ezzing3DLayout (v2.6.0)

PV planning tool

API Documentation

ezzingsolar®

September, 2016

Table of Contents

Introduction

Ezzing 3DLayout is a PV planning tool that allows you to generate a 3d model of a building based on a satellite image. You can model any number of buildings, select between up to five different type of roofs, define keepouts and trees with custom heights...

Inside each roof area you can customize different structures, select module models and get automated previews of your setup.

It also provides you with a perspective view and a sun simulator to determine where the shadows will be in your installation.

Ezzing 3DLayout is an embebible webapp. You can integrate it inside your own system and customize many elements inside, from module models to prefered default settings for each roof type.

In this document you will find a brief showcase of the different areas of the app, a technical explanation on how to integrate this webapp inside your platform, a full description of the API that will allow you to communicate with the 3DLayout, and finally a description on how to customize different parts of the app.

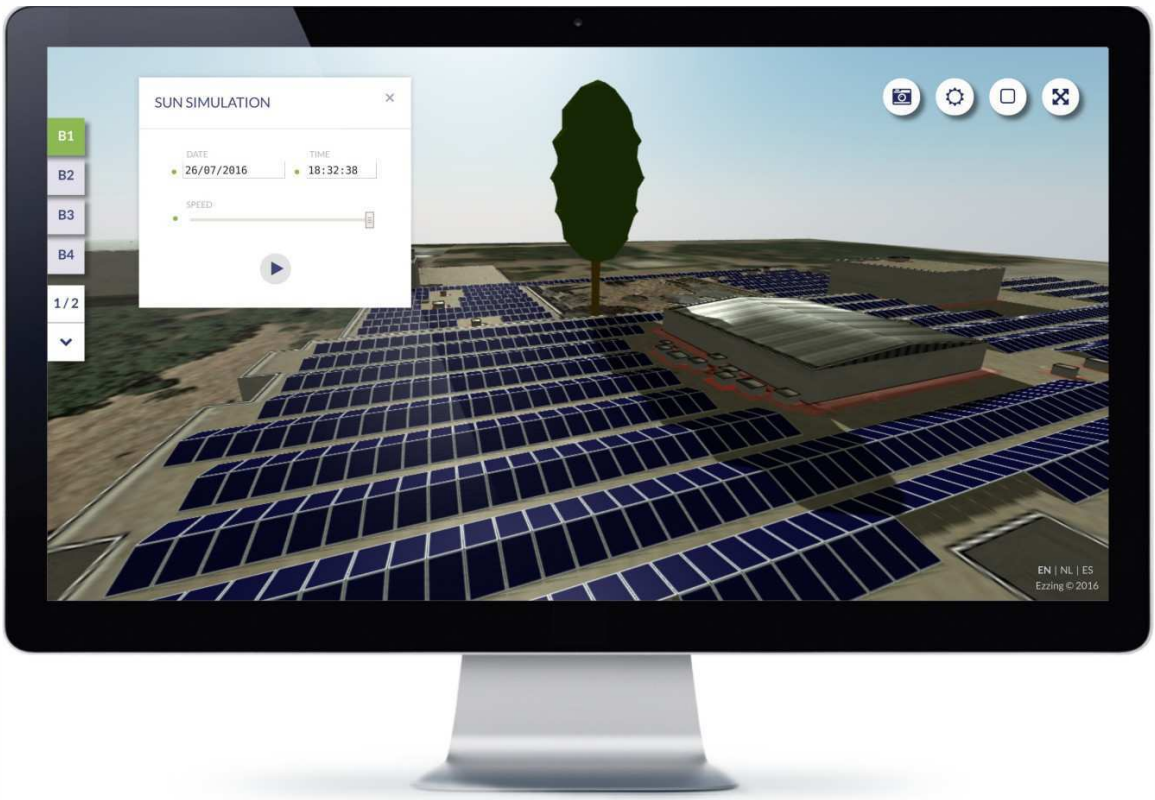
You can test the app by visiting this link:

https://layout.ezzing.com/#/demo

Also you can follow a tutorial to learn the basics of the 3DLayout in this link:

