Administration Panel

Functional Requirements

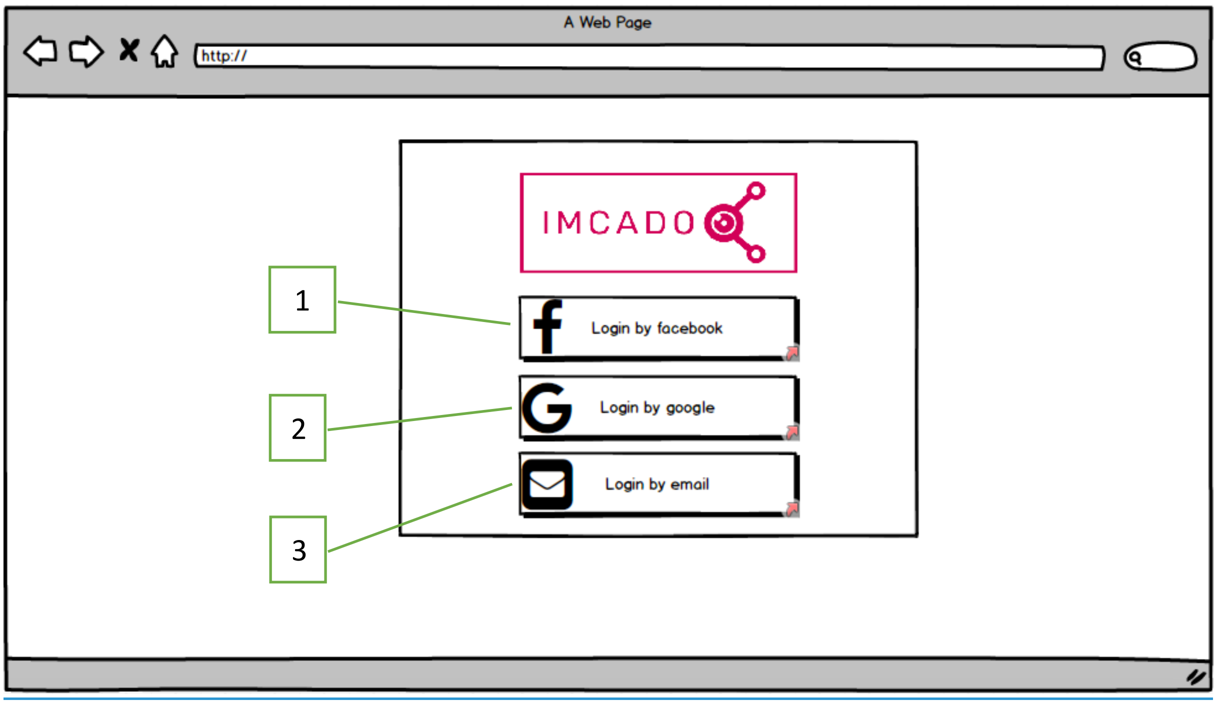
Version	Author	Date	Changes
1.0	Matt	2.12.2019	Initial version
1.1	Matt	5.12.2019	Fixed found issues

Spis treści

1. Technical requirements

- Use react.js (latest major version)
- Use web design and development best-practices
- Use redux in react
- Eslint Airbnb validation passed - eslint-config-airbnb
- Html validation passing - https://validator.w3.org
- Google page speed over 90 - https://developers.google.com/speed/pagespeed/insights/
- Responsive design
- I18N usage for all labels (see chapter 17)
- Date and time format should be defined in I18N file
- User permissions done on bit flags (see chapter **Błąd! Nie można odnaleźć źródła odwołania.**)
- Use optimistic UI update (https://medium.com/@_erikaybar/optimistic-ui-updates-in-react-9e139ffa2e45)
- Use appropriate feedback progress widgets when waiting for Azure backend to respond
- When an icon is used for button use a tooltip to describe the function
- Use metronic react theme - https://keenthemes.com/metronic/preview/react/demo1/dashboard - it will be provided
- Environment depending settings to configure:
    - URL to endpoints: slideshow_url, backend_url
    - Google authentication clientId: ios_client_id, android_client_id, web_client_id

2. Login screen



a. Login by facebook

When the user clicks button 1 show the Facebook Single Sign On (SSO) screen. After the user authenticates himself send a request with the facebook authentication token to Azure backend under endpoint ".auth/login/facebook". The Azure backend responses with an authentication token with a *zumo* token inside. Use this token for all further backend communication in the header of all requests as *"x-zumo-auth"*.

b. Login by google

When the user clicks button two show the google SSO screen. After the user authenticates himself send a request with the google authentication token to azure backend under endpoint *".auth/login/google*". The azure backend responses with an authentication token with a *zumo* token inside. Use this token for all further backend communication in the header of the requests as "x-zumo-auth".

c. Login by Azure AD B2C

When the user clicks button 3 show the azure AD B2C SSO screen. After the user authenticates himself send a request with the azure authentication token to azure backend under endpoint ".auth/login/aad". The Azure backend responses with an authentication token with a *zumo* token inside. Use this token for all further backend communication in the header of the requests as "x-zumo-auth".

d. Anonymous user

Anonymous user has access only to login page. All other pages are not avaiable for him.

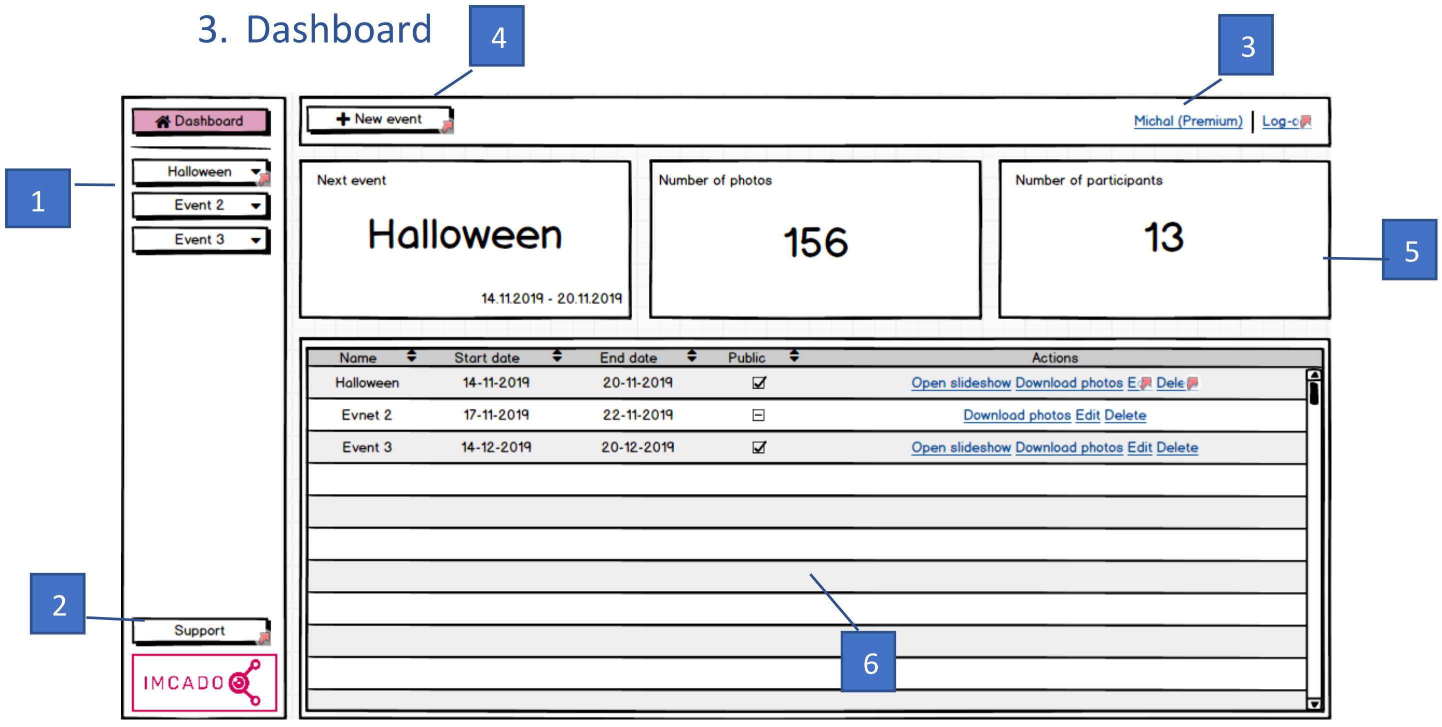
For options a, b and c use the SSO provider to get the following data:

- User display name
- Email
- Profile picture link

This data is sent to Azure backend under “{backend_url}/users”. Add the determined language to this request (see chapter 17).

