



РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ”

Факултет „Електротехника, електроника и автоматика“

Катедра: „Компютърни системи и технологии“

# КУРСОВА ЗАДАЧА

по дисциплината

„Информационни системи“

Изготвили:

Нерджан Имам, 223074

Деян Събев, 223072

Месру Фикрет, 223075

Специалност: КСТ

Група: 26Б

Приел:.....

/гл. ас. д-р Цветелина Кънева/

Русе

2025

# Съдържание

<b>1. Функционални изисквания.....</b>	<b>3</b>
1.1. Управление на потребители: .....	3
1.2. Управление на системата:.....	3
1.3. Мониторинг и визуализация:.....	3
1.4. Известия:.....	3
<b>2. Нефункционални изисквания.....</b>	<b>4</b>
2.1. Производителност .....	4
2.2. Сигурност .....	4
2.3. Надеждност и достъпност.....	4
2.4. Съвместимост и интеграция .....	5
2.5. Практичност .....	5
<b>3. Описание на трите основни процеса .....</b>	<b>5</b>
3.1. Автоматизирано събиране на телеметрични данни (Data Collection) .....	5
3.1.1. Диаграма на дейност (Activity) .....	6
3.2. Мониторинг и визуализация в реално време (Dashboard Monitoring).....	7
3.2.1. Диаграма на дейност (Activity) .....	8
3.3. Засичане на аварии и известяване (Alert Handling) .....	9
3.3.1. Диаграма на дейност (Activity) .....	10
<b>4. Диаграми на основните функционалности и процеси .....</b>	<b>11</b>
4.1. Диаграми на потоците от данни (ДПД, DFD).....	11
4.2. Таблици за вземане на решения .....	11
4.3. Псевдо-код на някои алгоритми.....	11
4.4. Relational схема .....	13
<b>5. Описание как системата може да се използва като:.....</b>	<b>13</b>
5.1. TPS – обработка на ежедневни транзакции .....	13
5.2. DSS – анализ на данни и справки за вземане на решения .....	13
5.3. EIS – визуализация на ключови показатели за мениджмънта .....	13
<b>6. GitHub Repository .....</b>	<b>13</b>

## **1. Функционални изисквания**

### **1.1. Управление на потребители:**

- Потребителят/Администраторът може да влиза в системата с име и парола.
- Администраторът може да променя и задава права на потребителите.
- Администраторът може да променя и задава статуса на потребителите.
- Администраторът може да променя и задава информация за потребителите.
- Системата трябва да поддържа функционалност за „Забравена парола“ и възстановяване на достъпа чрез имейл.

### **1.2. Управление на системата:**

- Администраторът регистрира нови устройства (инвертори, батерии, електромери, сензори и т.н.) чрез въвеждане на сериен номер или сканиране на QR код.
- Администраторът може да премахва устройство/а.
- Администраторът променя информация за устройство/а.
- Потребителят може да проверява свързаните устройства и тяхното им състояние.
- Администраторът променя състоянието на устройство/а.
- Администраторът може да обновява дистанционно фърмуера (firmware) на свързаните устройства.
- Системата трябва автоматично да засича дали устройството е онлайн или офлайн.

### **1.3. Мониторинг и визуализация:**

- Системата позволява на потребителите да правят извадка от данни във онлайн средата.
- Потребителят може да генерира отчет/и във Excel формат файл/а.
- Системата запазва в архив, за определено време, предишни извадки от потребителите.
- Потребителят може да прави сравнения с запазените извадки (ако има налични в системата).
- Потребителите получават информация за метеорологичните условия от системата за избран от тях период от време (във вид на диаграма).
- Изчисляване на спестени емисии CO<sub>2</sub> и финансова равностметка (на база въведена цена на тока).

### **1.4. Известия:**

- Системата изпраща съобщение за състоянието си на администратора.
- Системата изпраща съобщение за състоянието на устройствата на потребителите.

- Потребителите (включително и администратора) биват уведомявани от системата в случай на проблем.
- Потребителите изпращат съобщение към администратора в случай на проблем.
- Потребителите получават съобщение за промени в метеорологичните условия за избрания от тях период от време (при актуализация на информацията).