https://layout.ezzing.com/#/tutorial

Screenshots

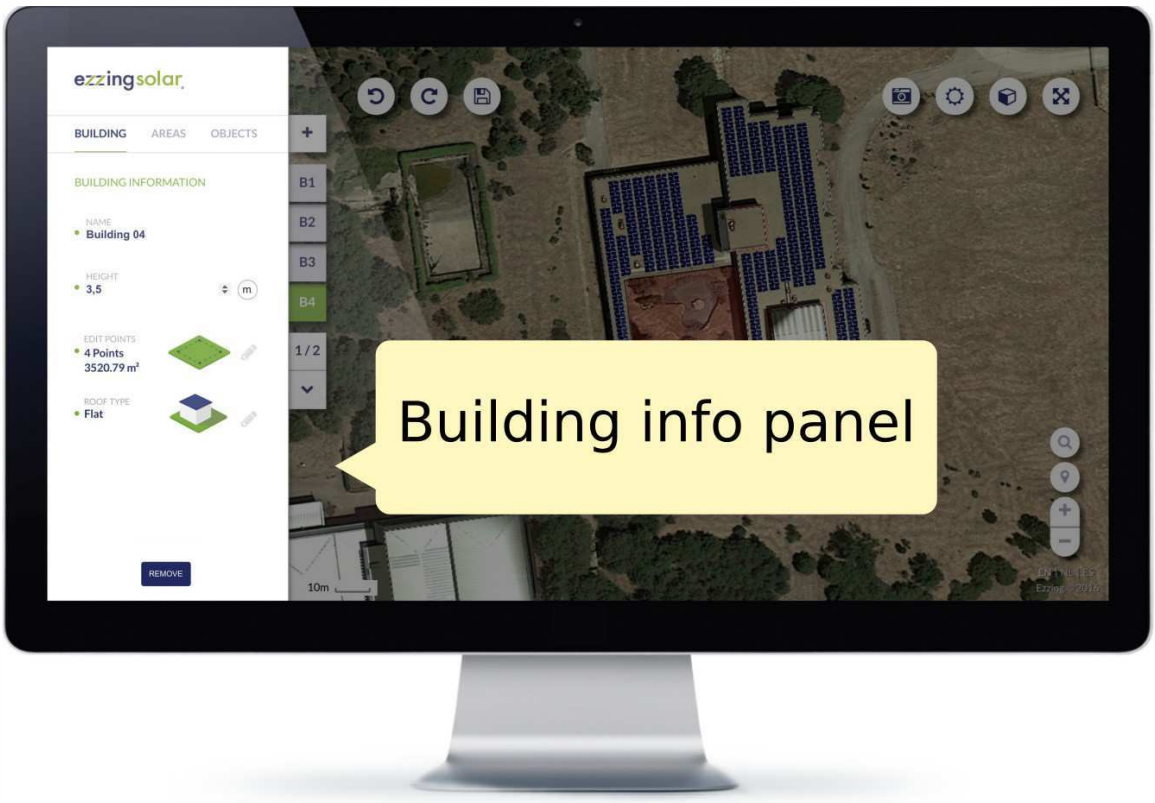


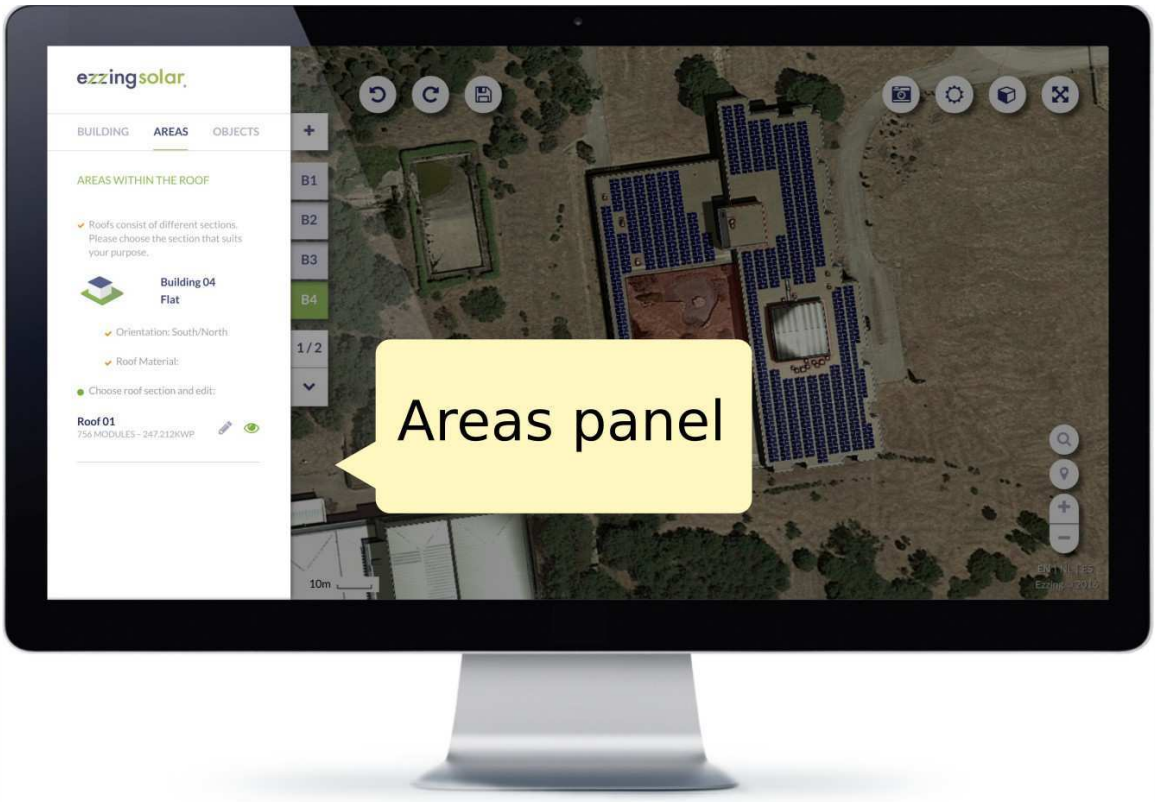
3DLayout Interface

The 3DLayout interface has two diferent parts. The **aside panel** and the **canvas area**.

