

# ENTREGABLE 9

## MÓDULO DE MEDIDAS, VERIFICACIÓN Y FACTURACIÓN

Acceso a la aplicación : <https://app.energysequence.com>

user : thd@thd.es |password : thd

### ÍNDICE

- Especificación
- Capturas de pantalla

### 1- ESPECIFICACIÓN

**Título de la especificación:** MÓDULO DE MEDIDAS, VERIFICACIÓN Y FACTURACIÓN AUTOMÁTICA

**Alcance de la especificación:**

País de aplicación: España

Elaborado por: Pit Stenzel

Responsable de la revisión: Yesnier Bravo

Fecha de la especificación:

| Versión              | Fecha      |
|----------------------|------------|
| V0 (versión inicial) | 7/01/2020  |
| V1 (revisión)        | 17/02/2020 |
|                      |            |

### 1. Especificación

List of case of uses and smartcontract methods that each of them are expected to consume accosume:

Use case 0: PPA contract management

- createPPA()
- getMyPPAs()
- getPPADetails()
- getPPATotals()

- `signPPA()`
- `linkMeterToPPA()`

Use case 1: Generated and consumed energy update

- `saveEnergy()`
- Use case 2: PPA contract monitoring alert
- `closeValidationPeriod()`

Use case 3: Energy balance recover

- `closeBalancePeriod()`

## 2. Events

The events are the way external apps can get notified about actions performed by the smartcontract. The apps can get notified in real-time or they can browse and filter events emitted from the beginning of the smartcontract deployment.

- `PPACreated(bytes32 ppald, address investor, address user)`: emitted when the PPA is successfully created
- `PPASigned(string ppald, address user)`: emitted when the user signs the PPA
- `MeterSet(bytes32 ppald, string meterId, address meterAccount)`: emitted when a meter is successfully added to a PPA
- `EnergySaved(bytes32 ppald, string meterId, address meterAccount, uint32 energyAmount, uint8 energyType)`: emitted when the energy measures are successfully aggregated into the PPA
- `ClauseViolationAlert(bytes32 ppald, string clause, uint32 delta, bytes32 periodId)`: emitted when any clause is violated. "clause" possible values:

"CONSUMPTION\_MIN", "CONSUMPTION\_MAX", "GENERATION\_MIN", "GENERATION\_MAX".

- `BalancePeriodClosed(bytes32 ppald, uint start, uint end, uint32 consumed, uint32 generated, uint32 cost, address closedBy)`: emitted when the current open period is successfully calculated, closed, registered and reseted.
- `ValidationPeriodClosed(bytes32 ppald, uint start, uint end, uint8 alertsCount, bytes32 periodId, address closedBy)`

## 3. Smart Contract public methods

### `createPPA`

*function createPPA(address user, uint32 genAnnualMin, uint32 genAnnualMax, uint32 genPriceUnder, uint32 genPriceNormal, uint32 genPriceOver, string calldata otherFields)*

### Description

It creates a new PPA and sets the address that makes the request as the investor. It also opens a new period in which the measures received from the meters will be aggregated. The properties initialized during the PPA creation are the following:

- id: auto-generated
- investor: the address that makes the request
- user: parameter
- meters cloud: parameter
- signed: false
- generated energy minimum: parameter (generatedRange[0])
- generated energy maximum: parameter (generatedRange[1])
- current period generated total: 0
- PPA's generated total: 0
- consumed energy minimum: parameter (consumedRange[0])
- consumed energy maximum: parameter (consumedRange[1])
- current period consumed total: 0
- PPA's consumed total: 0
- consumed energy price below range: parameter
- consumed energy price inside range: parameter
- consumed energy price over range: parameter
- current period start: current block timestamp
- other fields: parameter

## Restrictions

- PPA ID cannot already exist

## Parameters

- user: address. Address that identifies the user of the PPA contract.
- cloud: address. Address that identifies the related meters cloud.
- generatedRange: uint32[2]. Array with the expected range of the generated energy clause in this order: min, max.
- consumedRange: uint32[2]. Array with the expected range of the consumed energy clause in this order: min, max.
- energyPrices: uint32[3]. Array with the 3 prices of the energy in this order: below, inside and over range.

- othersFields: string. JSON that contains any other field of the PPA.

#### Returns

- Nothing

#### Events

- PPACreated

## getPPADetails

*getPPADetails(bytes32 ppald) external ppaExists(ppald)*

*onlyInvestorOrUser(ppald) view returns(address[2] memory owners, uint32[2] memory generationRange, uint32[3] memory prices, address[2] memory metersAccounts, string memory consumMeterId, string memory genMeterId, uint signed)*

### Description

It gets the details of a previously created PPA contract set during the PPA creation.

