WannaSport Provider Admin

Introduction

This document illustrates the requirements for our “Provider Administration” back office system Frontend.
This project is tailored to our partners, in order to let them handle a subset of the functionalities the system offers.

The goal of this first version is to make it possible for a partner - or “Provider Administrator” - to

- Choose a facility
- Create an activity
- List the created activities
- Cancel an activity

Further details about these features can be found within each section of this document.
Authentication and authorizations are not taken into account for this version.

Interaction with the backend

Instead of interacting with an actual backend, the developer will use a client that mocks the calls. Data will be persisted in browser local storage using the key “`ws_provideradmin`”;

The client is available at this url as a single file that can be imported as a es6 module.
All methods are asynchronous. As a requirement, the await / async should be used, instead of the Promise syntax.

```
https://static.wannasport.dk/misc/client.js
```

Details about data retrieval will be provided within each section of this document.

Styling, implementation and middlewares

The following schemes have to be intended as wireframes, not as mockups.
It will be your responsibility to provide something that has a good look and feel, and usability as well.

This result can be achieved either through custom css or by using any existing styling middleware.

What to ship and technical requirements

The result should be a Single Page Application which should be able to run even with static hosting. Also:

- It should be an npm project, handling dependencies through npm
- It should use a bundler (possible webpack) to handle dependencies and pre-processing
- It should be an actually working project implementing all our requirements, making use of javascript and any other needed asset (css, html, jsx, etc...)
- It should include our mocked client in code
- Any framework or middleware can be used, as long as they're **completely** free to use and open source.
- It should be able to run by running a simple command (eg: npm run start)

Menu

The left menu should be hidden, and show up upon clicking a button.
Menu points:

- Partner info: placeholder (should not do anything)
- Create facility: placeholder (should not do anything)
- Select below “Facilities”: The facilities list is accessible by invoking the method getFacilities. It should be possible to choose which facility to interact with.
- Create activity: link to “Create Activity” Section.
- Activities: Link to “Activities” section.
- Settings: placeholder (should not do anything)

The default section should be the “create activity” section.
The current facility should be read from the url: all the urls will start with the facility id.

Header

User data in the header can be retrieved by invoking the method getUser()

Urls:

Create activity form and list/activities should have the following urls:

```
/{facilityId}/create-activity

/{facilityId}/list-activities
```

Other pages might or might not have a url. This choice is left to the developer.

Activity Creation

Introduction

An activity in WannaSport is a sport activity, taking place at a certain time, and that can be purchased (booked) by the final user.

Layout

It should be adaptive, so this form should scale nicely in a mobile device that has at least 320 of horizontal resolution. That could be achieved by hiding the left menu and having each input to take the whole line, under a certain resolution.