After login the user is shown data collection consents. Display the privacy policy found under URL which is taken from the language file. Considering the language from the used browser. User can proceed only when agree policy.

3. Dashboard



Name	Start date	End date	Public	Actions
Halloween	14-11-2019	20-11-2019	☑	Open slideshow Download photos E Dele
Evnet 2	17-11-2019	22-11-2019	⊟	Download photos Edit Delete
Event 3	14-12-2019	20-12-2019	☑	Open slideshow Download photos Edit Delete

a. Data source

Data for this screen is to be found on Azure backend under “{backend_url}/events”. Then search in the received list of events for the closest to begin and receive summary information you need by calling separate endpoint - “{backend_url}/event/${eventId}/”.

b. Sidebar

The sidebar is visible always.

Section nr 1 contains:

- *Dashboard* button to navigate to the Dashboard screen
- all events the user has as expandable submenus that contain the buttons *Details* (see chapter 7), Participants (see chapter 10) and Photos (see chapter 14).
  - when there is a lot of events scroll bar should appear; it should be visible only when the number of an event is bigger to show on the screen.

Sections nr 2 contain a *Support* button that opens the Support site that includes information on the current plan with subscription dates and an email address to the support team (see chapter 15).

c. Top menu

Section nr 3 and nr 4 in the top menu is visible always.
Section nr 3 contains the following data:

- Name and information about a subscription is received from azure backend "{backend_url}/users/current"
- Log out button to log out the user and navigate to the *login page*

Section nr 4 contains a *New event* button and opens the *New event modal* (see chapter 4).

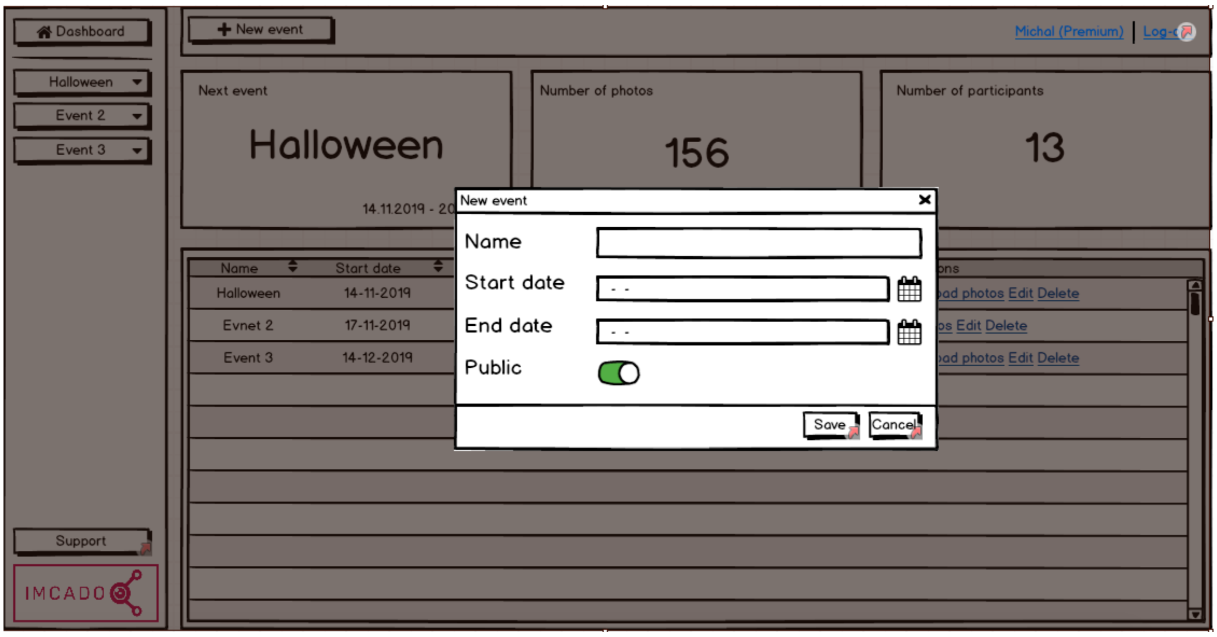
d. Dashboard content

Section 5 contains data for the event that is closest to begin. The first block contains the name and start and end date of the event. The second block contains the number of photos in the event. The third block the number of participants on this event.

Section nr 6 contains a list of all events the user has. It can be ordered ascending and descending on a name, start and end date and public. The initial order is provided in the response.
Each line contains:

- Event name
- Event start date
- Event end date
- Event is public
- Actions that can be done on event:
  - Open slideshow – only visible if the event is public – opens in new window link to slideshow {slideshow_url}/${eventId})
  - Download photos – only visible if the user uses a paid plan – triggers the azure backend API "{backend_url}/event/${eventId}/photos/archive" and shows modal to the user that the link to the archive will be sent per mail
  - Edit – open Edit event modal (see chapter 5)
  - Delete – open Delete confirmation modal (see chapter 6)

4. New event



Halloween

14.11.2019 - 20

156

13

Name	Start date	ons
Halloween	14-11-2019	oad photos Edit Delete
Evnet 2	17-11-2019	os Edit Delete
Event 3	14-12-2019	oad photos Edit Delete

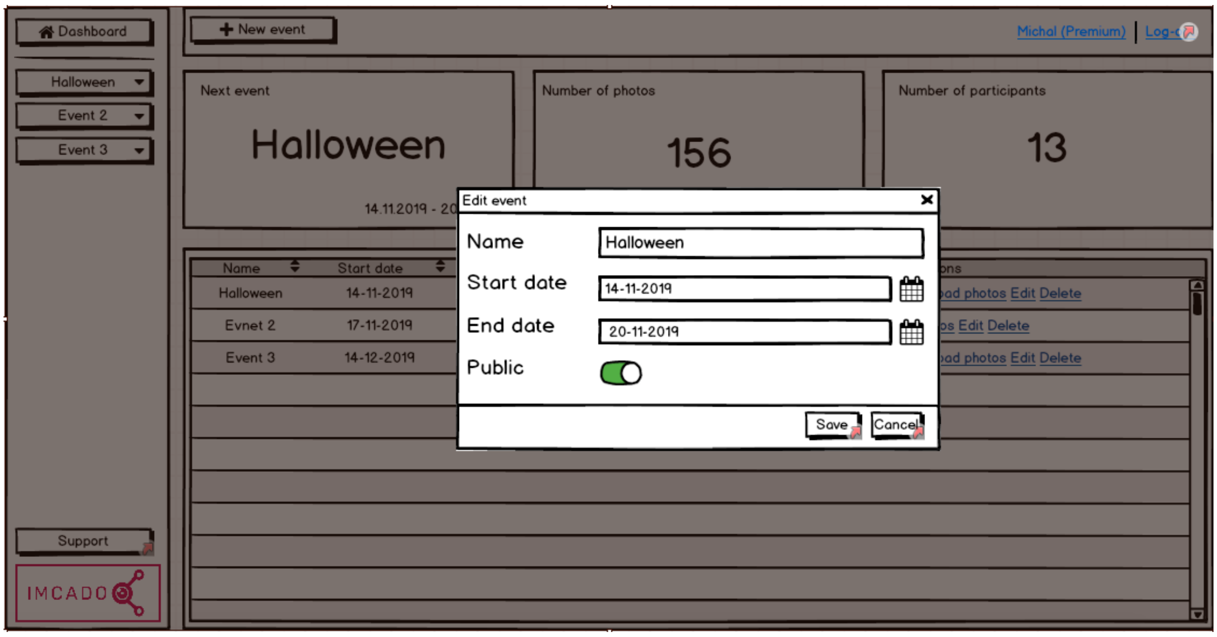
Name	
Start date	- -
End date	- -
Public	✓

