

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

Jiskefet - UI

Bastiaan Reinalda and Ramon Gill

December 17, 2018

Overview

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

1 Dependencies

2 Views

3 Components

4 State Management

5 Planning

Dependencies

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

- Mithril
- Typescript (TSX)
- Marked
- Bootstrap

Views

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/views/Logs.tsx

```
<Badges
  filters={State.FilterModel.getFilters('log')}
  onEvent={(key: string) => {
    State.FilterModel.setFilter('log', key, null);
    this.fetchWithFilters();
  }}
  onEventAll={() => {
    State.FilterModel.setFiltersToDefaults('log');
    this.fetchWithFilters();
  }}
  ignoredFilters={[
    'orderBy',
    'orderDirection',
    'pageSize',
    'pageNumber'
  ]}
/>
<Spinner
  isLoading={State.LogModel.isFetchingLogs}
  component={createDummyTable(State.FilterModel.getFilters('log').pageSize, LogColumns)}
>
  <Table
    data={State.LogModel.list}
    columns={LogColumns}
    orderBy={State.FilterModel.getFilters('log').orderBy}
    orderDirection={State.FilterModel.getFilters('log').orderDirection}
    onHeaderClick={(accessor: string) => {
      State.FilterModel.switchOrderBy('log', accessor);
      this.fetchWithFilters();
    }}
  />
</Spinner>
```

Views

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/views/CreateLog.tsx

```
<div class="form-group">
  <div class="card shadow-sm bg-light">
    <Tabs
      tabs=(CreateLogTabs)
      entity=(State.LogModel.createLog)
      func={(content: string) => this.addDescription(content)}
      caller={'description'}
    />
  </div>
</div>
<AttachmentComponent
  attachTo="Log"
  hideImagePreview={true}
  isExistingItem={false}
/>
<br />
<button type="submit" class="btn btn-primary">Submit</button>
<button
  type="button"
  class="btn btn-info float-right"
  data-toggle="modal"
  data-target="#MarkdownHelpText"
>
  Formatting help
</button>
<Modal id="MarkdownHelpText" title="Markdown help">
  <MarkdownViewer id={'MarkdownHelpTextViewer'} content={MarkdownHelpText} />
</Modal>
```

Views

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/views/Log.tsx

```
<dl class="row">
  <dt class="col-sm-6">Log id</dt>
  <dd class="col-sm-6">{State.LogModel.current.logId}</dd>

  <dt class="col-sm-6">Subtype:</dt>
  <dd class="col-sm-6">
    {State.LogModel.current.subtype === 'run' ?
      <span class="badge badge-warning">
        {State.LogModel.current.subtype}
      </span>
      : State.LogModel.current.subtype}
    </dd>

  <dt class="col-sm-6">Origin:</dt>
  <dd class="col-sm-6">
    {State.LogModel.current.origin === 'human' ?
      <span class="badge badge-success">
        {State.LogModel.current.origin}
      </span>
      : State.LogModel.current.origin === 'process' ?
      <span class="badge badge-primary">
        {State.LogModel.current.origin}
      </span>
      : State.LogModel.current.origin}
    </dd>
</dl>
```

Components - Attributes

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/components/Filter.tsx

```
interface InputField {  
  name: string; // name should exist as a key in the filters attribute.  
  type: string;  
  placeholder?: string;  
  label?: string;  
  event: string;  
}  
  
interface Attrs {  
  /**  
   * The input fields used for filtering.  
   */  
  inputFields: InputField[];  
  /**  
   * Function being called when the event happens on an input field.  
   */  
  onEvent: (key: string, value: string | number) => void;  
  /**  
   * The values of the filters.  
   */  
  filters: { [key: string]: string | number | null };  
}
```

Components - Filters

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies


Views

Components

State
Management

Planning

- Filter names and labels
- Collapseble

 Filters

^ Log id

^ Title

Components - Filters

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/components/Filter.tsx

```
<Collapse
  id={'filters'}
  icon={<span class="fas fa-filter" />}
  title={'Filters'}
>
  {inputFields && inputFields.map((inputField: InputField) =>
    (
      <Collapse
        id={inputField.name}
        icon={
          <i
            class="fas fa-angle-down mt-2 jf-rotate-if-collapsed jf-rotate-if-not-collapsed"
          />
        }
        title={inputField.label ? inputField.label : inputField.name}
      >
        <div class="form-group mt-2">
          <input
            type={inputField.type}
            class="form-control form-control-sm"
            id={inputField.name}
            {...{
              [inputField.event]: (event: Event) => {
                onEvent(inputField.name, event.target.value);
              }
            }}
            value={filters[inputField.name]}
            placeholder={inputField.placeholder}
          />
        </div>
      </Collapse>
    )
  )}
</Collapse>
```

Components - Layout

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

■ Wrapper component

The screenshot displays the Jiskefet application interface. On the left is a dark sidebar with a menu containing 'Logs', 'Runs', 'Create new log', and 'Subsystems Overview'. The main content area is titled 'Jiskefet' and features a 'Filters' section with expandable sections for 'Log id', 'Title', 'startCreationTime', and 'endCreationTime'. Each filter has a text input field. To the right of the filters is a table of logs. The table has columns for 'Log id', 'Title', 'Sub-type', and 'Ori'. It contains two rows of data. Below the table is a pagination control showing 'Page size' as 16, '1-2 of 2 rows', and buttons for 'Previous', '1', and 'Next'.