Aside Panel

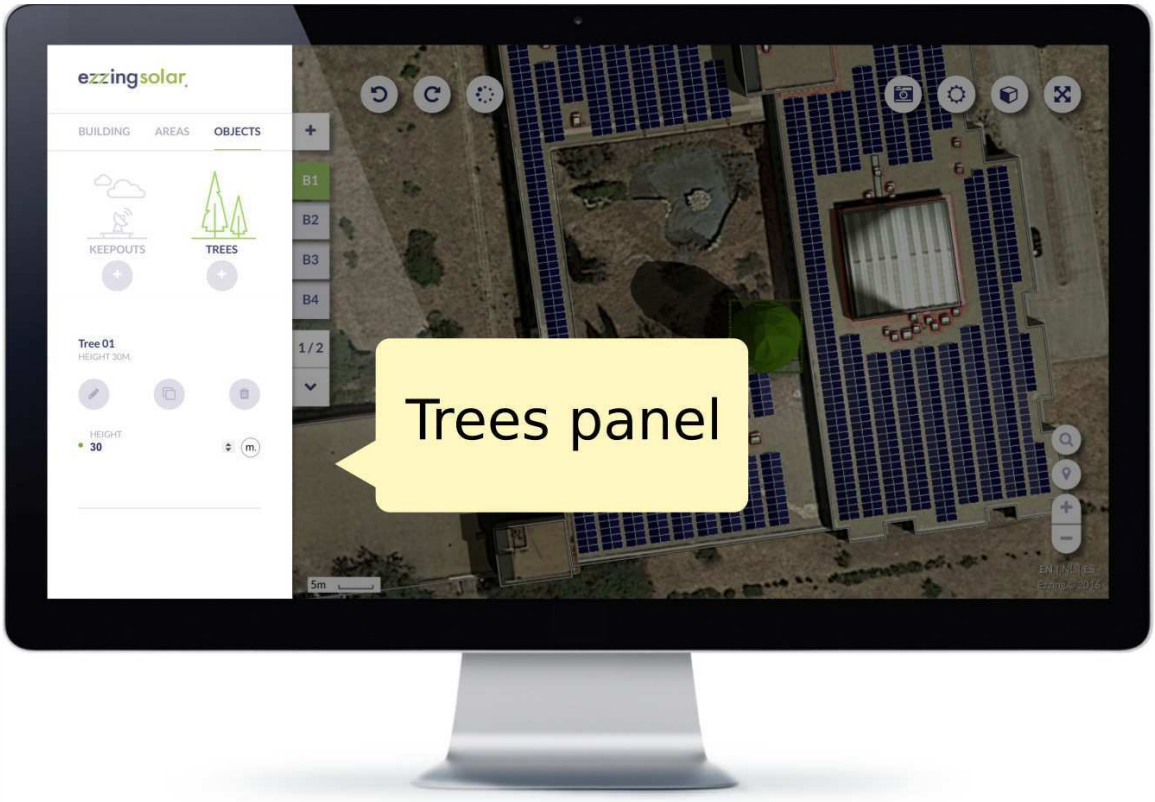
In the aside panel you can find functionalities related to the current active building and other objects in the scene.




ezzingsolar
BUILDING
AREAS
OBJECTS
AREAS WITHIN THE ROOF
Roofs consist of different sections. Please choose the section that suits your purpose.
Building 04
Flat
Orientation: South/North
Roof Material:
Choose roof section and edit:
Roof 01
756 MODULES - 247.212KWP
B1
B2
B3
B4
1 / 2
Areas panel
10m

