ścianą i ID**

```
1 static void send_http_post_request(int currentWall, char *id_koska) {  
2 }
```

## 1.4 readAndDisplayTimerValue.c

**Funkcja do inicjalizacji i uruchamiania GPTimera**

```
1 gptimer_handle_t timer_init(void) {  
2 }
```

#### **Funkcja do pobierania bieżącej wartości timera w milisekundach**

```
1 uint64_t getTimerValueMs(gptimer_handle_t gptimer) {  
2 }
```

#### **Funkcja do ustawiania wartości timera w milisekundach**

```
1 void setTimerValueMs(gptimer_handle_t gptimer, uint64_t value) {  
2 }
```

#### **Funkcja do resetowania wartości timera do zera**

```
1 void resetTimer(gptimer_handle_t gptimer) {  
2 }
```

### **1.5 led.h**

#### **Funkcja do inicjalizacji diody LED**

```
1 static void led_init(void) {  
2 }
```

#### **Funkcja do ustawiania koloru diody RGB LED**

```
1 static void led_setColor(color_t color) {  
2 }
```

#### **Wyłączanie diody LED**

```
1 static void led_turn_off(void) {  
2  
3 }
```

### **1.6 i2cInitialize.h**

#### **Funkcja do inicjalizacji I2C z podanymi pinami SDA i SCL**

```
1 void main_i2c_init(uint8_t MMA8452Q_SDA, uint8_t MMA8452Q_SCL);
```

#### **Funkcja do inicjalizacji komunikacji I2C z podanymi parametrami**

```
1 void init_Ic2_With_Given_Parameters(uint8_t MMA8452Q_ADDR, uint8_t  
    register_address, uint8_t bits_to_set);
```

### **1.7 accelReadAndCheckPosition**

#### **Funkcja do sprawdzania pozycji na podstawie wartości przyspieszenia**

```
1 int checkPosition(int16_t AccelX, int16_t AccelY, int16_t AccelZ);
```

#### **Funkcja do odczytywania wartości przyspieszenia z sensora**

```
1 void readAccel(int16_t *AccelX, int16_t *AccelY, int16_t *AccelZ);
```

#### **Funkcja do drukowania wartości przyspieszenia**

```
1 void printValues(int16_t AccelX, int16_t AccelY, int16_t AccelZ);
```

## 2 Błędy po stronie esp

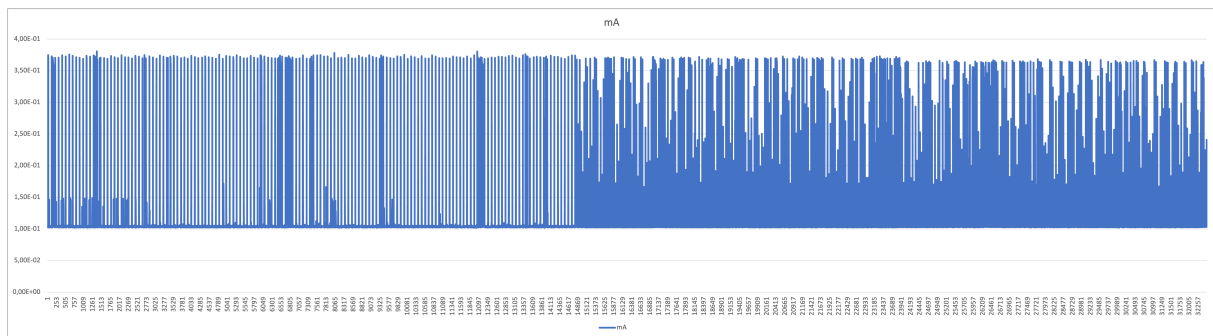
1. Dioda led nie sygnalizuje czytelnie rodzaju błędu - błędne połączenie, źle zadeklarowana biblioteka do obsługi led w kodzie by używać jej w wielu miejscach
2. Esp nie wyczytuje ze pól tekstowych serwisu web wszystkich znaków specjalnych
3. Deep Sleep nie jest prawidłowo inicjalizowany, dochodzi tylko do modem sleepu
4. Obsługa błędów przez użytkownika jest realizowana na zasadzie włączenia deep sleepu i wyłączenia go

## 3 Zużycie baterii

Dane zostały zebrane przez oscyloskop połączony z zasilaczem symulującym baterię.