## 2. Нефункционални изисквания

### 2.1. Производителност

- Данните от инверторите и сензорите трябва да се визуализират в системата със закъснение не по-голямо от 5 до 30 секунди (почти реално време).
- Системата трябва да може да обработва данни от поне 100 на брой устройства едновременно, без да се забавя работата на интерфейса.
  - **Реално предложение:** За съхранение на данните от сензорите и инверторите може да използваме Time-Series база данни (като InfluxDB или Prometheus), вместо стандартна SQL база, тъй като те са оптимизирани за запис на милиони измервания в секунда.
  - **Справки за период от 1 година трябва да се генерират за под 3 секунди.**

### 2.2. Сигурност

- Всички данни между устройствата (инверторите и сензорите) и сървър, както и между потребителя и сървър, трябва да са криптирани (SSL/TLS).
  - **Реално предложение:** Комуникацията ще се криптира чрез TLS 1.3 протокол. Автентикацията на потребителите ще се извършва чрез OAuth 2.0 / OpenID Connect (за да може да се влиза и с Google/Apple акаунти), а паролите ще се хешират с алгоритъм Argon2 или bcrypt.
- Паролите трябва да се съхраняват в хеширана стойност, а не в чист текст. Препоръчително е изискване за двуфакторна автентикация (2FA) за администраторите.
- Потребител А не трябва по никакъв начин да има достъп до данните или устройствата на потребител Б.

### 2.3. Надеждност и достъпност

- Системата трябва да бъде достъпна 99.9% от времето (тъй като спирането ѝ може да скрие аварии).
- Ако връзката с интернет прекъсне, устройствата трябва да могат да запазят данните локално за поне 24 часа и да ги изпратят, когато връзката се възстанови.

- **Реално предложение:** В софтуера на комуникационния модул (logger) на място ще бъде имплементиран локален буфер (Local Buffer/Cache). При загуба на връзка, данните ще се записват на SD карта или във вътрешната памет и ще се изпратят автоматично, когато връзката се възстанови.

## 2.4. Съвместимост и интеграция

- Системата трябва да поддържа стандартни индустриални протоколи за комуникация с инвертори (напр. Modbus TCP, MQTT, JSON REST API).
  - **Реално предложение:** Ще използваме протокол MQTT (Message Queuing Telemetry Transport), който е лек и стандартен за IoT устройства. Сървърът ще има ролята на MQTT Broker (напр. чрез Mosquitto), който ще приема съобщения (publish) от инверторите и ще ги препраща към базата данни.
- Системата трябва да предоставя API за интеграция с външни системи (напр. Smart Home системи или системи на електроразпределителните дружества).

## 2.5. Практичност

- Интерфейсът трябва да е напълно функционален както на десктоп, така и на мобилни устройства (смартфони/таблети), тъй като потребителите най-често проверяват производството през телефона си.
  - **Реално предложение:** Ще използваме популярния frontend framework React с допълнителни модули като MUI и Recharts, които ни дават възможност за коректното поддръждане и пълното визуализиране на данните.
- Системата трябва да поддържа смяна на езика (Български/Английски) и валутата.