### Restrictions

- The PPA must have been previously created
- Only the owners of the PPA (the investor or the user) can call this method.

### Parameters

- ppald: String. ID of the new PPA

### Returns

- address[2] memory owners, uint32[2] memory generationRange, uint32[3] memory prices, address[2] memory metersAccounts, string memory consumMeterId, string memory genMeterId, uint signed

## getMyPPAs

*getMyPPAs() external view returns(bytes32[] memory)*

### Description

It returns a list of the PPAs associated to the address making the request.

### Restrictions

- None

## Parameters

- None

## Returns

- bytes32[]: List of PPA identifiers

## getPPATotals

*getPPATotals(bytes32 ppald) external ppaExists(ppald) onlyInvestorOrUser(ppald) view returns(uint absoluteStart, uint32 absoluteGenerated, uint32 absoluteConsumed, uint balanceStart, uint32 balanceGenerated, uint32 balanceConsumed, uint validationStart, uint32 validationGenerated, uint32 validationConsumed)*

## Description

It gets the totals of a previously created PPA contract.

## Restrictions

- The PPA must have been previously created
- Only the owners of the PPA (the investor or the user) can call this method.

## Parameters

- ppald: String. ID of the new PPA

## Returns

- absoluteStart: Start timestamp in epoch format of the absolute period
- absoluteGenerated: Amount of generated energy during the PPA life
- absoluteConsumed: Amount of consumed energy during the PPA life
- balanceStart: Start timestamp in epoch format of the current open balance period
- balanceGenerated: Amount of generated energy during the current open balance period
- balanceConsumed: Amount of consumed energy during the current open balance period
- validationStart: Start timestamp in epoch format of the current open validation period
- validationGenerated: Amount of generated energy during the current open validation period
- validationConsumed: Amount of consumed energy during the current open validation period

## signPPA

*signPPA(bytes32 ppald) external ppaExists(ppald) onlyUser(ppald)*

### Description

It sets a PPA as signed.

### Restrictions

- Only the user of the PPA can call this method.

### Parameters

- ppald: String. ID of the PPA.

Returns •Nothing

### Events

- PPASigned

## linkMeterToPPA

*linkMeterToPPA(bytes32 ppald, string calldata meterId, uint8 meterType, address meterAccount) external ppaExists(ppald) onlyInvestorOrUser(ppald)*

### Description

It links a meter to a previously created PPA

### Restrictions

- Only the owners of the PPA (the investor or the user) can call this method.

### Parameters

- ppald: String. ID of the PPA.
- meterId: String. ID of the meter.
- meterType: uint8. Type of the meter
  - 1 = Consumption
  - 2 = Generation
- meterAccount: Address. Used by the cloud

Returns • Nothing

### Events

- MeterSet

## saveEnergy

*saveEnergy(string calldata meterId, uint32 energyAmount, uint8 energyType) external*

### Description

It aggregates the given energy measure into a PPA. It increases the PPA total and the current open period total. The PPA and type of energy (consumed/generated) is deduced from the meter.

### Restrictions

- Only an address set as meters' cloud in the PPA can call this method
- The PPA must have been previously signed

### Parameters

- meterId: String. Id of the meter
- energyAmount: uint32. Amount of energy
- energyType: unit8. Type of energy to be saved
  - 1: consumption
  - 2: generation

### Returns

- Nothing

### Events

- EnergySaved

## closeBalancePeriod

*closeBalancePeriod(bytes32 ppald) external onlyInvestorOrUser(ppald) ppaSigned(ppald)*

### Description

It calculates the energy cost of the current open period, registers it in the blockchain and resets the period.

### Restrictions

- The PPA must have been previously created
- The PPA must have been previously signed
- Only the owners of the PPA (investor or user) can call this method

## Parameters

- ppald: String. ID of the PPA.

## Returns

- Nothing

## Events

- EnergyPeriodClosed

# closeValidationPeriod

closeValidationPeriod(bytes32 ppald) external onlyInvestorOrUser(ppald) ppaSigned(ppald)

## Description

It validates the clauses of the contract on demand. These clauses were defined on the contract creation. If any clause is violated a new “PPA Clause violation” event would be emitted. Clauses:

- PPA’s total generated energy of the current open period is over maximum
- PPA’s total generated energy of the current open period is below minimum
- PPA’s total consumed energy of the current open period is smaller than the generated energy.

## Restrictions

- The PPA must have been previously created
- The PPA must have been previously signed.
- Only the owners of the PPA (investor or user) can call this method.