### 3.1 Start up

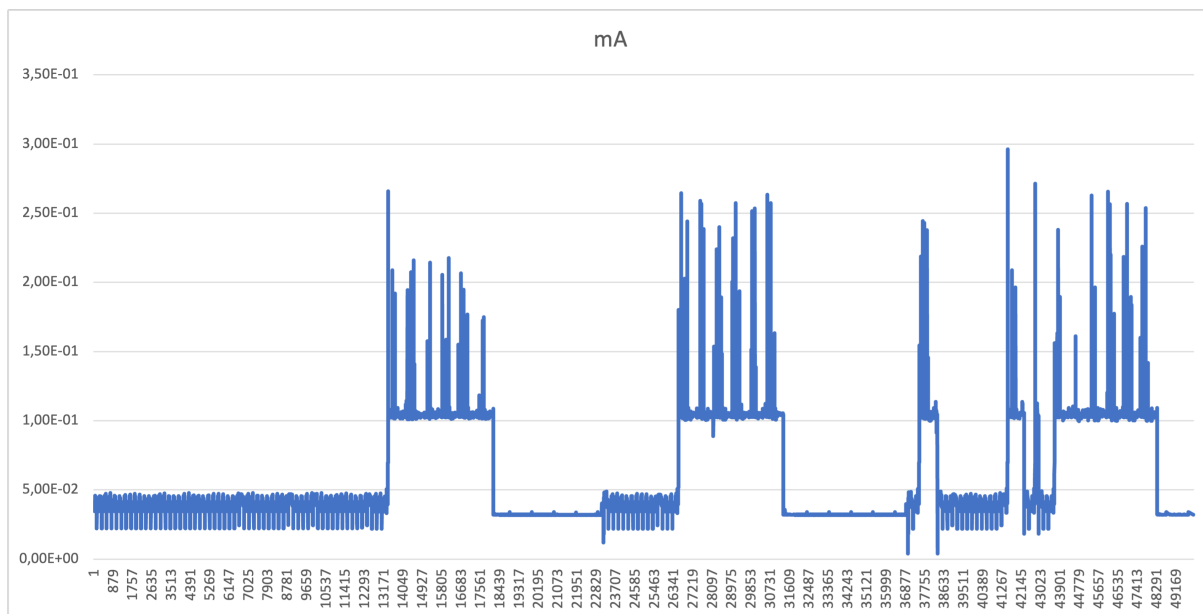
Kostka uruchamia serwer wif i stronę internetową. W 15 sekundzie telefon jest połączony i zużycie jest bardziej wypełnione. Maksymalne zużycie jest stałe