## 3. Описание на трите основни процеса

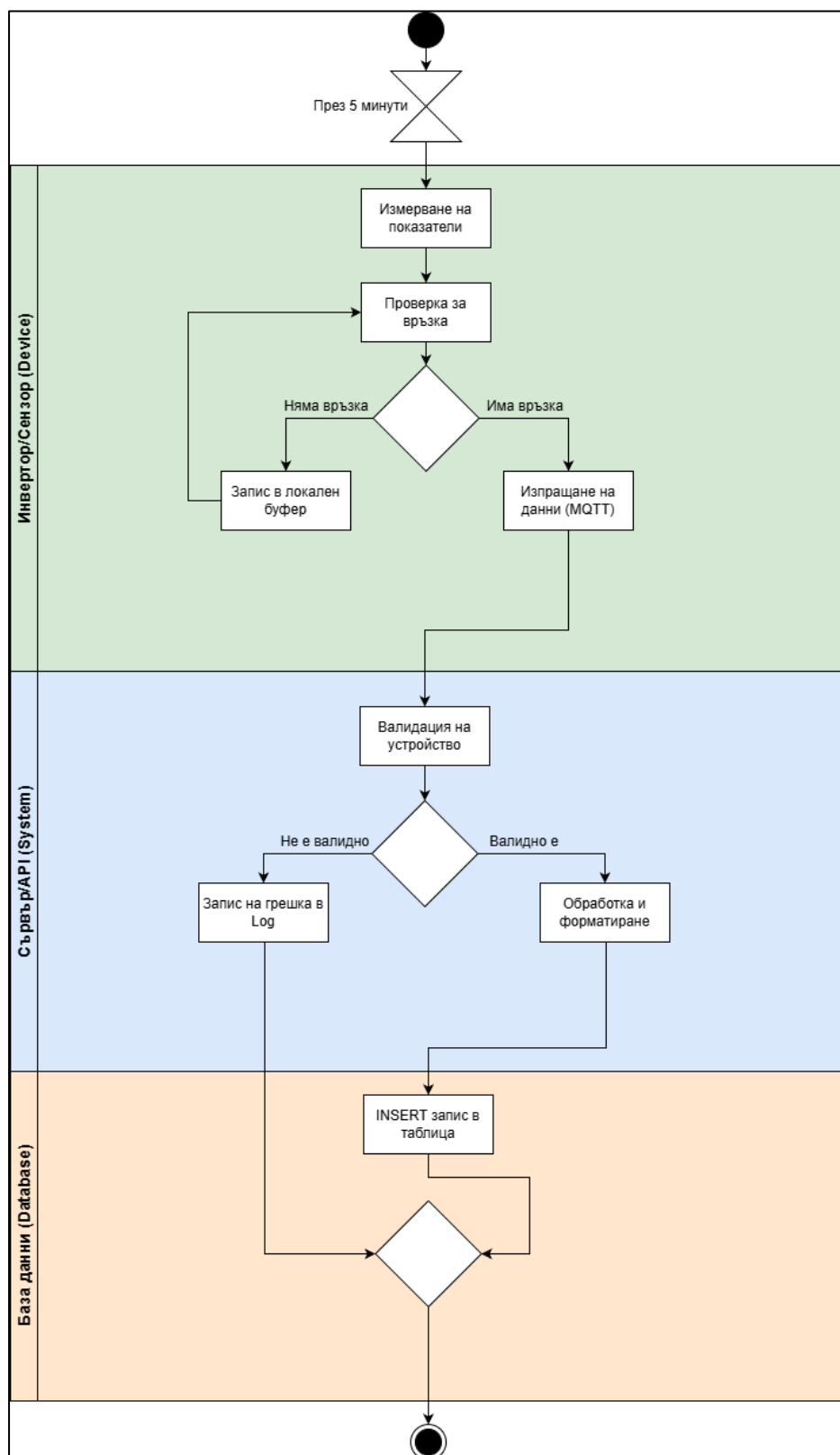
### 3.1. Автоматизирано събиране на телеметрични данни (Data Collection)

- **Участници:** Инвертор/Smart Meter, Комуникационен модул, Сървър, База данни.
- **Описание на потока:**
  1. Инверторът генерира данни за текущото производство (kW) и напрежение (V).
  2. Комуникационният модул чете данните на фиксиран интервал (напр. всеки 5 минути).
  3. Модулът криптира данните и ги изпраща към сървъра чрез MQTT или HTTP протокол.
  4. Сървърът валидира пакета (проверява дали идва от оторизирано устройство).
  5. Ако валидацията е успешна, данните се записват в базата данни със съответния времеви печат (timestamp).

6. *Алтернативен поток*: Ако няма връзка, модулът запазва данните локално (буфериране) и опитва повторно изпращане по-късно.

- **Резултат**: Актуална информация за производството е съхранена и готова за визуализация.