## Parameters

- ppald: String. ID of the PPA.

## Returns

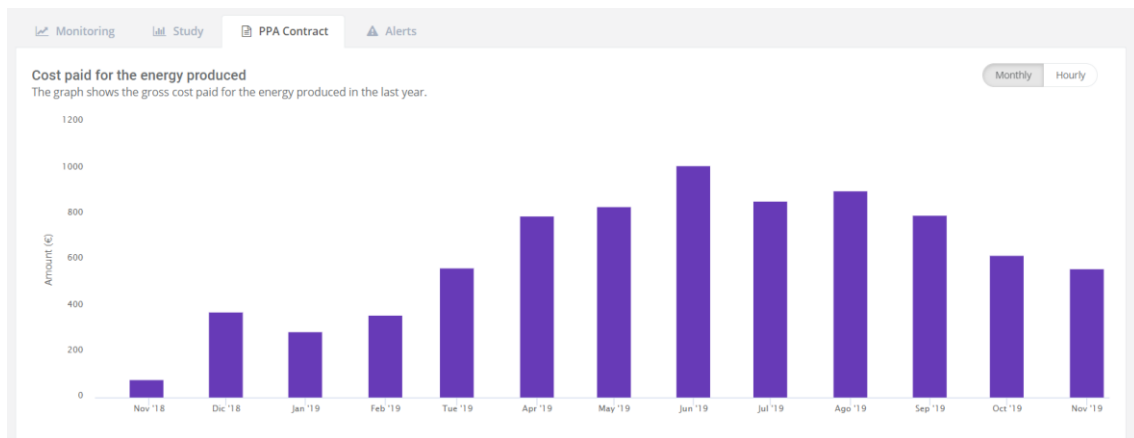
- Nothing

## Events

- ClauseViolationAlert



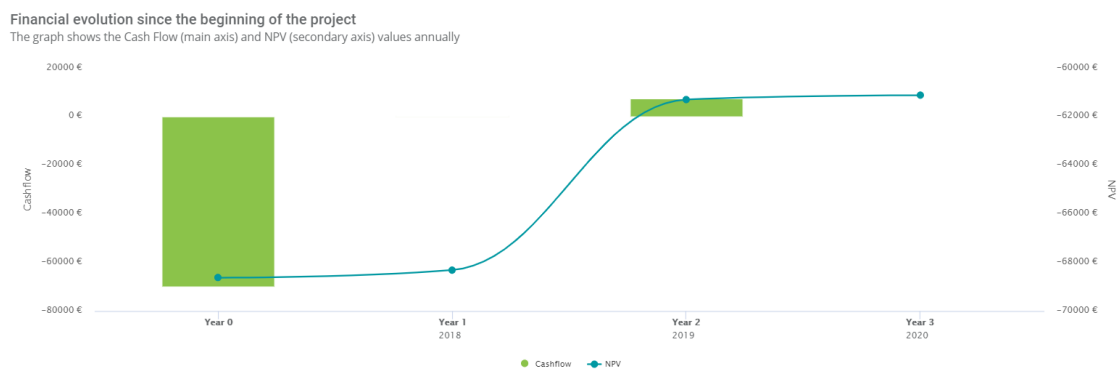
## 2-CAPTURAS DE PANTALLA



**Figura 1. Facturación diaria de costes PPA**



**Figura 2. Facturación diaria de costes PPA + Red + Vertido**



**Figura 3. Flujo de caja del contrato PPA**

**Registered meters**

- Acometida SinSabor
- Medidor PV con PPA
- Medidor PV

**Contract history**

- PPA 19-06-2018 al 19-06-2033 R

**Simulated contracts history**

No simulated contracts found for this meter

**Contract information**

**General data**

Utility: Bettergy Demo

Contract type: PPA - Power Purchase Agreement

Type of PPA: Fotovoltaico

Energy producer: Test Investor

**Contract duration**

Start date: 19/06/2018

Contract duration: 15 years

Suspensive condition: 5 meses

Liquidation Frequency: 5 meses

☐ It is a simulated contract?

**Technical details**

Installed power: 80.5 kWp

Minimum yearly generation: 128,002.0 kWh


Yearly equivalent hours: 1,590.09 kWh/kWp

**Economical**

Energy price:

Mandatory percentage of:

**SinSabor**



Partially monitored

519/519

**Meter details**

Real Invoices: 12 meses

SIPS Invoices: 0 meses

Load curves: 12 meses

**Load curves**

Descargar Datos

**Figura 4. Alta y registro del contrato PPA**