The screen should work like a modal and overlap the whole view.
Data validation:

- The name should not be empty
- Initially preset the start date as now and the end date now plus one day
- Start date and end date selected by a date and time picker
- If the user selects a start date after the end date, then set the end date to start date
- If the user selects an end date before the start date, then set the start date to end date
- The start date can be the same as the end date

When the user click saves trigger the azure backend “{backend_url}/events/”.

5. Edit event



Halloween

14.11.2019 - 20

156

13

Name	Start date	ons
Halloween	14-11-2019	oad photos Edit Delete
Evnet 2	17-11-2019	os Edit Delete
Event 3	14-12-2019	oad photos Edit Delete

Name	Halloween
Start date	14-11-2019
End date	20-11-2019
Public	✓

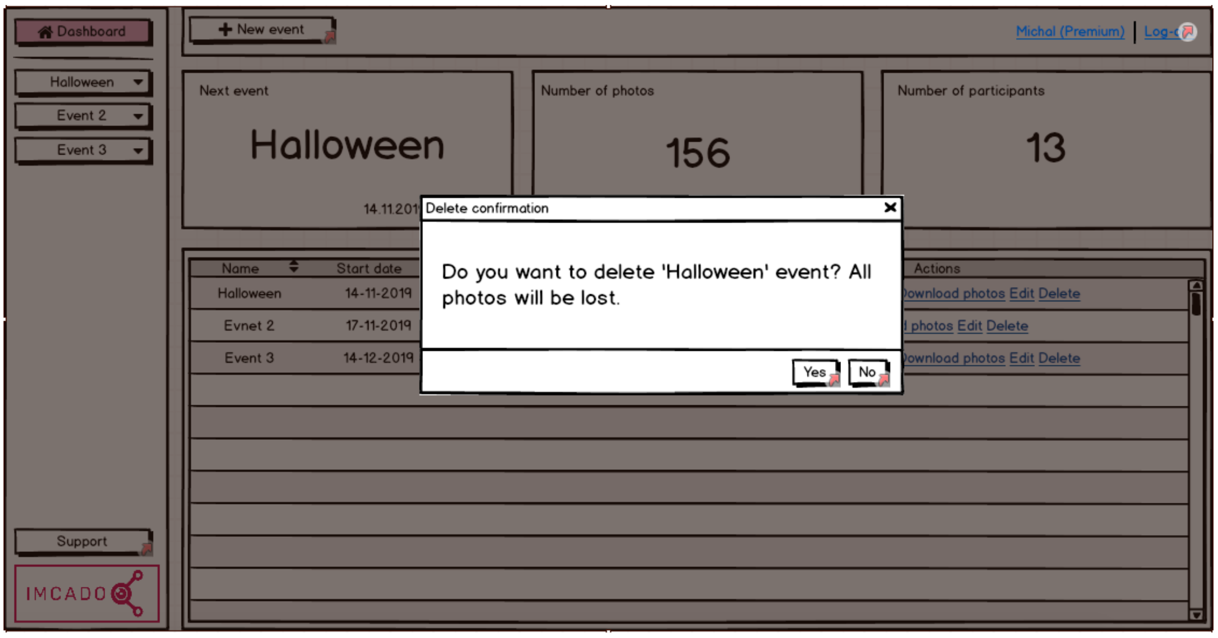
The screen should work like a modal and overlap the whole view. Use the data that was already loaded for the event.

Data validation:

- Name should not be empty
- Start date and end date selected by a date and time picker
- If the user selects a start date after the end date then set the end date to start date
- If the user selects an end date before the start date then set the start date to end date
- The start date can be exactly the same as the end date

When the user click save trigger the azure backend “{backend_url}/event/${eventId}”.

6. Delete event



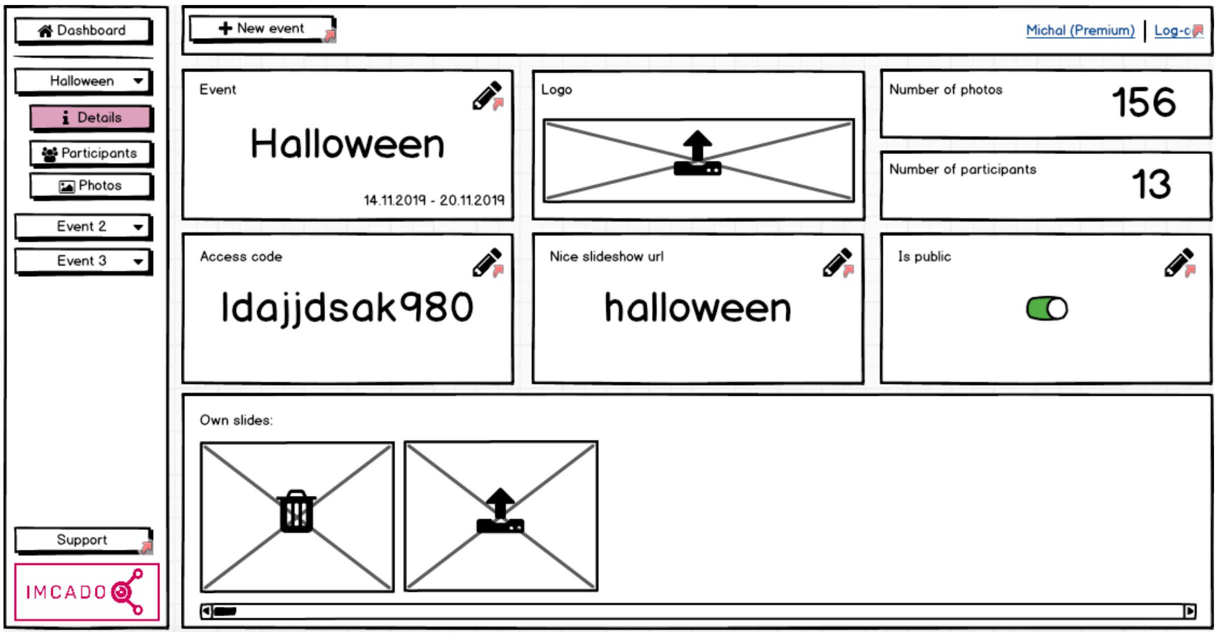
Do you want to delete 'Halloween' event? All photos will be lost.

Name	Start date	Actions
Halloween	14-11-2019	ownload photos Edit Delete
Evnet 2	17-11-2019	l photos Edit Delete
Event 3	14-12-2019	ownload photos Edit Delete

The screen should work like a modal and overlap the whole view. Use the data that was already loaded for the event.

When the user click yes trigger the azure backend “{backend_url}/event/${eventId}/delete”.