Rysunek 1: Zużycie baterii

### 3.2 Normalne działanie

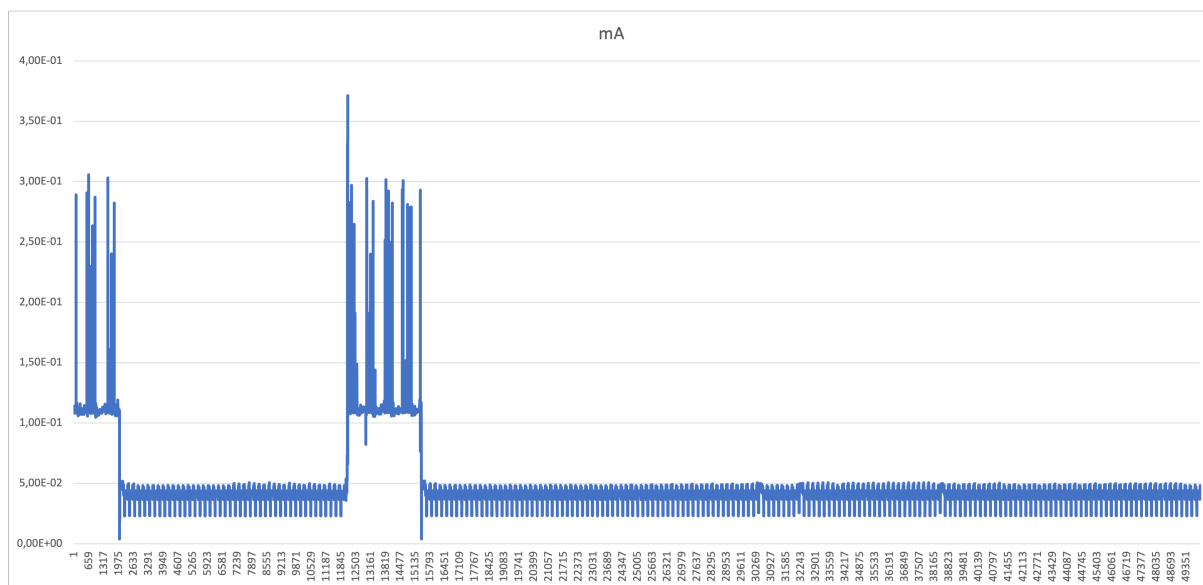
Po uruchomieniu kostka zużywa więcej prądu tylko przy pobieraniu danych z akcelerometru (co 1 sekundę), zużycie wzrasta przy wysłaniu danych oraz przy podtrzymaniu połączenia z wif lub zmianie kostki. Pobór maksymalny jest stały, minimalny jest na poziomie  $50mA$



Rysunek 2: Zużycie baterii, dane są zbierane co 1-2 milisekundy

### 3.3 Deep Sleep

Po uruchomieniu deep sleep, kostka spada do 30mA zużycia prądu, po wybudzeniu jest nagły wzrost związany z rebootem urządzenia. Po chwili wraca do normalnego zużycia i ponownie do uśpienia



Rysunek 3: Zużycie baterii, dane są zbierane co 1-2 milisekundy

## 4 Dokumentacja Frontend

link do githuba frontendu : <https://github.com/TimeCubeProject/TimeCubeProject>

### 4.1 plik settings.html

Plik settings.html jest interfejsem, który służy do dodawania i usuwania kostek. Cała funkcjonalność interakcji z kostkami jest zaimplementowana w pliku cube.js

Plik cube.js zawiera klasę Cube oraz klasę CubeList, gdzie Cube posiada pola **cube\_id**, **cube\_mac**. Klasa CubeList ma pole cubes które jest tablicą. Posiada również metody typu **addCube** i **removeCube**, które usuwają z bazy danych oraz z tablicy cubes, wybraną kostkę i jest jeszcze metoda **populateSelectOptions**, która dodaje nasłuchiwanie na zmianę wyboru adresu MAC, co powoduje aktualizację dostępnych ID kostek.

Najważniejsze funkcje w pliku cube.js

#### 4.1.1 'DOMContentLoaded'

Ta funkcja znajduje się na samym dole pliku cube.js i wywołuje się w momencie włączenia strony czyli w przypadku przełączenia na plik settings.html. Funkcja DOMContentLoaded dodaje nasłuchiwaniec na przycisk **btnAddCube** oraz **btnRemoveCube** dzięki, którym możemy dodać lub usunąć daną kostkę z bazy danych.

