DAT076 Rapport

walling, olofens, stride

March 2019

1 Introduction

The application created is a task-based time logger, where the primary functionality is to structure TODO-tasks with an estimated time, and successively toggle time to these task. Finally, the tasks can then be classified as done. The application has taken inspiration from the already existing, and widely popular application Toggl[2], but with the additional property of time estimation (Which then will be linked to statistics).

The long term goal of the application is to represent somewhat of a merge between previously mentioned Toggl, and the popular project planner Trello [1]. Instead of only being able to track time for existing tasks, the application instead takes the fundamentals of Trello, which is creating tasks that are yet to be done, thus allowing the user to plan ahead of tracking time.

2 Use cases

2.1 Create a task

- Press the "Add task" button
- Specify title, description and estimated time
- Press submit to finish

2.2 Edit a task

- Press the edit button in any task (located to the right)
- Edit title, description and/or estimated time
- Press submit to finish

2.3 Delete a task

- Press the delete button in any task

2.4 Start/stop timer

- If the task is in the left or right column, drag the task to the middle column
- Press the ”start timer” button
- When finished, press the ”stop timer” button

2.5 Move a task

- Press and hold any task
- Move the mouse while still holding onto the task
- Drop the task inside of the desired column, NOT on any existing task

3 User manual

The application presents the user with a view that consists all tasks, independent of their status. The view is divided into four parts: the header and the three columns. Each column represents the current state. To create a task, a user simply needs to press the blue button in the leftmost column, also known as the ToDo-column. Here, a modal panel opens up, which allows the user to specify a title, description and estimated time. Later on, a user can chose to edit or remove the task, by pressing the small icons on the task.

Now, the task appears in the leftmost column, and can be dragged to the remaining columns. Dragging a task to the middle column will enable the user to toggle the time tracker. This feature counts each second for which the task has been active. Pressing the same button again pauses the timer. When the current session is done, the user can drag the corresponding task to the rightmost column, also known as the Done-column. Here, the tasks which are considered done, are displayed. Any task that is in the done-column can be dragged back, if the user changes his/her mind.

4 Design

The application is built using a composition of React, Flask and SQL. React serves as a library for JavaScript code to build user interfaces. This, combined with flask connects the application to the database, which is written in SQL.

On a deeper level, Redux is used to structure the program into a MVC-like style. Redux is implemented as a global state machine, that stores the current state of the program. Any changes made in the outer components are passed to the reducer class, which then adjusts accordingly, and updates the global state. This then triggers the program to notify and update, which changes the interface.

The two most important classes are under Reducers and Actions. When an "action" is triggered by the components (such as an onClick method for example), the event is passed to the Action-class, which then passes information to the reducers. The reducers proceed to calculate a new state and updates the "store" (which can be seen as the global state). The store then updates all components with the new state, which makes them render any updated visuals.

When handling user input (as in the "Add/edit task" example), the library Formik, alongside Yup is used. As Formik's slogan suggests, it is used to "Build forms in React, without the tears". Formik simplifies forms input, as well as the submission itself. Yup is used after the input is collected to validate the data. Simple requirements and other constraints can be stacked, which greatly siplifies validation.

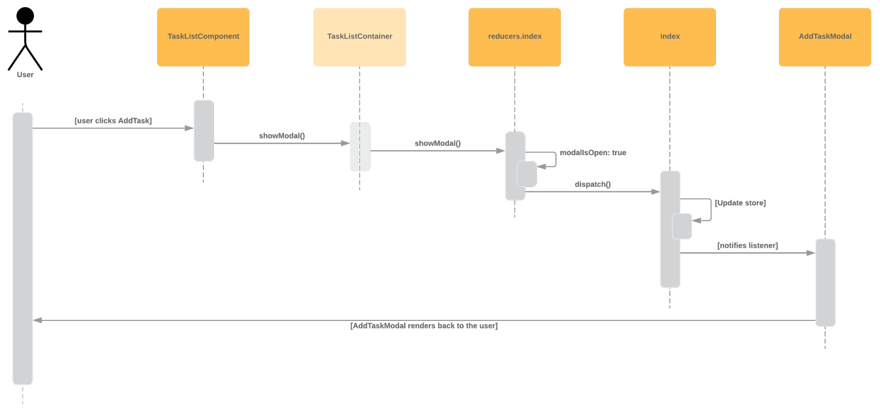
Jest, coupled with Enzyme was used to test the front-end. Jest is the most popular and preferred testing framework for React currently. Enzyme helps us mock our components in different ways so we can evaluate and assess their states.