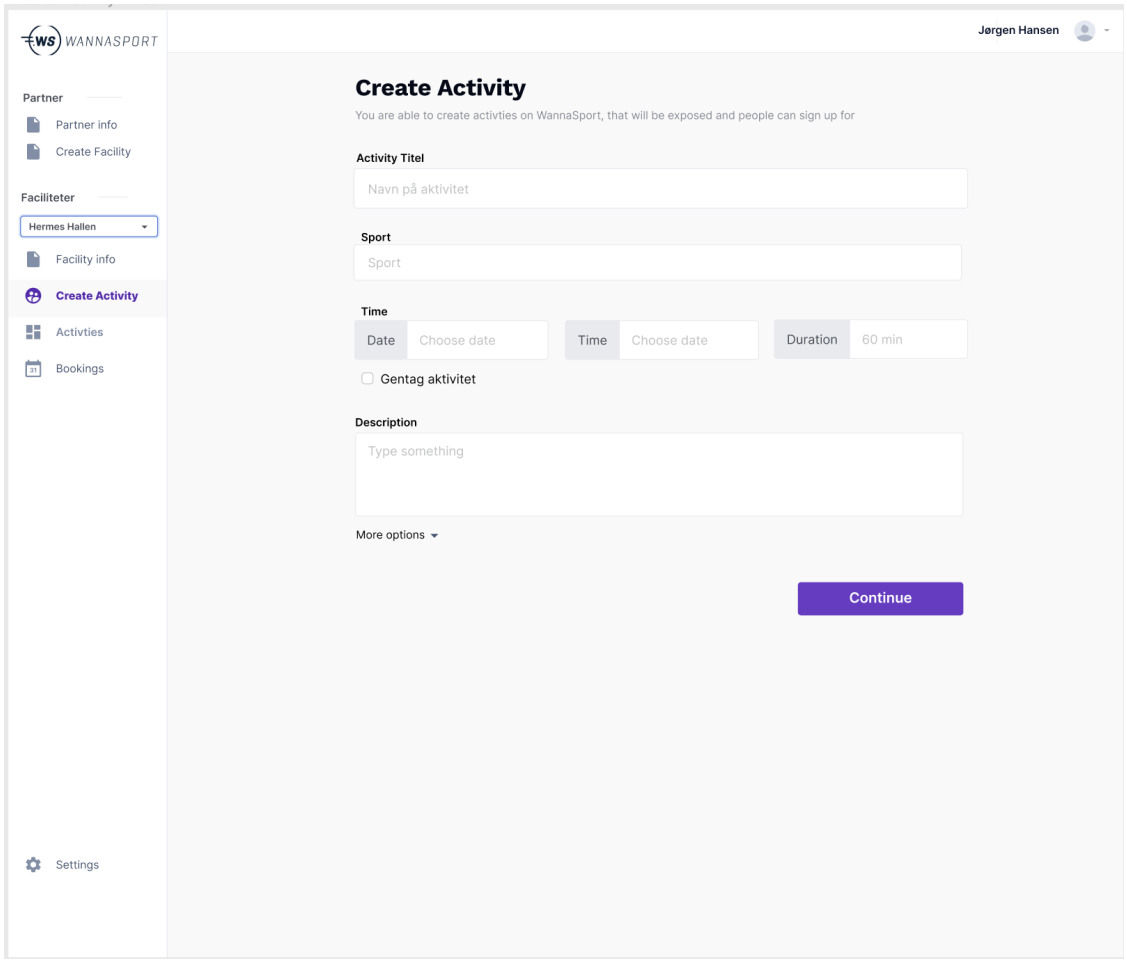
Form

The inputs showing up immediately are all mandatory.

- **Title**: Text, Mandatory, Length 5 - 50 characters
- **Sport:** Mandatory**,** Should be taken from a list of sports.
  - It should be selectable through some kind of suggestion box: a box where the user can type in and/or select through a list (see TypeAhead.js as a reference)
  - List can be obtained by calling getCategories
- **Date**:
  - A date in the future.
  - Should be selectable through a specific component (see datepicker as a reference)
- **Time:** Expressed in hours and minutes.
- **Duration:** Should be expressed in minutes.
- **Description**: Text, Mandatory, length 10 - 200 characters

Validation of fields should occur on the frontend side priorly to invoking the createActivity method.

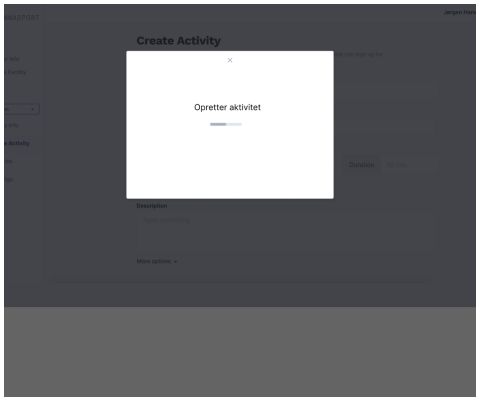
Whenever the form data is ready to be sent, the method createActivity should be invoked. Error messages returned by createActivity should be displayed to the user.

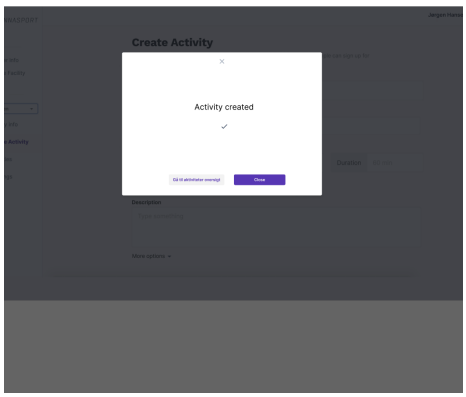


During the execution of createActivity, a loader should be displayed, and in case of success, a confirmation message with the option to

- return to the activity form (reset from all the changes)
- Go to the “activities” section.