#### 4.1.2 metoda populateSelectOptions

Metoda populateSelectOptions wypełnia pola wyboru adresu MAC i ID kostki, bazując na dostępnych kostkach użytkownika. Na początku ta metoda pobiera macSelect, który jest elementem w select w pliku settings.html. To samo się dzieje z cubeIdSelect. W kolejnym kroku wywołujemy metodę forEach która ustawia opcje(option) w cubeIdSelect na podstawie adresów mac z **this.cubes**. Następnie jest wywoływana metoda updateCubeIdOptions, która dla wybranego adresMac pokazuje dostępne cubeId .

```
populateSelectOptions() {  
  //pobieramy mac i cube  
  const macSelect = document.getElementById( 'macSelect');  
  const cubeIdSelect = document.getElementById( 'cubeIdSelect');  
  
  // zamiana macSelect i cubeIdSelect na Select MAC Address i Select Cube ID  
  macSelect.innerHTML = '<option value="">Select MAC Address</option>';  
  cubeIdSelect.innerHTML = '<option value="">Select Cube ID</option>';  
  
  // umieszczenie w option wszystkich adresów mac danego użytkownika  
  const uniqueMacAddresses = new Set(this.cubes.map(cube => cube.Mac));  
  uniqueMacAddresses.forEach( callbackfn: mac => {  
    const option = document.createElement( tagName: 'option');  
    option.value = mac;  
    option.textContent = mac;  
    macSelect.appendChild(option);  
  });  
  
  // dodanie nasłuchiwaniec na zmianę mac  
  macSelect.addEventListener( type: 'change', listener: () => {  
    const selectedMac = macSelect.value;  
    // wywołanie funkcji która wyświetli w option cube id tylko te cube_id które sa przypisane do konkretnego maca  
    this.updateCubeIdOptions(selectedMac);  
  });  
}
```