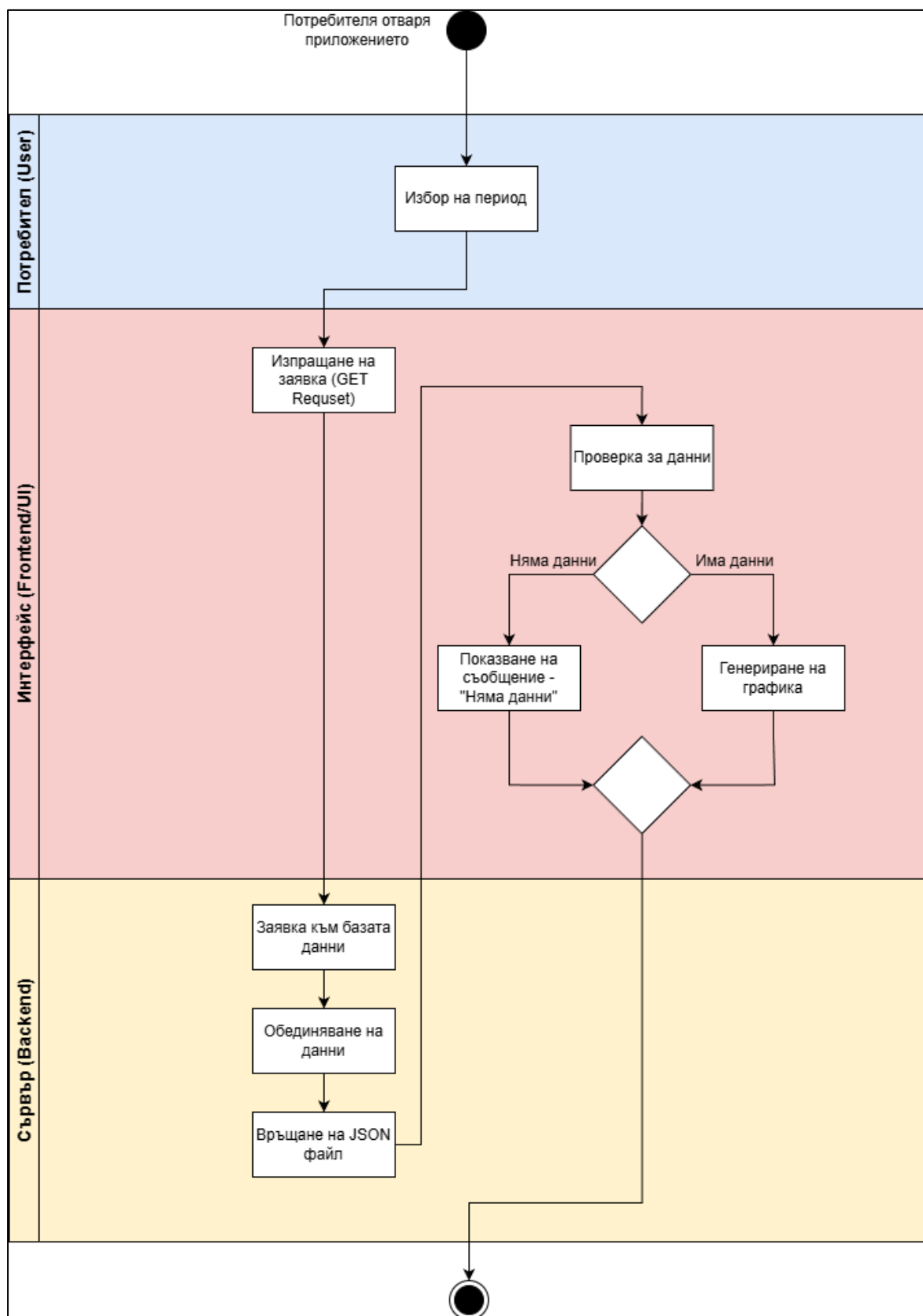
### 3.1.1. Диаграма на дейност (Activity)



### 3.2. Мониторинг и визуализация в реално време (Dashboard Monitoring)

- **Участници:** Потребител, Уеб/Мобилно приложение, Сървър.
- **Описание на потока:**
  1. Потребителят влиза в системата и отваря „Наблюдателен център“.
  2. Приложението изпраща заявка към сървъра за последните налични данни.
  3. Сървърът извлича обединена информация от базата данни (текуща мощност, дневен добив, статус на батерията).
  4. Сървърът връща данните в JSON формат.
  5. Приложението визуализира данните под формата на динамични графики.
  6. Приложението отваря WebSocket връзка за автоматично обновяване на данните на екрана, без презареждане на страницата.
- **Резултат:** Потребителят вижда моментното състояние на фотоволтаичната централа.