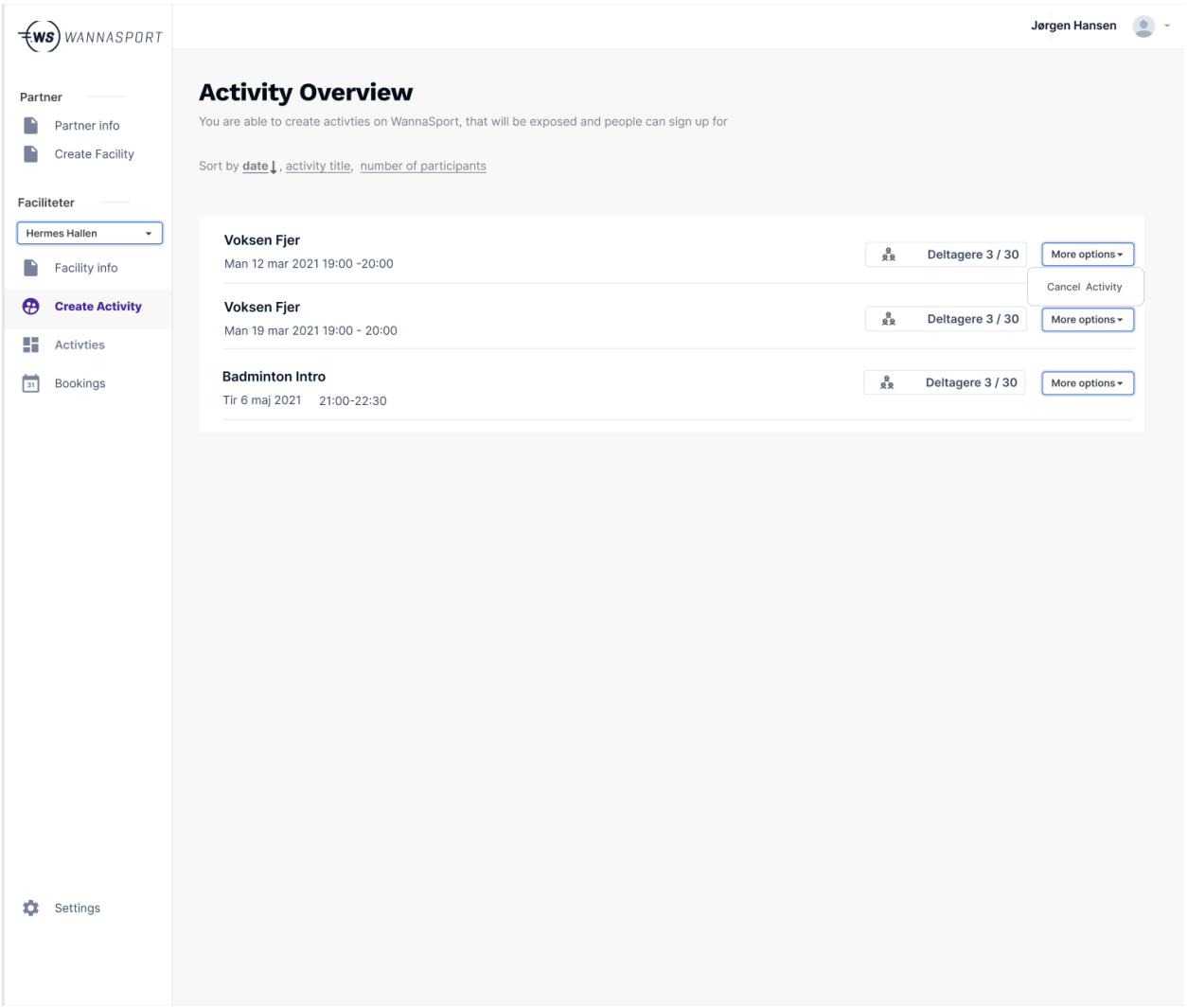
In case the creation failed, the user should be brought back to the form page, and the specific error message should be displayed.