7. Event details



The screen contains all event details. All date for this screen can be received from "{backend_url}/event/${eventId}". When the user has a paid account the following can be edited: *Logo*, *Access code*, *Nice slideshow URL* and *Own slides*. Otherwise, the fields are visible but not editable – greyed out.

You can find information about feature definition algorithm in chapter 16.

The Event block and the Is Public block are edited by the Edit event modal (see chapter 5). Access code block and Nice slideshow URL block are labels and become editable elements after the user the edit icon. Use the Edit access code and Edit slideshow URL modals to edit the values (see chapter 8 and 9).

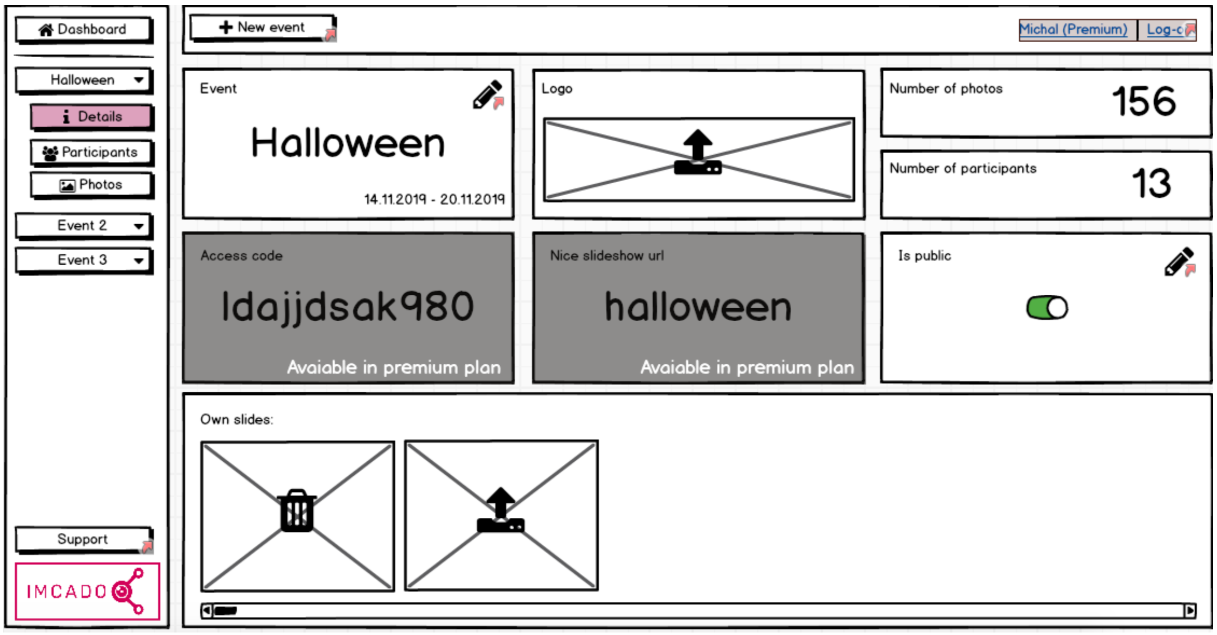
Event block contains Name, start and end date.

Logo block allows the user to upload a logo picture. The saved logo picture can be received from "{backend_url}/event/{eventIdString}". When the user wants to set the logo picture it is sent to "{backend_url}/event/{eventIdString}/logo". When the user wants to remove the logo the delete request under "{backend_url}/event/{eventIdString}/logo" is called

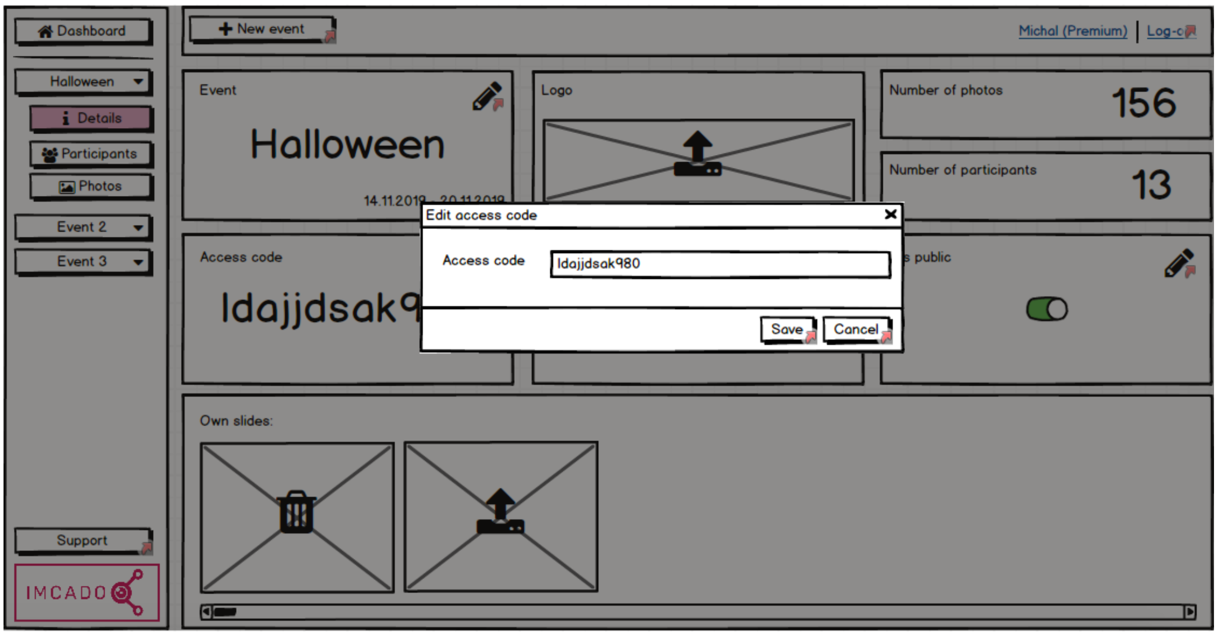
The number of photos and the number of participants contain the data and are not editable.

Own slides block displays the user-defined photos for the event that can be received from "{backend_url}/event/{eventIdString}". It allows to remove the already uploaded photos using a delete request to "{backend_url}/event/{eventIdString}/slide/{slideIdString}" and upload new ones using "{backend_url}/event/{eventIdString}/slides". The scroll bar appears when there are enough photos.

This is how the screen looks when the user has not Access code and Nice slideshow URL in his subscription.