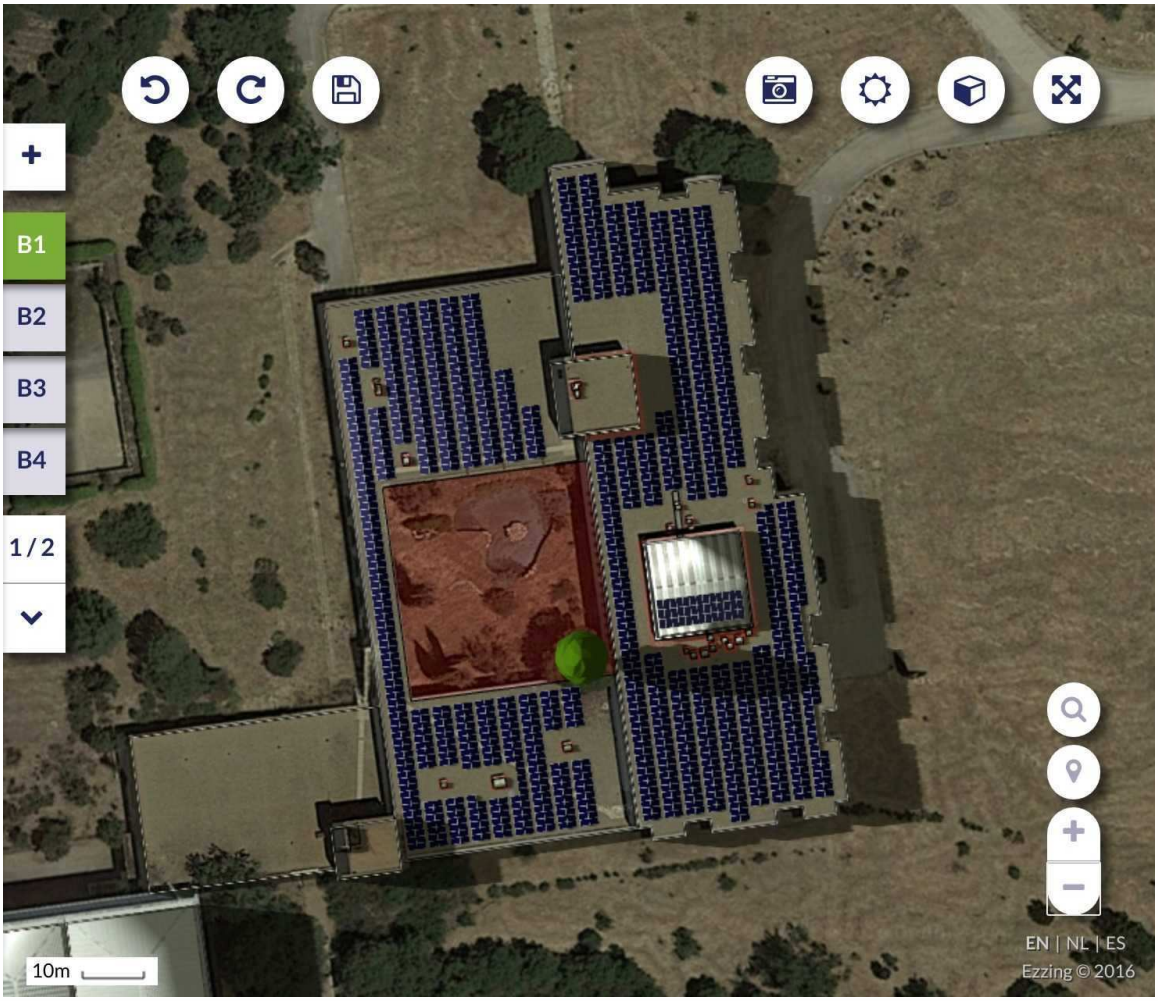

ezzingsolar
BUILDING
AREAS
OBJECTS
Roof 01
756 MODULES - 247.212KWP
MODULE MODEL
BenQ - 327Wp Mono Zwart
MODULE ORIENTATION
Portrait
MOUNTING TYPE
East/West
MODULE INCLINATION
15
SAVE
B1
B2
B3
B4
1 / 2
Area info panel
10m


ezzingsolar
BUILDING
AREAS
OBJECTS
KEEPOUTS
TREES
B1
B2
B3
B4
1 / 2
Keepout 01
KEEPOUT HEIGHT
0,02
KEEPOUT OFFSET
0
Keepout 02
10m
Keepouts panel


ezzingsolar
BUILDING
AREAS
OBJECTS
KEEPOUTS
TREES
B1
B2
B3
B4
1 / 2
Tree 01
HEIGHT
30
5m
Trees panel

Canvas Area

In the canvas area you can see the satellite view and three diferent sets of elements, the **buildings index** on the left side, the **main options buttons** on the top, and the **control buttons** on the bottom-right corner.



Buildings Index

In the buildings index you can see the active building and select another one to become active.

Main Options

These are the main options in the canvas area.

The three left-sided buttons are the **main options fixed buttons**:

- redo
- undo
- save

The right aligned buttons are the **main options custom buttons**. You can customize this set of buttons by hidding some of them or by adding new buttons.

The default custom buttons are:

- snapshot
- sun simulation

- perspective view
- fullscreen
- satellite provider selector (only showed if available)

> Please, visit the section Custom Buttons to learn how to add your own functions.

Control Buttons

These are map related buttons. You can also customize the upper section of this set of buttons by hidding some of them or by adding new buttons

Fixed buttons in this area:

- zoom in
- zoom out

Default control custom buttons

- search address
- geolocation

> Please, visit the section Custom Buttons to learn how to add your own functions.

How To Use

Api Key And Autentication

To start using the 3DLayout in your platform, you need to add the following script:

```
<script data-key="API_KEY_HERE" src="https://layout.ezzing.com/lib.js"></script>
```

where you would replace `API_KEY_HERE` by an API key we provide you for your account.

DOM Element

You need an element in the body of your html page, a div where **the 3DLayout will fit inside this element**.

> This div can't be bigger than the view size, and **no scroll** has to be applied to the html page.

You should not change the ezzing3d element size (width or height) but change the size of this container div.

```
<div style='height:100vh; width: 100vw'>
   <ezzing3d id='ezzing3d'></ezzing3d>
</div>
```

Basic Functions

Adding the `client.min.js` script with a valid API Key makes available the `Ezzing3DApi` global object, which we will use to start using the 3DLayout.

The Ezzing3DApi object has the following functions:

- createLayout
- getLayout
- listLayouts
- loadLayout

CreateLayout

Create a new layout with the specified information.

```
var data = {
    title: "EzzingSolar",
    latitude: 40.428121,
    longitude: -3.698695,
    address: "Calle de Sagasta, 18",
    zip: "28004",
    city: "Madrid",
    province: "Madrid",
    country: "Spain"
};
```

Where all values are optional except latitude and longitude that are required.

```
Ezzing3DApi.createLayout(data, function(err, layoutData) {
    if (err) throw err;
    console.log(layoutData);
});
```

Which will return the information from the created layout:

```
{
  id: 1093,
  title: "EzzingSolar",
  address: "Calle de Sagasta, 18",
  zip: "28004Madrid",
  city: "Madrid",
  province: "Madrid",
  country: "Spain",
  latitude: "40.428121",
  longitude: "-3.698695",
  created_at: "2016-08-18T17:15:15+0000",
  updated_at: "2016-08-19T10:14:34+0000",
  url: "https://layout.ezzing.com/#/GXXlgzDk0rPsrdxWfDsE5Cdi9FwUrBPx7GfuxSf0::1093"
}
```

```
{
  id: the layout id, you need this id to load the project or retrieve information,
  title: A title for the project,
  address: the address,
  zip: the zip code,
  city: the city,
  province: the province,
  country: the contry,
  latitude: latitude value in decimal degrees (remember to include the negative sign for south and west coordinates) ,
  longitude:longitude value in decimal degrees (remember to include the negative sign for south and west coordinates),
  created_at: creation date,
  updated_at: modification date,
  url: an url to visit the project or embed it as an iframe
}
```

> The url can be used to embed a readonly version of the project, please, visit bla bla section to find how to use it...