Activity Listing

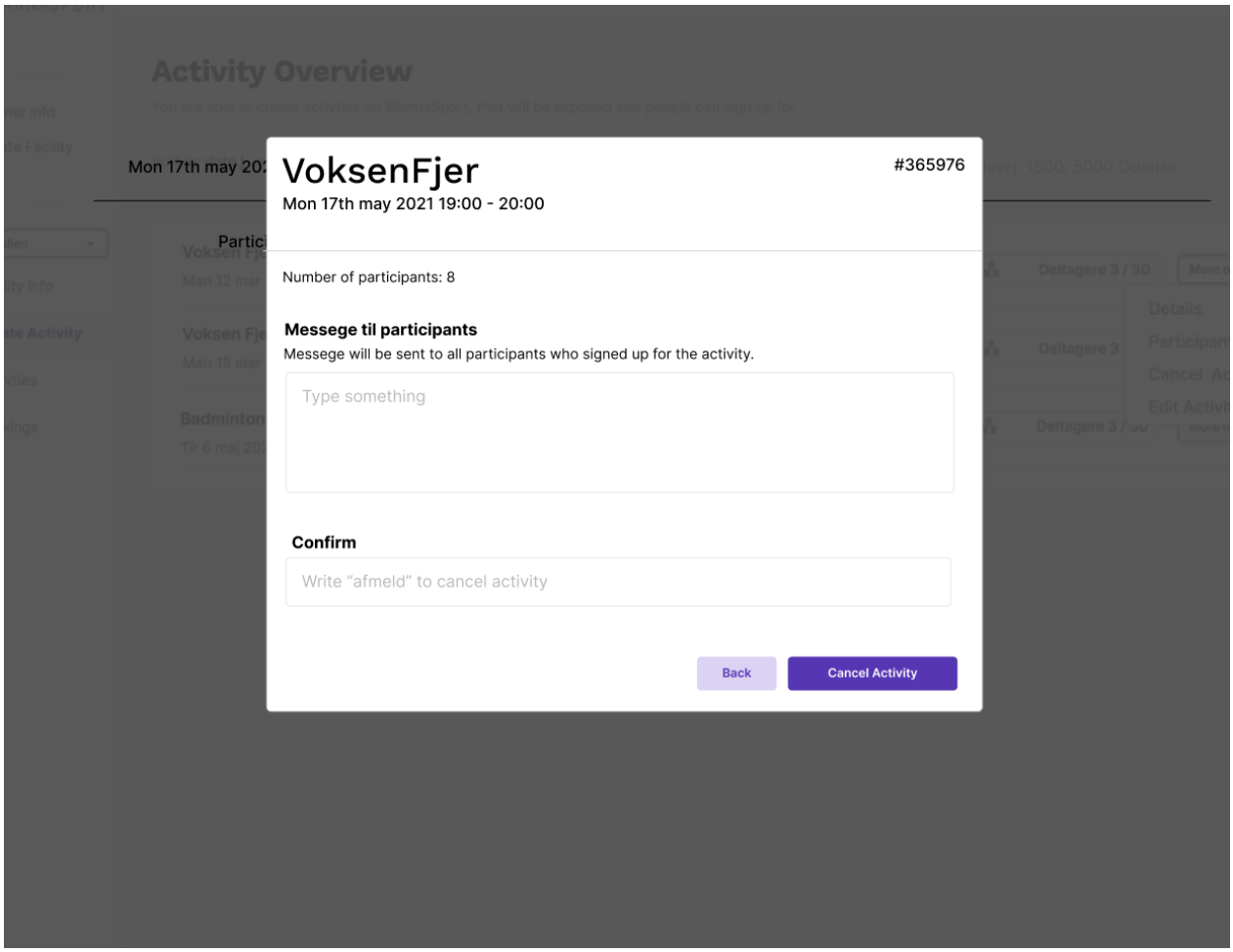
It should be possible to see all the activities for this facility.



Data should come from the method “getActivities”.
Also this view should scale well on mobile devices.