8. Edit access code

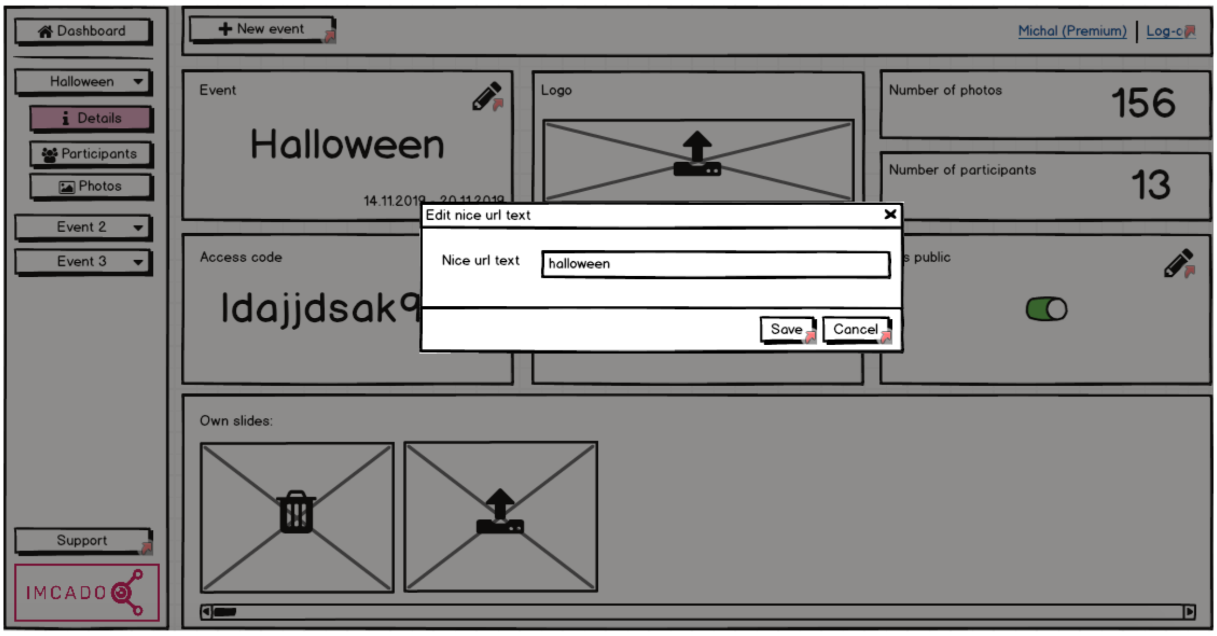


The screen should work like a modal and overlap the whole view.

When the user click Save trigger the Azure backend "{backend_url}/event/${eventId}/accessCode".

When text filed is empty show additional information that – *Access code will be generated by the system.*

9. Edit slideshow url

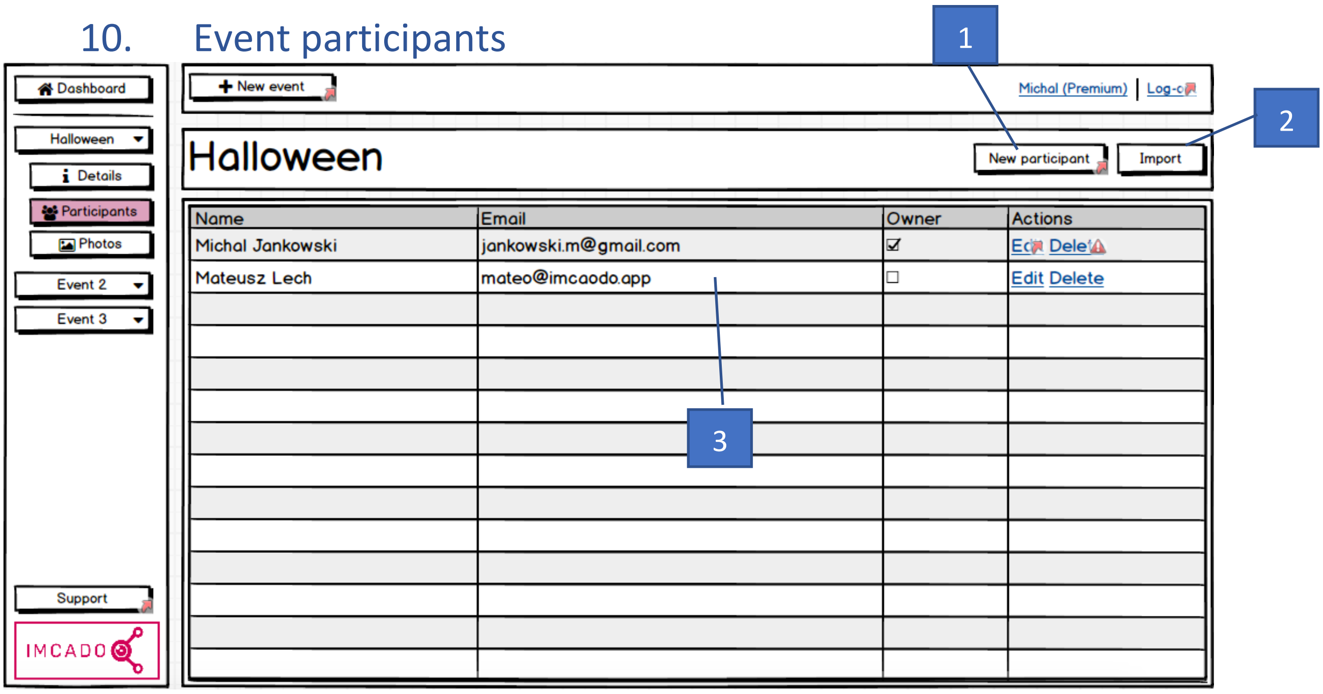


The screen should work like a modal and overlap the whole view.

When the user click Save trigger the azure backend “{backend_url}/event/${eventId}/slideShowUrl”.

When text filed is empty show additional information that – *Slide show url will be generated by the system.*

10. Event participants



Halloween

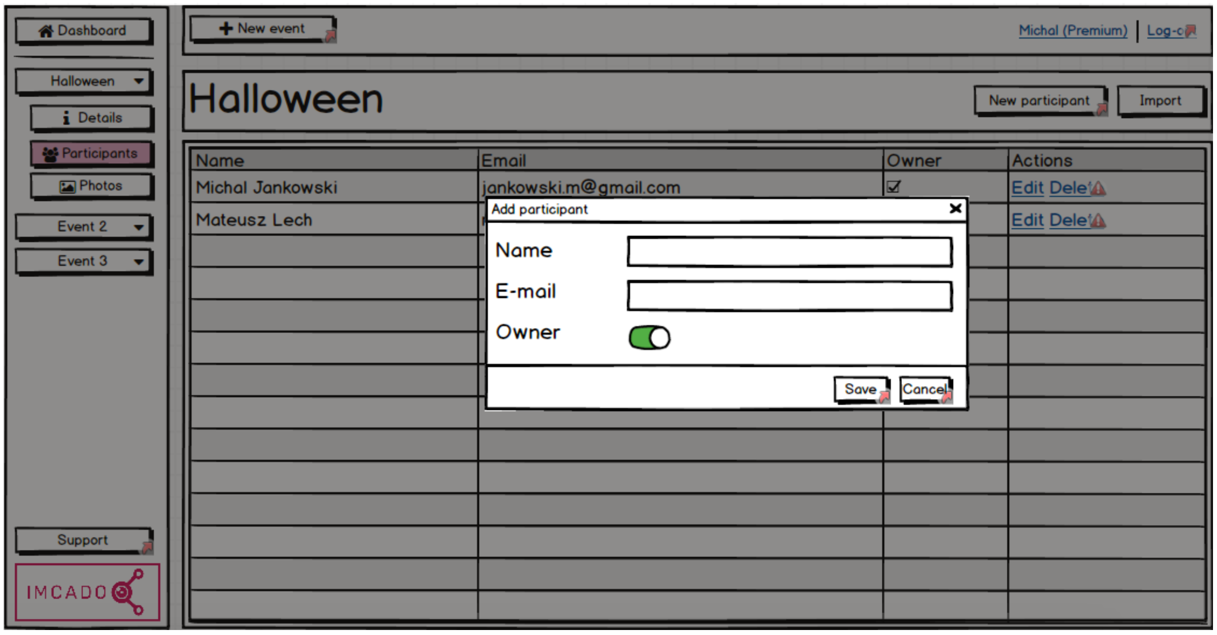
Name	Email	Owner	Actions
Michal Jankowski	jankowski.m@gmail.com	☑	Edit Delete
Mateusz Lech	mateo@imcaodo.app	☐	Edit Delete

The screen contains all event participant details. You can get data from {backend_url}/event/${eventId}/patricipants

Section nr 1 contains a New participant button that shows the New participant modal (see chapter 11).
Section nr 2 contains a Import button that allows the user to upload a CSV with Name and email data.
Section nr 3 contains the participant details. Each line contains Name, Email, Admin flag and available actions:

- Edit – show Edit participant modal (see chapter 12)
- Delete – show Delete participant modal (see chapter 13)

11. Add participant



Name	Email	Owner	Actions
Michal Jankowski	jankowski.m@gmail.com	☑	Edit Dele
Mateusz Lech			Edit Dele

Add participant

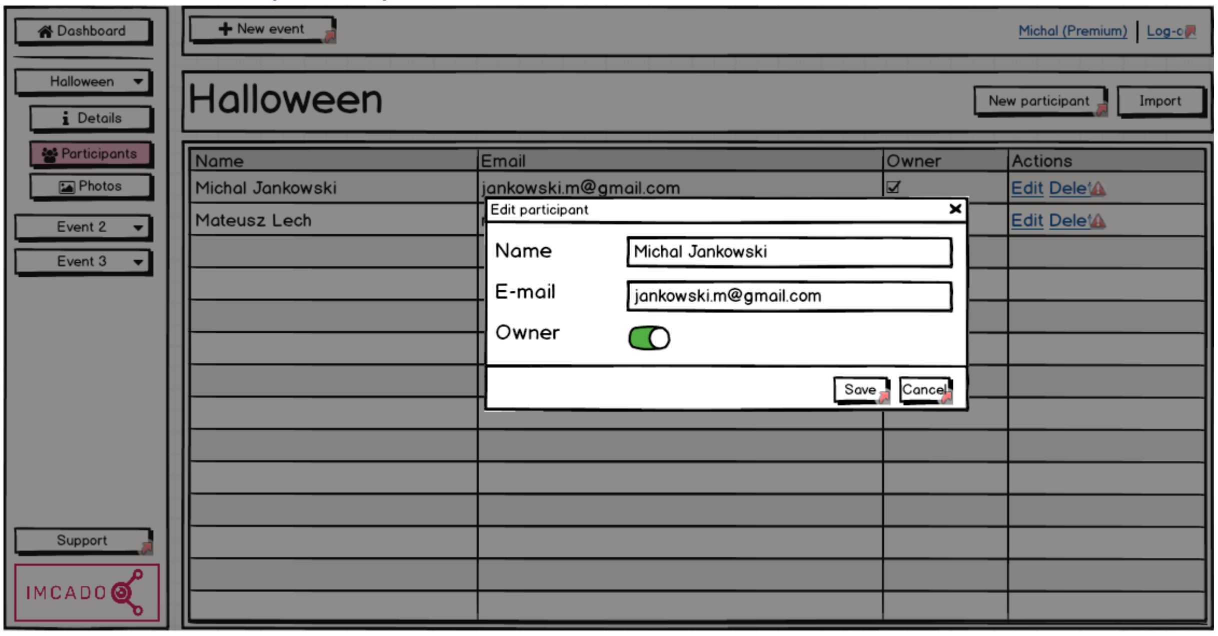
Name	
E-mail	
Owner	●

The screen should work like a modal and overlap the whole view.
Data validation:

- Name should not be empty
- E-mail not empty and a valid email address

When the user click save trigger the Azure backend
{backend_url}/event/${eventId}/participant/${email}

12. Edit participant



Halloween

Name	Email	Owner	Actions
Michal Jankowski	jankowski.m@gmail.com	☑	Edit Delete
Mateusz Lech			Edit Delete

Edit participant

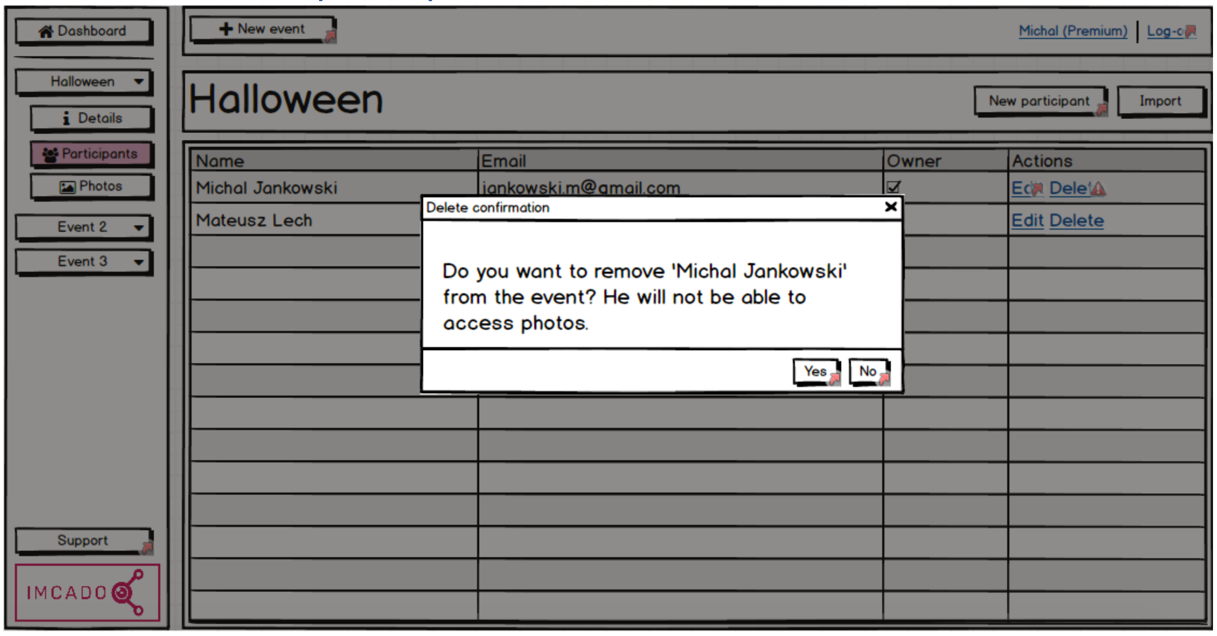
Name	Michal Jankowski
E-mail	jankowski.m@gmail.com
Owner	●

The screen should work like a modal and overlap the whole view.
Data validation:

- Name should not be empty
- E-mail not empty and a valid email address

When the user click save trigger the azure backend
{backend_url}/event/${eventId}/participant/${email}

13. Delete participant



Halloween

Name	Email	Owner	Actions
Michal Jankowski	jankowski.m@gmail.com	☑	Edit Delete
Mateusz Lech			Edit Delete