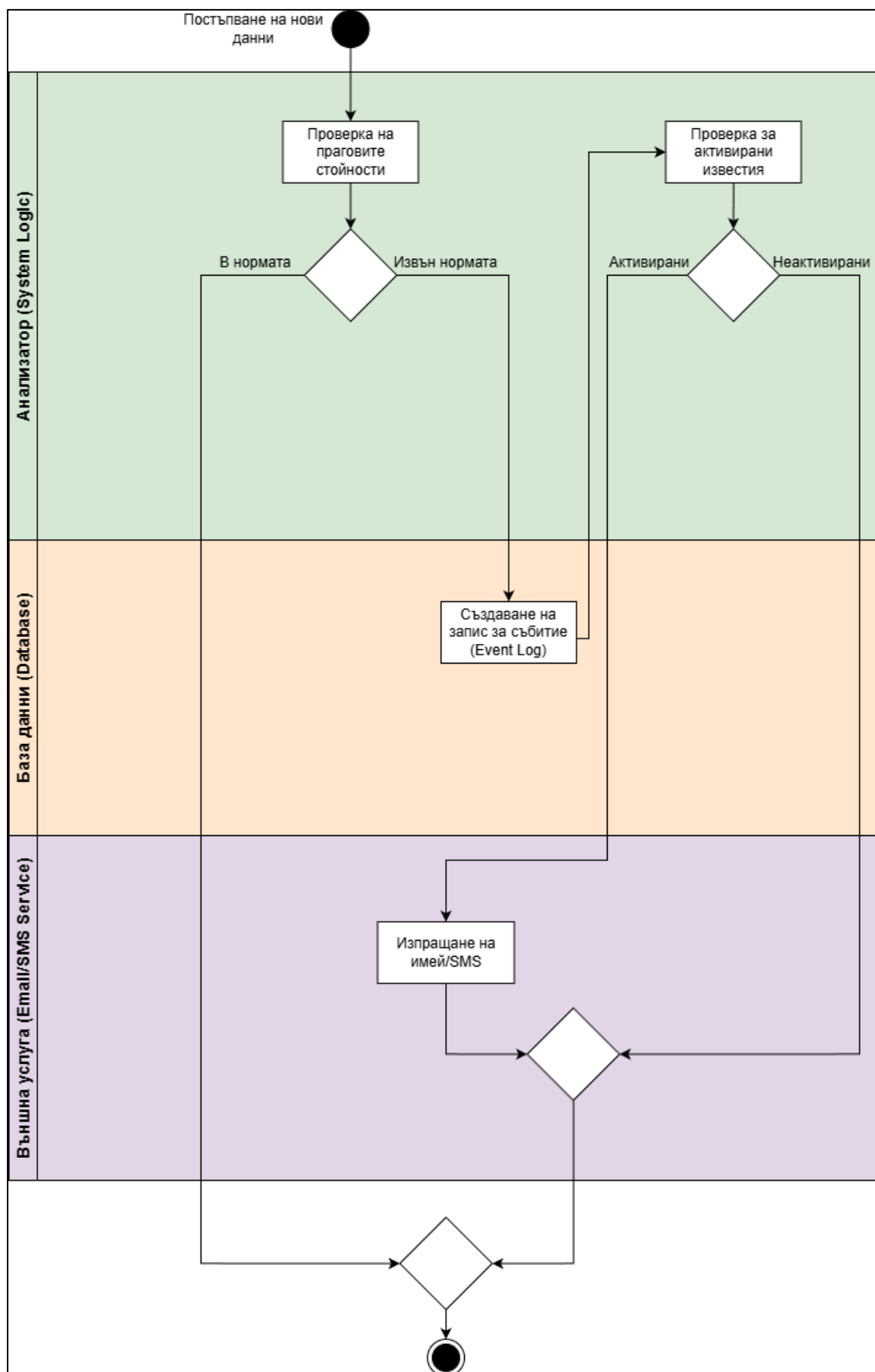
### 3.2.1. Диаграма на дейност (Activity)



### 3.3. Засичане на аварии и известяване (Alert Handling)

- **Участници:** Система (Backend logic), Администратор/Собственик, Уведомителна услуга (Email/SMS).
- **Описание на потока:**
  1. Системата постоянно анализира постъпващите данни от устройствата (мониторинг на заден фон).
  2. Системата засича аномалия (напр. „Напрежението е 0V през деня“ или „Инверторът не предава данни повече от 15 мин“).
  3. Алгоритъмът проверява настроените прагове за аларми за конкретния потребител.
  4. Системата генерира събитие „Авария“ и го записва в регистъра (Log).
  5. Ако е настроено известяване, системата изпраща Push нотификация или имейл до собственика и техника.
- **Резултат:** Определените лица са информирани за проблема мигновено, за да предприемат действия.