GetLayout

Returns a layout's information related to the given id

```
Ezzing3DApi.getLayout(id, function(err, layoutData) {
    if (err) throw err;
    console.log(layoutData);
});
```

ListLayouts

Returns a list of all your created layouts.

```
Ezzing3DApi.listLayouts(function(err, layoutData) {
    if (err) throw err;
    console.log(layoutData);
});
```

LoadLayout

Sets up the 3DLayout interface into the ezzing3D container and loads the project related to the given id.

```
Ezzing3DApi.loadLayout(id, function(err, layout, container) {
    if (err) throw err;
});
```

loadLayout can receive an options argument where you can setup some customizations.

> You can read a description of this methods in the Layout Rules section.

```
var rules = {};

Ezzing3DApi.loadLayout(id, rules, function(err, layout, container) {
    if (err) throw err;
});
```

This method returns two objects, where:

- layout: Exposes an object with methods to interact with the 3DLayout.

> You can read a description of this methods in the 3DLayout Communication System section.

- container: the DOM element where the 3DLayot is created.

Showcase Mode

If you want to show the layout to a customer or embed it in read-only mode in another page of your platform (to act as a thumbnail of the project) you can do it by adding an iframe element with a modified version of the url of the layout.

Showcase Without Camera Spin

```
<iframe src=(url + "/showcase")> </iframe>
```

In this mode the 3DLayout will show the project in perspective mode without any gui elements and a quiet 3d view. You can click and drag with the mouse to rotate the view and zoom with the mouse wheel.

Showcase With Camera Spin

```
<iframe src=(url + "/spin-showcase")> </iframe>
```

In this mode the 3DLayout will show the project in perspective mode without any gui elements and a rotating 3d view. You can click and drag with the mouse to rotate the view and zoom with the mouse wheel. Once clicked the rotation will stop.

3DLayout Communication System

Info Events Sent By 3DLayout

The 3DLayout trigger different events to report actions when they are accomplished or to inform on GUI changes.

An example on how to listen this events

```
var container = window.document.getElementById('ezzing3d');

container.addEventListener("buildingSelected", function(event, data){
    console.log(event.detail);
});
```

The full list of events emmited by the 3DLayout are:

- zoomChanged
- fullscreen
- tabChanged
- editArea
- editKeepout
- editTree
- areaChanged
- buildingChanged
- roofChanged
- editRoof
- editVertices
- buildingRemoved
- buildingSelected
- buildingCreated

ZoomChanged

This event is triggered when the zoom is changed in the canvas. It sends the zoom level value.

Fullscreen

This event is triggered when the user changes from normal view to fullscreen. It sends **true** when changing to fullscreen and **false** when disabling fullscreen mode

TabChanged

This event is triggerd each time the user changes the aside panel navigation tab. It sends a string with the current tab name, the values can be one of this: ["building", "areas", "keepouts", "trees"]

EditArea

This event is triggerd each time the user enters the edit section of an area. The event sends the **area.id**

EditKeepout

This event is triggerd each time the user enters the edit section of a keepout. The event sends the **keepout.id**

EditTree

This event is triggerd each time the user enters the edit section of a tree. The event sends the **tree.id**

AreaChanged

This event is triggerd each time an area attribute is changed. The event sends back an array with this info:

```
[area.id, attribute, value]
```

BuildingChanged

This event is triggerd each time a building is changed. The event sends back an array with this info:

```
[building.id, building attribute, value]
```

RoofChanged

This event is triggerd each time a roof attribute is changed. The event sends back an array with this info:

```
[building.id, roof attribute, value]
```

EditRoof

This event is triggerd each time the user enters the roof edit section of a building. The event sends the **building.id**

EditVertices

This event is triggerd each time the user enters the vertices edit section of a building. The event sends the **building.id**

BuildingRemoved

This event is triggerd each time a building is deleted. The event sends the **building.id** (after this operation this building no longer exist in the project)

BuildingSelected

This event is triggerd each time a new building becomes active. The event sends the **building.id**

BuildingCreated

This event is triggerd each time a new building is created. The event sends the **building.id**

Functions To Retrieve Info From 3DLayout

There are a set of functions to retrieve information from the 3DLayout.

For all these functions you can pass a callback as an argument to be executed when data is retrieved.

Generic Functions

Set of generic functions to retrieve project information from the layout. You just need to pass the **callback**, no other arguments are needed.

- getCurrentBuildingId
- getLayoutData
- getNumberOfModules
- getTotalPower
- getPower

GetCurrentBuildingId

```
layout.getCurrentBuildingId(callback);
```

This function returns the id value of the current active building.

GetLayoutData

```
layout.getLayoutData(callback);
```

This function returns a JSON with an array of buildings.

Each building in the array contains:

```
{
    id: the building id,
    name: the building name,
    areas: an array of areas in the building
}
```

each area in the areas array contains:

```
{
    id: the area id,
    name: the area name
}
```

GetNumberOfModules

```
layout.getNumberOfModules(callback);
```

This function returns a JSON with an array of buildings.