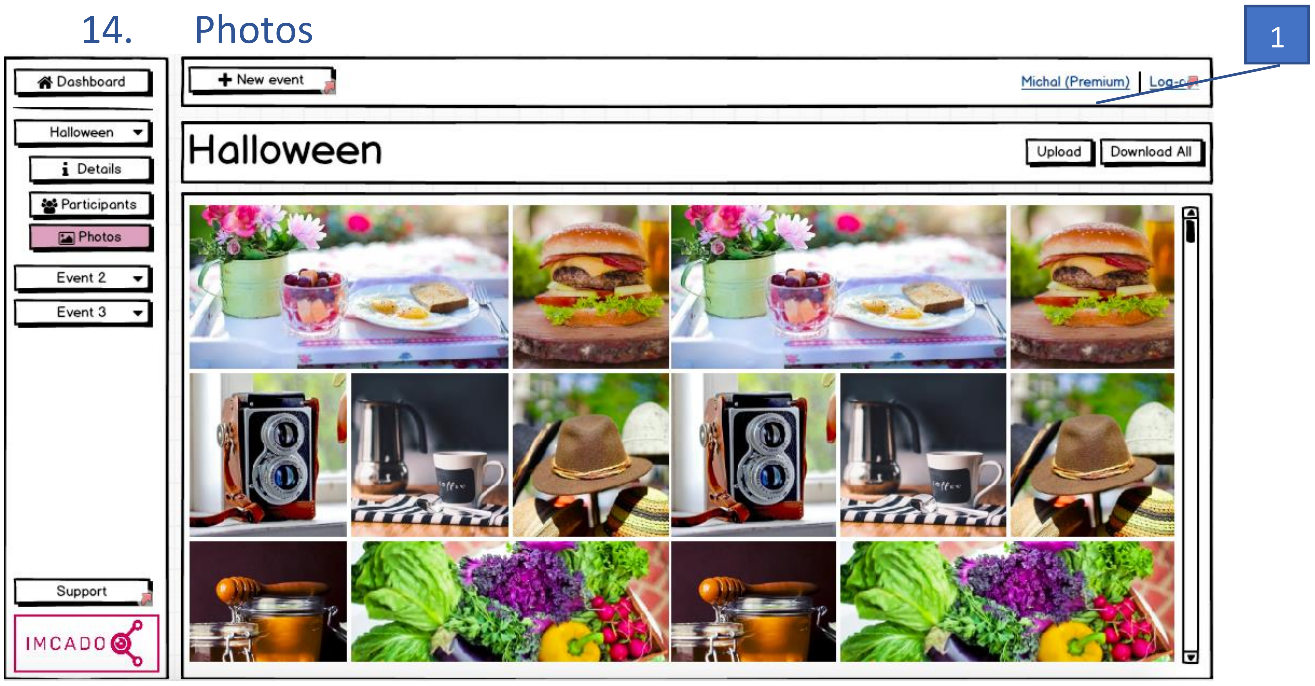
Delete confirmation

Do you want to remove 'Michal Jankowski' from the event? He will not be able to access photos.

The screen should work like a modal and overlap the whole view. Use the data that was already loaded for the event.

When the user clicks Yes trigger the azure backend {backend_url}/event/${eventId}/participant/${email}

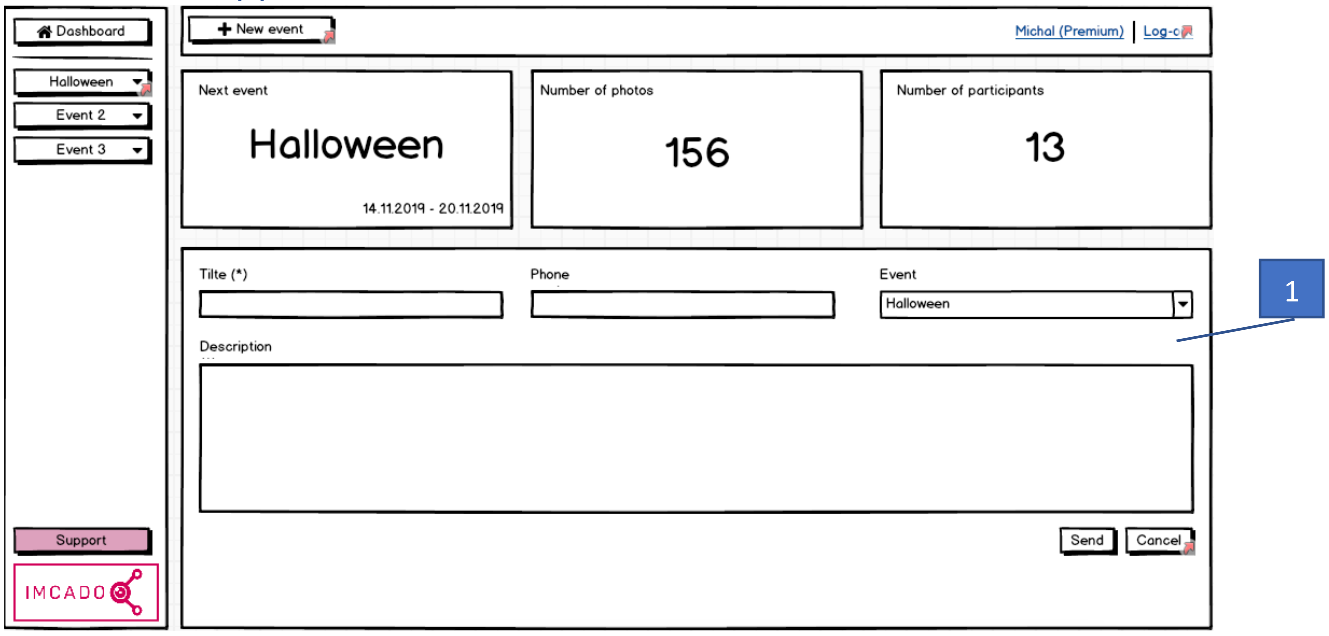
14. Photos



The screen contains all photos the current event contains. Get the data endpint {backend_url}/event/${eventId}/photos

When one of the photos is clicked open an overlay gallery lightbox for all photos of the event. Initially lightbox should be on the clicked photo.
Section nr 1 the Download All - if the user uses a paid plan and feature is enabled – triggers the azure backend API "{backend_url}/event/${eventId}/photos/archive" and shows modal to the user that the link to the archive will be sent per mail otherwise this button should be disabled.
Section nr 1 the Upload button allows to upload multiple photos to the event - triggers azure backend "{backend_url}/event/${eventId}/photos/dslr" multiple times.

15. Support



The Next event, Number of photos and Number of participants blocks are the same as in the Dashboard screen.
In Section nr 1 the user can define a problem that he has. The title is free text and can't be empty. The phone has no validation and is optional. Event dropdown allows selecting one of the user's events the problem is related to. The description is a text area. When the user clicks Send trigger the Azure backend {backend_url}/support.

16. User features settings

The permissions for a user are provided by the Azure backend {backend_url}/users/current in the "FeaturesFlag" value. The provided number is be parsed as bit flag array.

Bit meaning for "FeaturesFlag" – 1 means permission granted, 0 means permission denied:

Right name	Archive items/ photos	Own logo	Nice slideshow url	Own slides	Short code
Bit position	5	4	3	2	1
Example "29"	1	1	1	0	1
Example "4"	0	0	1	0	0

17. Language support

Determine the user language from the detected browser language. Depending on the found language use the corresponding I18N label file. If the found language is not defined use English as default.
The found language is sent to the Azure backend "{backend_url}/users" as Language parameter.

18. End points definiton

a. Login screen

CreateOrUpdateUser

Url: {BaseUrl}/users
Request Type: POST, PUT
Body:

```
{
    "Name" : "Michal3",
    "Email" : "aa@aa.pl",
    "ProfilePictureLink" : "http://........",
    "Language" : "pl",

}
```

Limitations:

- Name – required, max 500 characters
- Email – required, max 500 characters, valid email
- ProfilePictureLink – max 500 characters, link to profile picture

Response: 202

b. Dashboard

GetEvents

Url: {BaseUrl}/events
Request Type: GET
Param:

- "LastCheckDate" : "2018-02-15T16:54:09.823547Z",

Limitations:

- LastCheckDate – optional, when empty all events will be returned

Response: 200 – list of events with a date when last time it has been checked

Remarks:

- Events that have been modified after the last check date can be shown on the list.