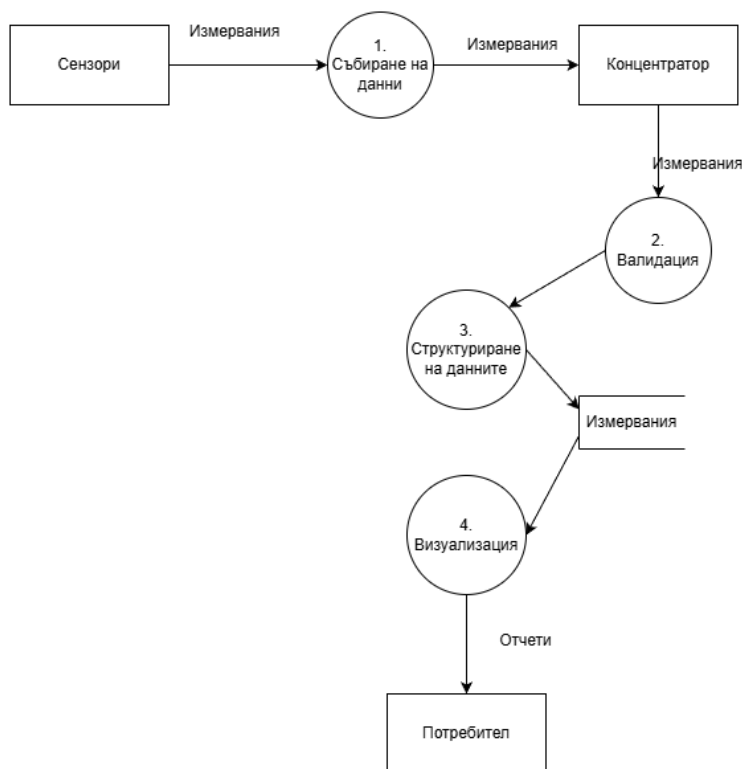
### 3.3.1. Диаграма на дейност (Activity)



## 4. Диаграми на основните функционалности и процеси

### 4.1. Диаграми на потоците от данни (ДПД, DFD)

Диаграма на потоците от данни



### 4.2. Таблицы за вземане на решения

Условия/Действия	Правило 1	Правило 2	Правило 3	Правило 4	Правило 5
Условия					
Времето е слънчево					
Времето е облачно/дъждовно					
Светлата част на деня е					
Тъмната част на деня е					
Товара се увеличава					
Товара се намалява					
Батерията е заредена					
Батерията е изтощена					
Мощността от панелите е малка					
Действия					
Обновявай базата данни по-рядко					
Превключи към централно електричество					
Превключи към фотоволтаичната система					
Зареди батерията					
Използвай мощност от батерията					

Правило 1 - Изпълнени условия (3,5,8,10) => Действия (16, 15)  
 Правило 2 - Изпълнени условия (4,16,7,9,11) => Действия (17)  
 Правило 3 - Изпълнени условия (6) => Действия (13)  
 Правило 4 - Изпълнени условия (11,4,5,10,7) => Действия (14)  
 Правило 5 - Изпълнени условия (3,5,7,10) => Действия (14)

Изпълнено условие - ■  
 Неизпълнено условие - ■  
 Изпълнение на действие - ■

### 4.3. Псевдо-код на някои алгоритми

```

// pseudo code

BOOL isSunny, isCloudy, isRainy, isDay, isNight, highLoad, lowLoad,
lowBatterLevel, highBatterLevel, lowPanelPowerOutput = FALSE

FLOAT currentTime = time.now() // get current time

FLOAT lastUpdatedTime = 0.0
  