Each building in the array contains:

```
{
    id: the building id,
    name: the building name,
    number of modules: total of modules in the building
    areas: an array of areas in the building
}
```

Each area in the areas array contains:

```
{
    id: the area id,
    name: the area name,
    number of modules: total of modules in the area
}
```

GetTotalPower

```
layout.getTotalPower(callback);
```

Returns the total power for all the buildings in the project

GetPower

```
layout.getPower(callback);
```

Returns an array of all buildings in the project

Each building in the array contains:

```
{
    id: the building id,
    name: the building name,
    power: the total power for this building,
    areas: array of areas in this building
```

```
}
```

Each area in the areas array contains:

```
{
    id: the area id,
    name: the area name,
    power: total power in this area
}
```

Building Related Functions

Set of generic functions to retrieve building related information from the layout. In this set of functions you should pass an existing building id, and a callback.

- getBuildingInfo
- getRoofInfo
- getBuildingPosition

GetBuildingInfo

```
layout.getBuildingInfo(id, callback);
```

Returns building information for a given building.id

The data returned is:

```
{
    id: the building id,
    name: the building name,
    height: building height (in meters),
    regular: true if building angles are all equal to 90º, false otherwise.
    buildingArea: building area measure (in square meters),
    vertices: building vertices in lat/long coordinates,
    modules: total of modules in the building
    power: total power of the building,
}
```

GetRoofInfo

```
layout.getRoofInfo(id, callback);
```

Returns roof information for a given building.id]

The data returned is:

```
{
    height: roof height (in meters, not including building height),
    inclination: roof angle (in degrees),
    material: roof material (i.e: tiled/corugated),
    orientation: roof orientation (i.e: east/west or nort/south),
    type: roof type (i.e: flat, pent, gabled, etc...)
}
```

GetBuildingPosition

```
layout.getBuildingPosition(id, callback);
```

Returns building position info for a given building.id

The data returned is:

```
{
    center: the building center in lat/long coords
    vertices: an array of building vertices in lat/long coords
}
```

Area Related Functions

Set of generic functions to retrieve Area related information from the layout. In this set of functions you should pass an existing area id, and a callback.

- getAreaInfo
- getModuleInfoByArea
- getModulesSructureByArea
- getAreaOffset

GetAreaInfo

```
layout.getAreaInfo(id, callback);
```

returns area info for a given area.id]

The data returned is:

```
{
    id: the area id,
    name: the area name,
    offset: the area offset,
    placement: placement (i.e: portrait / landscape),
    structure: i.e: east-west / standard,
    inclination: modules inclination (in degrees),
    azimuth: modules azimuthal inclination (in degrees),
    areaMCoords: array containing area vertices coordinates in meters (with origin in the building center),
    areaOffsetMCoords: array containing offseted area vertices coordinates in meters (with origin in the building center) ,
    wallSizes: size in meters for each area wall,
    wallAzimuth: azimuthal angle for the external area wall,
    power: total power of the area.
}
```

GetModuleInfoByArea

```
layout.getModuleInfoByArea(id, callback);
```

returns module info for a given area.id]

The data returned is:

```
{
    id: the module id,
    name: the module model name,
    reference: extra model information,
    width: the width of the module (in meters),
```

```
    height: the height of the module (in meters),
    length: the lenght of the module (in meters),
    power: the power of the module
}
```

GetModulesSructureByArea

```
layout.getModulesSructureByArea(id, callback);
```

returns a JSON with an array of modules for a given area.id

The data for each module in the array is:

```
{
    x: x position of the module in meters (with origin in the building center),
    y: y position of the module in meters (with origin in the building center),
    col: column to which the module belongs,
    row: row to which the module belongs,,
    rX: rotation of the module in the X axis (inclination),
    rZ: rotation of the module in the Z axis (azimuth),
    color: the color of the module (only exist if color is not default),
}
```

GetAreaOffset

```
layout.getAreaOffset(id, offset, callback);
```

Returns an array of vertices containing the offseted area for a given area.id and offset

If the offset is a negative value, then the area is reduced by the offset value (in meters)

Layout Rules

The user can customize many options in the 3DLayout. By passing a 'rules' attribute to the 3DLayout instance, with a collection of objects, you can define the default values, add special behaviours to the 3DLayout and customize the interface.

Example of rules object:

```
var rules = {
    'display': true,
    'zoom': 21,
    'logo': false,
    'CustomPalette': ['#ff0000', '#00ff00']
}
```

Available rule objects expected by the 3DLayout:

- Special
  - display
  - showcase
  - zoom
  - logo
- Default Values
  - modules
  - DefaultRoofs
  - DefaultBuilding
  - CustomPalette
- Custom Buttons
  - MainoptionsCustomButtons
  - ControlCustomButtons
  - AsideCustomButtons

Special Behaviours

Display

When this options is set to true, the project starts in perspective mode.

```
{'display': true}
```

Showcase

When this options is set to true, the project starts in perspective and write-only mode. This is an special feature to showcase

```
{'showcase': true}
```

Zoom

You can set the starting zoom value. Zoom values use to range between 17 (far) to 24 (near).

```
{'zoom': 21}
```

Logo

You can choose to show (true) or hide (false) the ezzingsolar logo from the top part of the aside panel.

```
{'logo': true}
```

Default Values

User can define the default values for modules, buildings, roofs and even the color palete for modules.

Modules

User can define the solar modules available in the 3DLayout.