Log id	Title	Sub-type	Ori
2	Dit is een wysiwig test	run	P
1	This is a log test for runs in log	run	P

Page size: 16 | 1-2 of 2 rows | Previous | 1 | Next

Components - Layout

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/components/Layout.tsx

```
export default class Layout extends MithrilTsxComponent<{}> {  
  view(vnode: Vnode) {  
    return (  
      <div>  
        <NavBar />  
        <div class="jif-wrapper">  
          <SideBar />  
          <Content>  
            {vnode.children}  
          </Content>  
        </div>  
      </div>  
    );  
  }  
}
```

Components - Pagination

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

- Page size
- Page number
- Amount of pages

Page size

16 ▾

Previous

1

2

Next

1-16 of 17 rows

Components - Pagination

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/components/Pagination.tsx

```
<ul class="pagination pagination-sm jf-pagination">
  <li class={`page-item ${currentPage <= 1 && 'disabled'}`}>
    <a
      class="page-link"
      onClick={() => {
        this.inputIsActive = false;
        onChange(this.previousPage(currentPage));
      }}
    >
      Previous
    </a>
  </li>
  {numberOfPages > 0
    ? this.createPageElements(pageValues, vnode.attrs).map(
        (pageElement: JSX.Element) => pageElement
      )
    : this.noDataFound()
  }
  <li class={`page-item ${!(+currentPage === +numberOfPages || +numberOfPages === 0) && 'disabled'}`}>
    <a
      class="page-link"
      onClick={() => {
        this.inputIsActive = false;
        onChange(this.nextPage(numberOfPages, currentPage));
      }}
    >
      Next
    </a>
  </li>
</ul>
```

State management

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/models/State.ts

```
/**
 * The single state container for the application.
 * See the README in this directory (./models) for more info.
 */
export default {
  AppState,
  HttpErrorModel,
  SuccessModel,
  LogModel,
  RunModel,
  SubsystemPermissionModel,
  FilterModel,
  AttachmentModel,
  SubsystemModel,
  AuthModel,
  SubsystemOverviewModel,
  UserModel,
```

State management

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/models/Log.ts

```
const LogModel = {
  isFetchingLogs: false as boolean,
  isFetchingLog: false as boolean,
  isPatchingLinkRunToLog: false as boolean,
  count: 0 as number, // number of total rows available.
  list: [] as Log[],
  current: {} as Log,
  createLog: {} as LogCreate, // log being created
  async fetch(query?: string) {
    LogModel.isFetchingLogs = true;
    return request({
      method: 'GET',
      url: `${process.env.API_URL}logs${query ? `?${query}` : ''}`,
    }).then((result: { logs: Log[], count: number }) => {
      LogModel.isFetchingLogs = false;
      LogModel.list = result.logs;
      LogModel.count = result.count;
    }).catch((error: HttpError) => {
      LogModel.isFetchingLogs = false;
      State.HttpErrorModel.add(error);
    });
  },
};
```

State models

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

path: src/app/models/Filter.ts

```
const FilterModel = {  
  /**  
   * Set a filter, i.e. filterKey.key = value.  
   */  
  setFilter: (filterKey: string, key: string, value: string | number | null): void => {  
    Filters[filterKey][key] = value || null;  
    updateUrlFromFilters(filterKey);  
  },  
  /**  
   * Sets the query params from the url into the filters.  
   */  
  setFiltersFromUrl: (filterKey: string) => {  
    const filtersFromUrl = m.route.param();  
    _.merge(Filters[filterKey], filtersFromUrl);  
  },  
  /**  
   * Returns the filter object on key filterKey  
   */  
  getFilters: (filterKey: string) => {  
    return Filters[filterKey];  
  },  
  /**  
   * Sets all the values for the object at filterKey to null;  
   */  
  setFiltersToDefaults: (filterKey: string) => {  
    Object.assign(Filters[filterKey], DefaultFilters[filterKey]);  
    updateUrlFromFilters(filterKey);  
  },  
}
```


Planning

Jiskefet - UI

Bastiaan Reinalda
and Ramon Gill

Dependencies

Views

Components

State
Management

Planning

- Atomic design
- Redux
- Testing