Rysunek 4: metoda populateSelectOptions w pliku cube.js



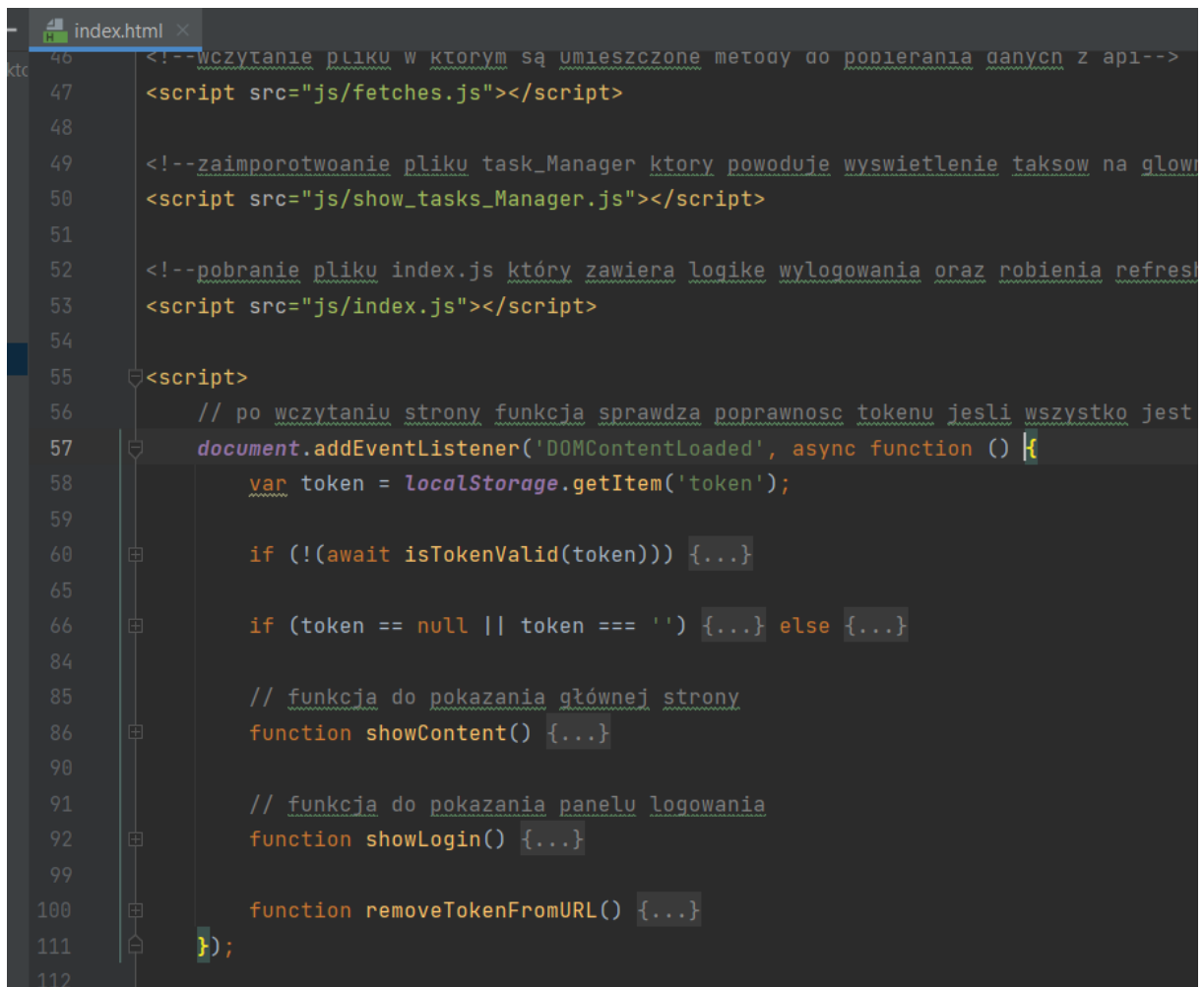
## 4.2 plik index.html

Plik index.html służy jako główny interfejs użytkownika aplikacji Task Manager, umożliwiając zarządzanie zadaniami, logowanie oraz dostęp do ustawień.

### 4.2.1 Opis skryptu z pliku index.html

Na początku po załadowaniu strony wywołuje się funkcja **isTokenValid**. Jeśli 'token' jest niepoprawny usuwa stary token z localStorage oraz usuwa token z URL i wywołuje funkcję showLogin.

Jeśli 'token' jest poprawny to pokazywany jest główny content aplikacji oraz usuwany jest token z URL.



```
46 <!--wczytanie pliku w którym są umieszczone metody do pobierania danych z api-->
47 <script src="js/fetches.js"></script>
48
49 <!--zaimportowanie pliku task_Manager który powoduje wyświetlenie taksów na głowi
50 <script src="js/show_tasks_Manager.js"></script>
51
52 <!--pobranie pliku index.js który zawiera logikę wylogowania oraz robienia refresh
53 <script src="js/index.js"></script>
54
55 <script>
56 // po wczytaniu strony funkcja sprawdza poprawność tokenu jeśli wszystko jest
57 document.addEventListener('DOMContentLoaded', async function () {
58     var token = localStorage.getItem('token');
59
60     if (!(await isTokenValid(token))) {...}
61
62     if (token == null || token === '') {...} else {...}
63
64
65
66
67
68
69
70
71 // funkcja do pokazania głównej strony
72 function showContent() {...}
73
74 // funkcja do pokazania panelu logowania
75 function showLogin() {...}
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100 function removeTokenFromURL() {...}
101
102 });
103
```

Rysunek 5: script z pliku index.html

### 4.3 plik show\_tasks\_Manager.js

Kod w pliku show\_tasks\_Manager.js zapewnia zarządzanie zadaniami, umożliwiając użytkownikom dodawanie oraz usuwanie nowych zadań, przeglądanie przypisanych i nieprzypisanych.

