Faculty of Computer Science and Mathematics
Prof. Dr. Felix Schwägerl
Summer semester 2025
02 May 2025

REGENSBURG

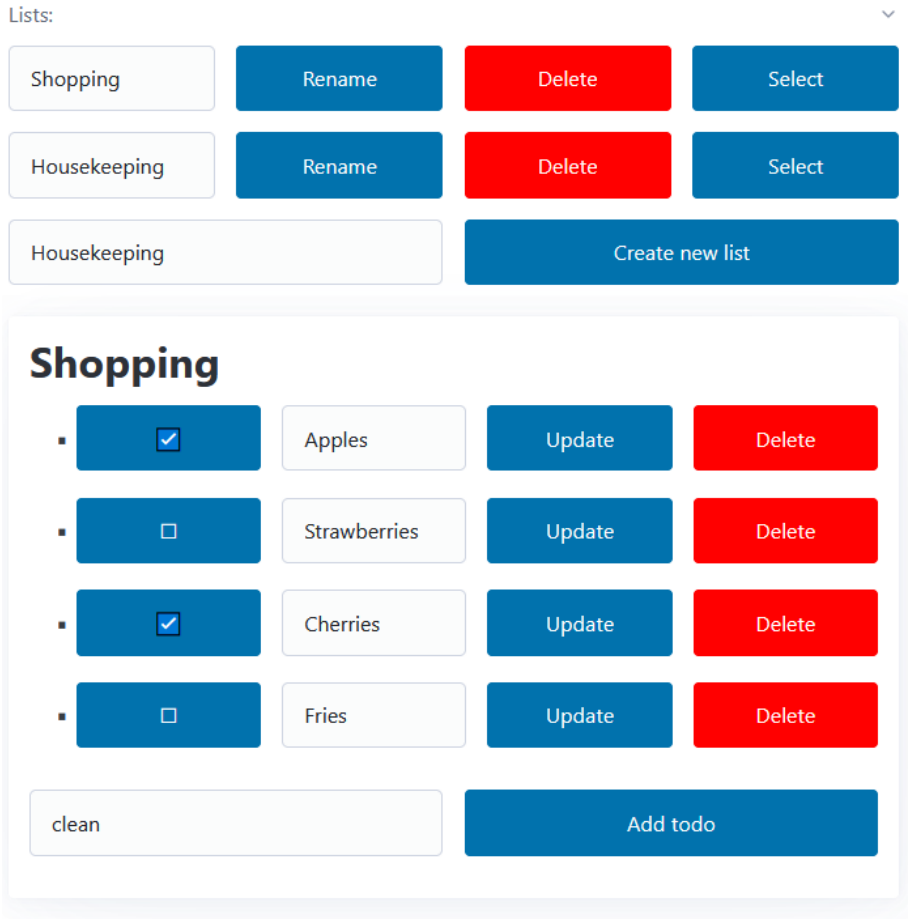
Web Technology Project (International Computer Science)

Exercise sheet 5 — Frontend Development

Deadline: 02 May 2025

Exercise 5.1 (Specification and UI Design)

In this exercise sheet, you will develop a the frontend of the to-do list application using React and PicoCSS as well as the REST API developed in Exercise sheet 2. The UI of the application is supposed to look as follows:



In the header part of the app, the user can create and manage their to-do lists. All existing lists are displayed here. They can be renamed or deleted using the corresponding buttons. The user can also create a new list. The application stores the current list ID in the background.

Once the user selects a list using the corresponding button, the list is displayed in the `main` part of the app. The tasks are displayed in an unordered list. Each list item contains the selection state, the name, as well as update and delete actions. Clicking the selection state button will toggle the selection state of the task. The update action renames the task according to the input on the left. The delete action removes the task from the list.

Exercise 5.2 (Setting up the environment and project)

(a) Install *npm* as described in chapter **00-Organization**. To make sure it is installed correctly, execute `npm -v` in the terminal. You should see a version number. Install the *Vite* package globally using the command `npm install -g create-vite`.

(b) Create a new IntelliJ *module* `ex05` in your exercises repository. Use module type *Vite* with template *react*.

(c) Replace the initial contents of your project with the contents of the archive `ex05-incomplete.zip` (available on the ELO course).

(d) Familiarize yourself with the existing React components. The component `ListSelector.jsx` already implements the header of the application.

(e) Disable React *props validation* by adding the following line to the `rules` section of the file `eslint.config.js` (see lecture notes):
`'react/prop-types': 0`

(f) Start the backend application from Exercise sheet 2 in the background. If you haven't solved this exercise, you may download a sample solution on the ELO course.

(g) If required, set the correct port of your running backend in the file `src/Context.js`. The default port is 8080.

(h) Right-click on `package.json` in IntelliJ and select *Run 'npm install'*. This will install all required dependencies. You can also run the command `npm install` in the terminal.

(i) Open `package.json` in IntelliJ. Hit the *Run* button next to line 7 (`"dev": "vite",`). This will start the Vite development server. You can also run the command `npm run dev` in the terminal. Open your browser and navigate to `http://localhost:5173`. You should see an error indicating that the component `TodoList.jsx` is missing.

Exercise 5.3 (Frontend development with React)
Add the missing component `TodoList.jsx` for the main part of the application. Consider the following remarks for the implementation:

(a) The component depends on the `currentListId`.

(b) Use a *state hook* for the list itself (initially empty array).

(c) Use an *effect hook* to retrieve the contents of the current list.
The hook depends on `[api, currentListId]`.

(d) If no list is selected, render the contents as `<article><h1>No list selected.</h1></article>`.

(e) Otherwise, if a list is selected, produce elements according to the following template:

```
<article>
  <h1>{list.name}</h1>
  <ul>
    <li> <!-- repeat per to-do -->
      <div className="grid">
        <button>...</button>
        <input/>
        <button>...</button>
        <button>...</button>
    </li>
  </ul>
  <div className="grid">
    <input/>
    <button>...</button>
  </div>
</article>
```

(f) Register listeners and delegate to corresponding functions for adding, updating (= toggling state or renaming) and deleting to-dos. For the implementation of the three functions as well as the effect hook, use the `fetch` API (see `ListSelector.jsx`).

Test your frontend manually by creating a to-do list and performing the CRUD operations on the to-dos. Make sure the application behaves as expected.