By clicking on “More options”, the “cancel activity” overlay should show up. Button should be clickable only if the property “canCancel” is set on the data.

Activity Cancellation

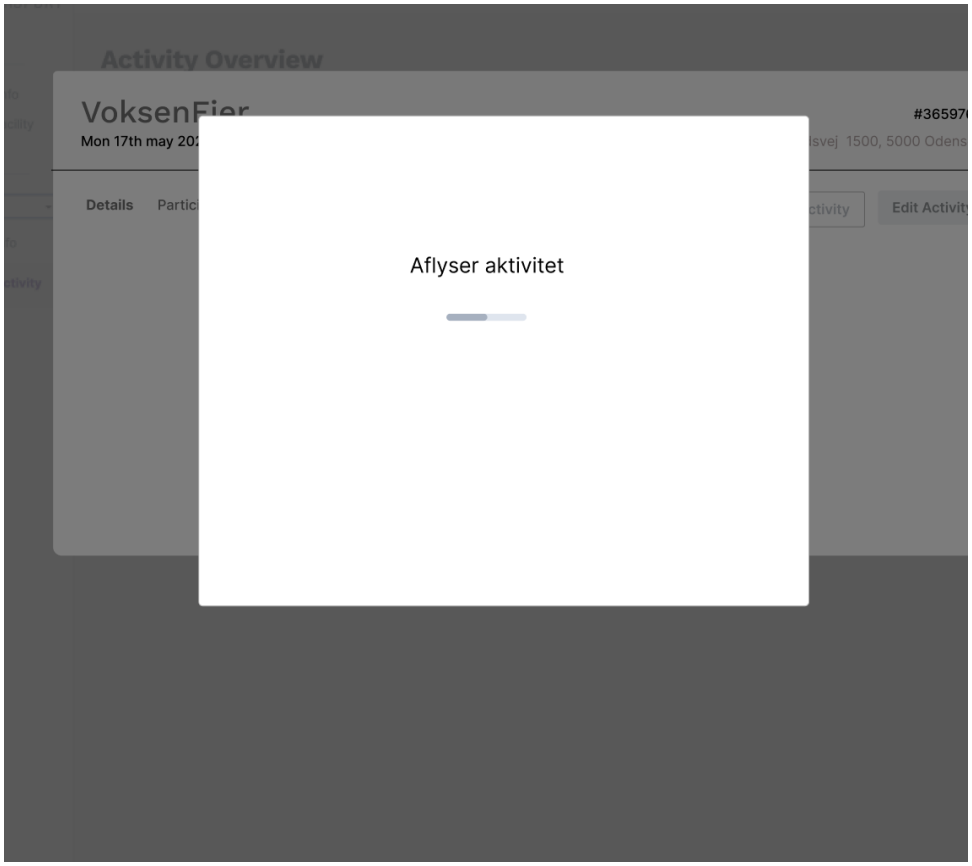


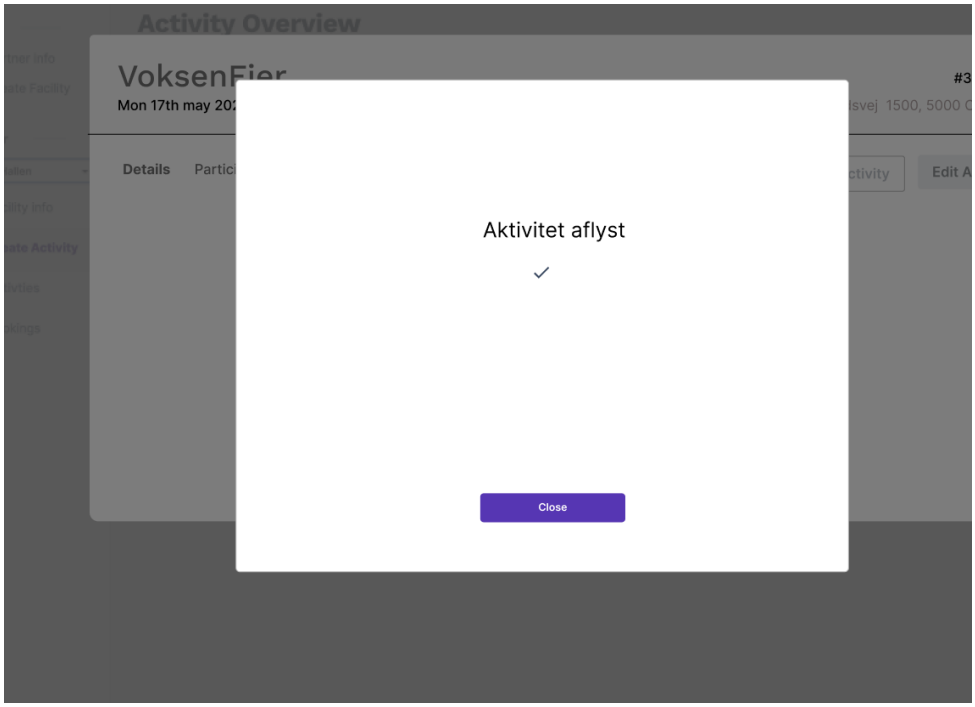
This section should be an overlay, showing some details about the activity, and letting the user insert a text message that is supposed to be sent to all the participants.