Opis najważniejszych funkcji/metod

#### 4.3.1 constructor

konstruktor pobiera między innymi:

- element **tasks-container** który jest divem w pliku index.html

```
<h3>Assigned Tasks</h3>
<div id="tasks-container" class="tasks-container">|
```

Rysunek 6: div id="tasks-container" class="tasks-container"

- element **unassigned-tasks-container** który też jest divem w pliku index.html

```
<div id="unassigned-tasks-container" class="tasks-container">
  <!-- Unassigned tasks content goes here -->
</div>
```

Rysunek 7: div id="unassigned-tasks-container" class="tasks-container"

- elementy typu **add-task-btn**, **load-more-assigned-btn**, **load-more-unassigned-btn** są też w pliku index.html

```
<div id="content-container" class="container">
  <button id="logout-btn" class="logout-btn">Logout</button>
  <h2>Task Manager</h2>
  <a href="settings.html" class="menu-option">Settings</a>
  <button id="add-task-btn" class="add-task-btn">Add Task</button>
  <button id="refresh-window-btn" class="refresh-window-btn">Refresh</button>

  <h3>Assigned Tasks</h3>
  <div id="tasks-container" class="tasks-container">
    <!-- Assigned tasks content goes here -->
  </div>
  <button id="load-more-assigned-btn" class="load-more-btn">Load More</button>

  <h3>Unassigned Tasks</h3>
  <div id="unassigned-tasks-container" class="tasks-container">
    <!-- Unassigned tasks content goes here -->
  </div>
  <button id="load-more-unassigned-btn" class="load-more-btn">Load More</button>
</div>
```

Rysunek 8: add-task-btn, load-more-assigned-btn, load-more-unassigned-btn w pliku index.html

#### 4.3.2 metoda renderTasks

metoda ta odpowiada za przefiltrowanie tasków z tablicy tasks na assigned i unassigned oraz dodania tasków assigned do diva **tasks-container**, a taski unassigned dodania do diva **unassigned-tasks-container**

```
// metoda która renderuje zadania na stronie głównej, dzieląc je na przypisane i nieprzypisane.
renderTasks() {
  this.tasksContainer.innerHTML = '';
  this.unassignedTasksContainer.innerHTML = '';

  // podzielenie tasków na przypisane i nie przypisane
  const assignedTasks = this.tasks.filter(task => task.Side !== -1);
  const unassignedTasks = this.tasks.filter(task => task.Side === -1);

  // pokazanie początkowo tylko 6 tasków (dotyczy tasków assigned i unassigned)
  const assignedTasksToShow = this.loadMoreAssignedClicked ? assignedTasks : assignedTasks.slice(0, 6);
  const unassignedTasksToShow = this.loadMoreUnassignedClicked ? unassignedTasks : unassignedTasks.slice(0, 6);

  assignedTasksToShow.forEach(task => {
    const taskElement = task.createTaskElement();
    this.tasksContainer.appendChild(taskElement);
  });

  unassignedTasksToShow.forEach(task => {
    const taskElement = task.createTaskElement();
    this.unassignedTasksContainer.appendChild(taskElement);
  });
}
```

Rysunek 9: renderTasks

#### 4.3.3 metoda initTheWebsite

Ta metoda pobiera taski z serwera i inicjalizuje je, następnie jest wywoływana metoda handleAddTask, która dodaje możliwość kliknięcia w przycisk i dodania taska. Oraz jest jeszcze metoda handleLoadMore która pozwala na pokazanie więcej tasków niż 6

```
// Inicjalizuje stronę, tworząc instancję TaskManager i uruchamiając główne metody.
function initTheWebsite() {
  const taskManager = new TaskManager();

  taskManager.getTheProjects();
  taskManager.handleAddTask();
  taskManager.handleLoadMore();
}

document.addEventListener('DOMContentLoaded', initTheWebsite);
```

## 4.4 plik Task.js

Plik Task.js definiuje klasę Task, która reprezentuje pojedyncze zadanie w aplikacji Task Manager.