Sample values to define modules:

```
{
   "modules": [
      {
```

```
            "id": 4410,
            "name": "Canadian Solar",
            "reference": "250Wp Polykristallijn",
            "power": 250,
            "length": 1.638,
            "width": 0.982,
            "height": 0.04
        }, {
            "id": 41403,
            "name": "Canadian Solar",
            "reference": "260WP All Black",
            "power": 260,
            "length": 1.638,
            "width": 0.982,
            "height": 0.04
        }
    ]
}
```


The attributes for each module are:

```
{
    id: a reference number (should be unique for each model)
    name: brand and model of the module
    reference: the reference of the model
    power: power generated by the module (in watts peak)
    lenght: lenght of the module (in meters)
    width: width of the module (in meters)
    height: height of the module (in meters)
}
```

Note: the **id** value should be diferent for each module in the list.

DefaultRoofs

User can define the default values for each roof type.

The 3DLayout currently has five available roof types: **flat**, **pent**, **gabled**, **hipped** and **pyramid**.

Only roof types described in this objects are available to the user.

Sample values to define Default Roofs:

```
{
    "DefaultRoofs": {
        'gabled': {
            //roof related
            "roofMaterial": "tiled",
            "availableRoofMaterial": ["tiled", "corrugated"],
            "orientation": "EW",
            "availableorientation": ["EW", "SN"],
            "roofInclination": 30,
            //area related
            "moduleId": 47113,
            "structure": "Standard",
            "availableStructures": ["Standard", "EW"],
            "offset": 0.4,
            "inset": {
                "x": 0.02,
                "y": 0.02
            },
            "azimuth": "",
            "moduleInclination": 0,
            "moduleType": "portrait",
            "availableModuleType": ["portrait", "landscape"],
            "locked": [],
            "hidden": []
        },
        'hipped': {
            //roof related
            "roofMaterial": "tiled",
            "availableRoofMaterial": ["tiled", "corrugated"],
            "orientation": "EW",
            "availableorientation": ["EW", "SN"],
            "roofInclination": 30,
            //area related
            "moduleId": 47113,
            "structure": "Standard",
```

```
        "availableStructures": ["Standard", "EW"],
        "offset": 0.4,
        "inset": {
            "x": 0.02,
            "y": 0.02
        },
        "azimuth": "",
        "moduleInclination": 0,
        "moduleType": "portrait",
        "availableModuleType": ["portrait", "landscape"],
        "locked": [],
        "hidden": []
    },
    'pyramid': {
        //roof related
        "roofMaterial": "tiled",
        "availableRoofMaterial": ["tiled", "corrugated"],
        "orientation": "",
        "availableorientation": [],
        "roofInclination": 30,
        //area related
        "moduleId": 47113,
        "structure": "Standard",
        "availableStructures": ["Standard", "EW"],
        "offset": 0.4,
        "inset": {
            "x": 0.02,
            "y": 0.02
        },
        "azimuth": "",
        "moduleInclination": 0,
        "moduleType": "portrait",
        "availableModuleType": ["portrait", "landscape"],
        "locked": [],
        "hidden": []
    },
    'pent': {
        //roof related
```

```
        "roofMaterial": "tiled",
        "availableRoofMaterial": ["tiled", "corrugated"],
        "orientation": "S",
        "availableorientation": ["E", "W", "N", "S"],
        "roofInclination": 30,
        //area related
        "moduleId": 47113,
        "structure": "Standard",
        "availableStructures": ["Standard", "EW"],
        "offset": 0.4,
        "inset": {
            "x": 0.02,
            "y": 0.02
        },
        "azimuth": "",
        "moduleInclination": 0,
        "moduleType": "portrait",
        "availableModuleType": ["portrait", "landscape"],
        "locked": [],
        "hidden": []
    },
    'flat': {
        //roof related
        "roofMaterial": "Bitum",
        "availableRoofMaterial": ["Bitum", "Concrete", "EPDM", "PVC"],
        "orientation": "",
        "availableorientation": [],
        "roofInclination": 0,
        //area related
        "moduleId": 47113,
        "structure": "EW",
        "availableStructures": ["EW", "Standard"],
        "offset": 1,
        "inset": {
            "x": 0.02,
            "y": 0.01
        },
        "azimuth": "",
        "moduleInclination": 10,
```

```
            "availableModuleInclination": [10, 15, 20],
            "moduleType": "landscape",
            "availableModuleType": ["portrait", "landscape"],
            "locked": [],
            "hidden": []
        },
    }
}
```

Attributes explanation for default roof description

```
{
    //roof related
    "roofMaterial": should be one from the availableRoofMaterial list,
    "availableRoofMaterial": array of strings with the roof material names,
    "orientation": should be one from the availableorientation list, it can be an empty string for flat types,
    "availableorientation": array of strings with the orientation names. ['EW', 'SN'] or ["E", "W", "N", "S"]
    "roofInclination": angle for the roof (in degrees),
    //area related
    "moduleId": it is the default module model id, it is an optional attribute, if it doesn't exist the first model in the module list will be used,
    "structure": should be one from the availableStructures list,
    "availableStructures": array of strings with the structure names ["EW", "Standard"],
    "offset": the distance to avoid from area borders (in meters),
    "inset": {
        "x": the distance between module borders in the x axis (in meters) ,
        "y":  the distance between module borders in the x axis (in meters)
    },
    "azimuth": angle between the module and the north pole, it can be set as an empty string to let the 3DLayout to calculate the angle according to the external wall of the area,
    "moduleInclination": default module inclination. if availableModuleInclination is an array, this value should match one of the options in the array,
    "availableModuleInclination": this value can be an array of integers to restrict the inclination to this values. It can also be defined as an empty string to ignore the restriction,
    "moduleType": should be one from the availableModuleType list,
    "availableModuleType": array of strings with the available module types (i.e ["landscape",
```

```
 "portrait"])
    "locked": array of strings with the area attributes that should be locked,
    "hidden": array of strings with the area attributes that should be hidden
}
```

DefaultBuilding

User can define the default building. This are the default values for each new created building.