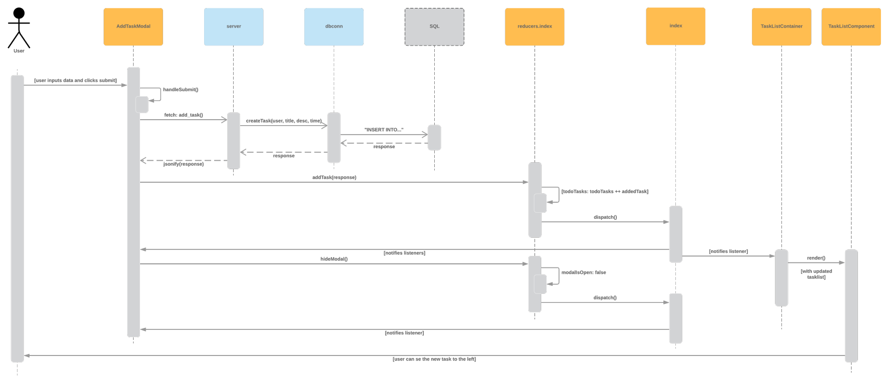
5 Application flow

5.1 Adding a task

Firstly, the user clicks on the AddTask button, which calls all the way back to reducers, where "showModal()" is executed. This updates the global state to where the AddTaskModal is shown. The user is then presented with an input form, which he/she can submit. When pressing the submit-button, a fetch is called to the server, which takes the input from the form, and calls a request to the database handler. This follows by an SQL query, sent from dbconn, that returns the task all the way back to AddTaskModal.

From here, "addTask" is called, which triggers the state to be updated with the new task. Furthermore, "hideModal()" is called, to hide the input form from the user. After these actions are finished, the global state is updated to show the newly created task. This finally makes the taskListComponent render the new Task.

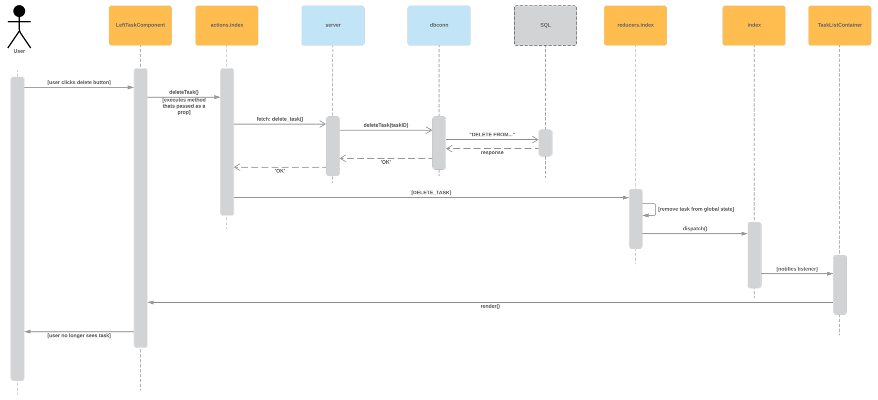




5.2 Delete a task

The user clicks the Delete button, which executes the database call (independent of which column the task is in). This fetches the request that deletes the specified task from the database. In server, deleteTask is called from dbconn, where dbconn executes the SQL query.

The reducer is then called, and removes the task from the global state. The store notifies all listeners that the state has been updated thereafter, which triggers the view to no longer display the deleted task.



6 Responsibilities

Erik: Database and Jest Olof: Architecture Gabriel: Tasks, Design, Timing and Report

References

[1] Atlassian. Trello application, 2019.

[2] Toggl OÜ. Toggl application, 2019.