**Najważniejsze metody**

## 4.5 createTaskElement

Metoda createTaskElement tworzy element HTML reprezentujący zadanie. Dodaje nazwę zadania, identyfikatory projektu i kostki, numer ścianki, czas, oraz przyciski do edycji, wyświetlenia historii i usunięcia zadania. Obsługuje również interakcje z przyciskami.

Na początku w tej metodzie tworzymy elementy typu i dodajemy do taskDiv. TaskDiv jest diemv, który zwraca nam danego taska i następnie ten div jest dodawany do assignedTasksShow lub unassignedTasksToShow.

- nameElement
- projectIDElement
- cubeIDElement
- sideElement
- timeElement

Każdy z tych elementów reprezentuje konkretne dane dotyczące zadania i jest dodawany do taskDiv, który reprezentuje całość zadania w interfejsie użytkownika. Dodatkowo, metoda createTaskElement dodaje przyciski do edycji, wyświetlania historii i usuwania zadania, umożliwiając pełną interakcję z zadaniem bezpośrednio z poziomu interfejsu.

## 4.6 metoda showEditPanelMetoda

Wyświetla panel edycji zadania, umożliwiającą zmianę adresu MAC oraz numeru ścianki. Obsługuje również logikę zapisu zmian do bazy danych po kliknięciu przycisku "Save".

Na początku jest tworzony panel, w którym możemy zmienić mac, cubeId oraz ściankę (wallID)

W tej funkcji jest tworzona także metoda populateCubeIdSelect, która na podstawie wybranego id kostki przypisuje konkretny adres mac

## 4.7 plik config.js

Plik config.js jest to plik w którym jest link do adresu server na który są wysyłane zapytania HTTP

## 5 Błędy po stronie frontendu

1. Po pierwsze w gdy chcemy dodać jakąś kostkę to jeśli ta kostka została dodana przez innego użytkownika to nie wyświetla się żaden komunikat. Po prostu kostka nie zostanie przypisana do innego użytkownika.
2. źle jest zaimplementowana metoda **createTaskElement** powinna podejmować decyzję o tym czy task jest unassigned na innej podstawie (Teraz jest że jeśli ścianka `=== -1`)

```
createTaskElement() {  
  const taskDiv = document.createElement( tagName: 'div');  
  taskDiv.classList.add('task');  
  
  // jeśli Side należy do -1 to zmiana task jest unassigned  
  if (this.Side === -1) {  
    taskDiv.classList.add('unassigned-task');  
  }  
}
```

Rysunek 10: Metoda create task element

Proponuje dodać kolejne pole do taska typu bool `isAssigned` na tej podstawie podejmować decyzje.

Gdybyśmy chcieli zmienić taska z `assigned` na `unAssigned` musimy użyć takiego zapytania żeby serwer nie naliczał na czasu oraz zmienić wartość pola `isAssigned`

`http://localhost:3000/set__project__active`

```
{  
  "token" : "wybrany token",  
  "project_id": "wybrany project id",  
  "cube_mac" : null,  
  "cube_id" : null,  
  "side" : "wybrana scianka"  
}
```

3. Funkcja do wyświetlania history jest myląca jeśli użytkownik ma położoną kostkę na danej ściance to historia pokaże czas rozpoczęcia zadania na podstawie czasu z serwera natomiast czas zakończenia wyświetli na podstawie lokalnego czasu danego komputera co jest mylące ponieważ dany task nie został zakończony więc w tym miejscu trzeba wyświetlić komunikat, że nadal to trwa.

W przypadku gdy patrzymy na historie danego taska i aktualna ścianka kostki nie jest taka sama jak na tasku, którego obserwujemy historię to wszystko działa poprawnie.

## 6 Donkumentacja Serwer

### 6.1 Działanie serwera

Opisy działania poszczególnych funkcji znajdują się w komentarzach kodzie.