GetEvent

Url: {BaseUrl}/event/{eventIdString}
Request Type: GET
Response: 200 – event details:

```
{
        "Name" : "Michal4",
        "StartDateTime" : "2012-04-23T18:25:43.511Z",
        "EndDateTime" : "2012-05-23T18:25:43.511Z",
        "Latitude": "10.0",
        "Longitude" : "20.0",
        "IsOpen" : "true"
        “NumberOfPhotos”: 25
        “NumberOfParticipants”: 40
         “AccessCode”: “Imcado”
        “SlideShowCode”: “Imcado”
        “LogoLink”: “https://imcado....”
        “Slides”:
                [
                        {
                                „id”: guid
                                „url”: „https:///
                        }
                ]
}
```

GetUser
Url: {BaseUrl}/users/current
Request Type: GET
Body:

```
{
}
```

Response: 200 – with user data:

```
{
        "Name" : “Michal”
        "Email" : “mail@imcado.app”
        "ProfilePictureLink" : “https://imcado.....”
        "DataCollectionConsent" : true
        "ApplicationRated" : true
        "SubcriptionName" : “Basic”
        "FeaturesFlag" : 32
}
```

ZipPhotos
Url: {BaseUrl}/event/{eventIdString}/photos/archive
Request Type: POST

Limitations:

- eventIdString – required, Guid()

Response: 202

c. New event

CreateEvent
Url: {BaseUrl}/events
Request Type: POST

Body:
```
{
	"Name" : "Michal4",
	"StartDateTime" : "2012-04-23T18:25:43.511Z",
	"EndDateTime" : "2012-05-23T18:25:43.511Z",
	"Latitude": "10.0",
	 "Longitude" : "20.0",
	 "IsOpen" : "true"
}
```

Limitations:

- Name – required, max 500 characters
- StartDateTime – required
- EndDateTime – required, later or equal than StartDateTime
- Latitude – optional, value between -180 and 180
- Longitude – optional, value between -90 and 90
- IsOpen – optional, default value False

*if Latitude and Longitude are used then both values must be provided

Response: 202 with body that contains IdEventu

d. Edit event

UpdateEvent

Url: {BaseUrl}/event/{eventIdString}
Request Type: PUT
Body:
```
{
	"Name" : "Michal4",
	"StartDateTime" : "2012-04-23T18:25:43.511Z",
	"EndDateTime" : "2012-05-23T18:25:43.511Z",
	"Latitude": "10.0",
	 "Longitude" : "20.0",
	 "IsOpen" : "true"
}
```

Limitations:

- EventIdString – required, guid
- Name – required, max 500 characters
- StartDateTime – required
- EndDateTime – required, laterod StartDateTime
- Latitude – optional, value between -180 and 180
- Longitude – optional, value between -90 and 90
- IsOpen – optional, default value False

*if Latitude and Longitude are used then both values must be provided

Response: 202 with body that contains IdEventu

e. Delete event

DeleteEvent

Url: {BaseUrl}/event/{eventIdString}/delete
Request Type: DELETE
Limitations:

- EventIdString – required, guid

Response: 202

f. Event details

GetEvent

Url: {BaseUrl}/event/{eventIdString}
Request Type: GET
Response: 200 – event details:

```
{
        "Name" : "Michal4",
        "StartDateTime" : "2012-04-23T18:25:43.511Z",
        "EndDateTime" : "2012-05-23T18:25:43.511Z",
        "Latitude": "10.0",
        "Longitude" : "20.0",
        "IsOpen" : "true"
        “NumberOfPhotos”: 25
        “NumberOfParticipants”: 40
         “AccessCode”: “Imcado”
        “SlideShowCode”: “Imcado”
        “LogoLink”: “https://imcado....”
        “Slides”:
                [
                        {
                                „id”: guid
                                „url”: „https:///
                        }
                ]
}
```

AddLogo

Url: {BaseUrl}/event/{eventIdString}/logo
Request Type: POST
Headers:

- “FileContentType” : “image/<typ obrazka>”
- “Content-Type”: “application/octet-stream”

Body:

- Image data stream

Limitations:

- eventIdString – required, Guid()
- FileContentType – Accepted types: png, tiff, jpg, gif

Response: 202

DeleteLogo

Url: {BaseUrl}/event/{eventIdString}/logo
Request Type: DELETE
Limitations:

- EventIdString – required, guid

Response: 202

AddSlide

Url: {BaseUrl}/event/{eventIdString}/slides
Request Type: POST
Headers:

- “FileContentType” : “image/<typ obrazka>”
- “Content-Type”: “application/octet-stream”

Body:

- Image data stream

Limitations:

- eventIdString – required, Guid()
- FileContentType – Accepted types: png, tiff, jpg, gif

Response: 202

DeleteSlide

Url: {BaseUrl}/event/{eventIdString}/slide/{slideIdString}
Request Type: DELETE
Limitations:

- EventIdString – required, guid
- SlideIdString – required, guid

Response: 202

g. Edit access code

UpdateAccessCode

Url: {BaseUrl}/event/{eventIdString}/accessCode
Request Type: POST
Body:

```
{
        “accessCode”:”imcado”
}
```

Limitations:

- eventIdString – required, Guid()

Response: 202

h. Edit slideshow url

UpdateSlideShowUrl

Url: {BaseUrl}/event/{eventIdString}/slideShowUrl
Request Type: POST
Body:

```
{
        “url”:”imcado”
}
```

Limitations:

- eventIdString – required, Guid()

Response: 202

i. Event participants

GetParticipants

Url: {BaseUrl}/event/{eventIdString}/participants
Request Type: GET
Limitations:

- EventIdString – required, guid

Response: 200 – list of participants with a date when last time it has been checked

j. Add participant

UpdateParticipantByEmail

Url: {BaseUrl}/event/{eventIdString}/participant/{participantEmail}
Request Type: POST, PUT
Body:

```
{
        "IsOwner" : false,
}
```

Limitations:

- EventIdString – required, guid
- ParticipantEmail – email to the person who should have access to an event as a participant; it does not need to exist in db. In this case, it will become a new login.
- IsOwner – true / false – is owner

Response: 202

k. Edit participant

UpdateParticipantByEmail

Url: {BaseUrl}/event/{eventIdString}/participant/{participantEmail}
Request Type: POST, PUT
Body:

```
{
        "IsOwner" : false,
}
```

Limitations:

- EventIdString – required, guid

- ParticipantEmail – email to the person who should have access to an event as a participant; it does not need to exist in db. In this case, it will become a new login.
- IsOwner – true / false – is owner

Response: 202

l. Delete participant

DeleteParticipantByEmail

Url: {BaseUrl}/event/{eventIdString}/participant/{participantEmail}
Request Type: DELETE
Limitations:

- EventIdString – required, guid

Response: 202

m. Photos

ZipPhotos

Url: {BaseUrl}/event/{eventIdString}/photos/archive
Request Type: POST
Limitations:

- eventIdString – required, Guid()

Response: 202

GetPhotos

Url:
{BaseUrl}/event/{eventIdString}/photos
Request Type: GET
Params:

- "LastCheckDate" : "2018-02-15T16:54:09.823547Z",
- Limitations:
- eventIdString – required, Guid()
- LastCheckDate – optional, when empty all photos will be returned

Response: 200 – list of photos with a date when last time it has been checked

AddDSLRPhoto

Url: {BaseUrl}/event/{eventIdString}/photos/dslr
Request Type: POST
Headers:

- "PhotoTakenDate" : "2012-04-23T18:25:43.511Z",
- “FileContentType” : “image/<image type>”
- “Content-Type”: “application/octet-stream”

Body:

- Image data stream

Limitations:

- eventIdString – required, Guid()
- FileContentType – Accepted types: png, tiff, jpg, gif

Response: 202

n. Support

CreateRequest

Url: {BaseUrl}/support/
Request Type: POST
Body:

```
{
        “userEmail”:”mail@imcado.pl”
        “title”:”title”,
        “phone”:”1234567”,
        “eventIdString”:”24......”,
        “eventName”:”My Event”,
        “message”:”message”
}
```

Limitations:

- userEmail – required, email
- title – required
- eventIdString – required, Guid()
- eventName – required
- message – required

Response: 202