```

```

FLOAT UpdatePeriod = 3000.0 //ms
FLOAT lowUpdatePeriod = 10000.0 // ms
WHILE(currentTime - lastUpdatedTime >= UpdatePeriod)
    IF (isSunny AND isDay AND lowLoad AND lowBatterLevel)
        SwitchToPV_Power(PV)
        ChargeBatory(PV, limit)
    END IF
    IF ((isRainy OR isCloudy) AND NOT(isNight) AND highLoad AND highBatterLevel
AND lowPanelPowerOutput)
        UseBatteryPower(PV)
    END IF
    IF (isNight)
        UpdatePeriod = lowUpdatePeriod
        UpdateDataBase(UpdatePeriod, database)
    END IF
    IF ((isRainy OR isCloudy) AND isDay AND highLoad AND lowBatterLevel AND
lowPanelPowerOutput)
        SwitchToCityPower(PV)
    END IF
    IF (isSunny AND isDay AND highLoad AND lowBatterLevel)
        SwitchToCityPower(PV)
    END IF
END WHILE

FUNCTION UseBatteryPower(PV)
PV.changePower(PV_Power)
END FUNCTION

FUNCTION ChargeBatory(PV, limit)
PV.chargeBatory(limit)
END FUNCTION

FUNCTION SwitchToPV_Power(PV)
// switch to PV power
UseBatteryPower(PV)
END FUNCTION

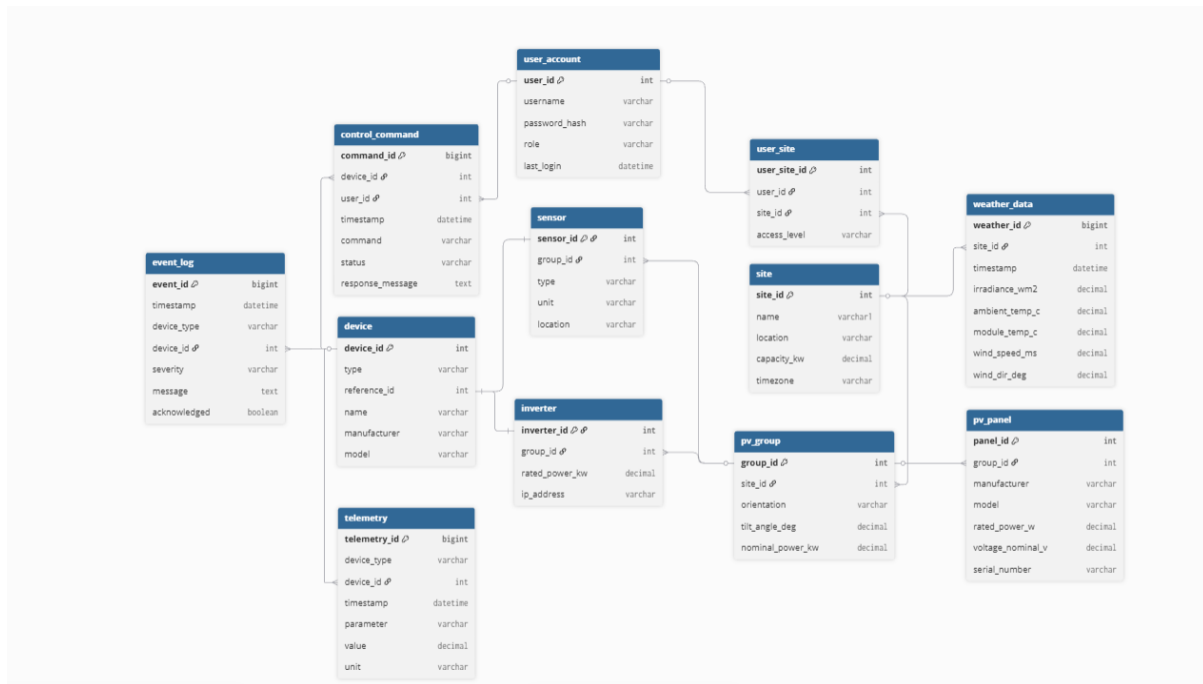
FUNCTION SwitchToCityPower(PV)
// switch to city power
PV.changePower(CityPower)

```

END FUNCTION

```
FUNCTION UpdateDataBase(period, database)
// update my database :)
database.update(period)
END FUNCTION
```

#### 4.4. Relational cхема



## 5. Описание как системата може да се използва като:

### 5.1. TPS – обработка на ежедневни транзакции

Като TPS тя събира и обработва ежедневни оперативни данни като произведена енергия, моментна мощност, напрежения, токове и алармени събития.

### 5.2. DSS – анализ на данни и справки за вземане на решения

Като DSS системата анализира натрупаните данни, генерира справки, сравнения и тенденции (напр. ефективност, загуби, отклонения от очакваното производство), които подпомагат вземането на управленски и технически решения.

### 5.3. EIS – визуализация на ключови показатели за мениджмънта

В ролята си на EIS тя предоставя визуализация на ключови показатели (KPI) чрез табла и графики, насочени към ръководството, позволявайки бърз преглед на състоянието и производителността на PV системата.

## 6. GitHub Repository

Линк: <https://github.com/Jahnnny9561/PVxSystem>