Funkcjonalności w poszczególnych plikach:

#### 6.1.1 Plik app.js

W pliku tym znajdują się funkcję odpowiedzialne za obsługę użytkowników(logowanie, obsługa tokenów dostępu do aplikacji), zbieranie logów z serwera oraz obsługa aktywnych projektów(projekt jest uznawany za aktywny po przesłaniu przez przypisaną do niego kostkę odpowiadającego projektowi numeru ścianki)

#### 6.1.2 Plik authRoutes.js

Plik zawiera obsługę logowania google oraz konfigurację wymaganych adresów.

#### 6.1.3 Plik config.js

Plik konfiguracyjny zawiera opcje serwera, takie jak adres bazy danych, nazwa tabeli, port serwera, adres aplikacji. Wszystkie pola opisane w komentarzach w pliku.

#### 6.1.4 Plik connect.js

Plik zawiera funkcję odpowiedzialną za przygotowanie połączenia do bazy danych.

#### 6.1.5 Plik cube\_class.js

Definicja aktywnej kostki(kostka, która wysłała zapytanie z numerem ścianki innym niż -1). Każda aktywna kostka jest obiektem tej klasy. Klasa posiada metody odpowiedzialne za dodawanie czasu do projektu, obsługę zmiany ścianki oraz zakończenie liczenia czasu po przesłaniu przez kostkę ścianki -1.

#### 6.1.6 Plik database.js

Plik zawiera wszystkie funkcje przeprowadzające operację na bazie danych.

#### 6.1.7 Plik index.js

Główny plik programu, odpowiada za skonfigurowanie oraz uruchomienie serwera. W pliku tym zawarte są endpointy pozwalające na komunikację z serwerem.

#### 6.1.8 Plik initDB.js

Plik pozwalający na utworzenie wymaganych tabel w bazie danych oraz reset bazy w przypadku gdy już istnieje. Po uruchomieniu kod automatycznie tworzy wszystkie wymagane tabele oraz kolumny w bazie danych z pliku konfiguracyjnego.

#### 6.1.9 Foldery views i public

Folder views zawiera strony html w języku ejs odpowiedzialne za wyświetlanie logów z serwera oraz strony głównej z przekierowaniem do logów lub aplikacji frontendowej. Folder public zawiera grafiki do wyświetlenia na stronie startowej oraz ikonę strony internetowej serwera.

#### 6.1.10 Plik package.json

Zawiera wszystkie wykorzystywane biblioteki i pozwala ich automatyczną instalację z wykorzystaniem menadżera pakietów np npm.

#### 6.1.11 Plik env\_example

Przykład pozwalający na ustawienie wymaganych pól w pliku zmiennych środowiskowych .env.

### 6.1.12 Plik Caddyfile

Zawiera konfigurację serwera Caddy przekierowującego ruch http oraz https na port aplikacji serwerowej.

## 6.2 Tabele w bazie danych

- Users - tabela zawiera dane użytkowników. Kolumny:

- UserID
- googleID
- Email

Przechowuje dane wymagane do logowania użytkowników lub tworzenie kont jeżeli dany użytkownik loguje się po raz pierwszy.

- Cubes - zawiera dane o kostkach. Kolumny:

- CubeID
- UserID(klucz obcy <-> Users)
- Mac
- Cube\_users\_ID

Przechowuje dane o kostkach przypisanych do kont użytkowników.

- Projects - zawiera dane o projektach. Kolumny:

- ProjectID
- UserID(klucz obcy <-> Users)
- CubeID(klucz obcy <-> Cubes)
- Side
- Name
- Time

Przechowuje dane o projektach i czasach pracy nad nimi. Pozwala na przypisanie projektu do danej kostki i wybranej ścianki.

- Events zawiera zdarzenia w danym proejkcie

- EventID
- ProjectID(klucz obcy <-> Projects)
- Name
- Time

Pozwala na dodawanie zdarzeń do historii projektu takich jak rozpoczęcie pracy, zakończenie pracy nad danym projektem wraz z datą i godziną danego zdarzenia.