Sample values to define default building:

```
{
    "DefaultBuilding": {
        data: {
            height: 10,
            roof: {
                type: 'gabled',
                inclination: 30,
                orientation: 'EW',
                material: 'tiled'
            },
            vertices: [],
            areas: [],
            keepouts: []
        }
    }
}
```

The attributes for the default building:

```
{
    data: {
        height: default height of the building (in meters),
        roof: {
            type: roof type (one of the available types in the defaultRoofs object),
            inclination: angle for the roof (in degrees),
            orientation: orientation of the roof ridge ('EW' for east-west or 'NS' for  north-south
```

```
orientation),
            material: material of the roof (one of the available materials in the defaultRoofs
object),
        },
        vertices: []
        areas: [],
        keepouts: []
    }
}
```

CustomPalette

User can define a custom set of hexadecimal colors. This colors are used to define module colors.

To define your own color palette use this squeme:

```
{
    "CustomPalette": [
        '#242345',
        '#000000',
        '#ff0000'
    ]
}
```

Custom Buttons

There are several areas in the 3DLayout where the user can add his own functions. This areas are the **main options buttons** and the **control buttons**. Each one are defined in a JSON description that can be added to the layoutRules object.

When the user adds a new button, this is defined as a event emitter.
We use the fontawesome icons collection, so you can use it to define new buttons.

MainoptionsCustomButtons

The buttons aligned to the right are custom buttons. User can define new custom buttons in this area.

Here you can hide some existing buttoms, change the order of them and create new ones with the ability to trigger an event.

This is the default MainoptionsCustomButtons:

```
{
    "MainoptionsCustomButtons": [
        {
            click: "snapshot()",
            tooltip: "takeSnapshot",
            class: "fa fa-fw fa-camera-retro"
        }, {
            click: "widgets.simulation.toggle()",
            tooltip: "sunSimulation",
            class: "fa fa-fw fa-sun-o"
        }, {
            click: "setCamera('orthographic')",
            hide: "display.camera.name === 'orthographic'",
            tooltip: "switchTo2D",
            class: "fa fa-fw fa-square-o"
        }, {
            click: "setCamera('perspective')",
            hide: "display.camera.name === 'perspective'",
            tooltip: "switchTo3D",
            class: "fa fa-fw fa-cube"
        },{
            click: "setScreen('full')",
            hide: "fullScreen",
            tooltip: "switch to full screen",
            class: "fa fa-fw fa-arrows-alt"
        }, {
            click: "setScreen('normal')",
            hide: "!fullScreen",
            tooltip: "cancel full screen",
            class: "fa fa-fw fa-desktop"
        }, {
```

```
            click: "setProvider(0)",
            hide: "map.provider === 0 || map.provider === 1",
            tooltip: "cyclomedia",
            class: "fa fa-fw fa-map"
        }, {
            click: "setProvider(1)",
            hide: "map.provider === 1 || map.provider === 2",
            tooltip: "grid",
            class: "fa fa-fw fa-th"
        }, {
            click: "setProvider(2)",
            hide: "map.provider === 2 || map.provider === 0",
            tooltip: "google",
            class: "fa fa-fw fa-google"
        }
    ]
}
```

To customize this buttoms you can comment out the lines of the button you want to hide or extend the list by adding new objects with the same structure

```
{
    click: a string with the event name you want to trigger,
    hide: a condition to hide the button (this value is optional),
    tooltip: a string with the operator description, to show as a tooltip,
    class: a fontawesome class to define the icon
}
```

ControlCustomButtons

The buttons over the zoom in and zoom out can be customized the same way the mainoptions custom buttons.

This is the default ControlCustomButtons:

```
{
   "ControlCustomButtons": [
      {
         click: 'widgets.geocoding.toggle()',
         tooltip: 'searchAddress',
         class: 'fa fa-fw fa-search'
      },
      {
         click: '!geolocating && geolocate()',
         hide: 'geolocating',
         tooltip: 'geolocation',
         class: 'fa fa-fw fa-map-marker'
      },
      {
         click: '!geolocating && geolocate()',
         hide: '!geolocating',
         tooltip: 'geolocation',
         class: 'fa fa-fw fa-spinner ng-class:{"fa-spin": geolocating}'
      }
   ]
}
```

To customize this buttoms you can comment out the lines of the button you want to hide or extend the list by adding new objects with the same structure

```
{
    click: a string with the event name you want to trigger,
    hide: a condition to hide the button (this value is optional),
    tooltip: a string with the operator description, to show as a tooltip,
    class: a fontawesome class to define the icon
}
```

Changelog

[2.6.0] - 2016-08-29

Added

Some minor GUI changes to clarify the workflow.

Two new showcase options, with and without camera spining.

Changed

Navigation panel, save button and building index now becomes disabled when creating a building.

Undo, redo and save buttons in mainoptions becomes custom buttons.

A helper message is showed when roof types are restricted due to irregular buildings. Also, restricted roof types are shown disabled.

A dialog window is showed when vertices or roof is edited, to alert the user about the reset of the building.

Fix style for loading animation to adjust to window size.

Area offset now is called **Edge Zone**

Fixed

Some errata in the pdf (added measure units for all values)

ezzingsolar®