The user is asked to type in “afmeld” to confirm to cancel the activity.

Button should not be enabled until the text message (10 characters minimum) and the Confirm text are typed in.

Cancellation can be done by invoking method cancelActivity


Activity Overview
Details
Aflyser aktivitet


Activity Overview
Details
Aktivitet aflyst
Close

Api Client

getFacilities()

Returns a list of facilities:

```
[
      {
          name: 'Hermes Hallen',
          id: '3fc33045-21e6-494d-bc96-967ae26741b5'
      },
      {
          name: 'Fredericia Tennisklub',
          id: '6807bae8-c51e-4343-bfef-31791a9f5488'
      }
]
```

getCategories()

Returns a list of categories:

```
[
  {
    "Id": 0,
    "Name": "None",
    "NameTranslated": "None"
  },
  {
    "Id": 1,
    "Name": "Football",
    "NameTranslated": "Football",
    "SportsIds": [
      1
    ]
  }
]
```

createActivity(formData)

Input: the form data

```
formData:
```

```
{
      title: 'activity title',
      sport: 1,
      date: '2020-04-20'
      time: '18:00',
      duration: 60,
      description: 'this is an activity description',
      facilityId: '6807bae8-c51e-4343-bfef-31791a9f5488'
}
```

Returns the id of the created activity.

In case of error, it throws an exception

```
{
      message:'Something went wrong'
}
```

getUser()

Returns the user data:

```
{
      name: 'John Doe',
      picture: 'https://static.wannasport.dk/misc/user.jpeg'
}
```

getActivities(facilityId, sorting)

Input:

```
{
      sorting: 'date'
}
```

Sorting values can be:

- 'date'
- 'name'
- 'participants'

Returns the activity data:

```
[
```

```
    {
        id: 'ef995215-5377-4e38-842a-0e2933bc54fb'
        facilityId: '6807bae8-c51e-4343-bfef-31791a9f5488'
        name: Voksen Fjer',
        date: '2021-03-20',
        startTime: '18:00',
        endTime: '19:00',
        participants: 3,
        maxParticipants: 20,
        canCancel: true
    },
    {
        id: 'ef995215-5377-4e38-842a-0e2933bc54fb'
        facilityId: '6807bae8-c51e-4343-bfef-31791a9f5488'
        name: Voksen intro',
        date: '2021-03-10',
        startTime: '18:00',
        endTime: '20:00',
        participants: 3,
        maxParticipants: 20.
        canCancel: true

    }

]
```

getActivity(activityId)

Returns the activity data:

```
[
    {
        id: 'ef995215-5377-4e38-842a-0e2933bc54fb'
        facilityId: '6807bae8-c51e-4343-bfef-31791a9f5488'
        name: Voksen Fjer',
        date: '2021-03-20',
        startTime: '18:00',
        endTime: '19:00',
        participants: 3,
        maxParticipants: 20
    }
]
```

cancelActivity(id, message)

Input:

```
{
      id: 'ef995215-5377-4e38-842a-0e2933bc54fb',
      message: 'this is a message to all participants'
}
```