|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Расулов Арсен ИУ5-35Б

Парадигмы и конструкции языков программирования

**ОТЧЁТ ПО**

**Домашнему заданию**

Москва

2023

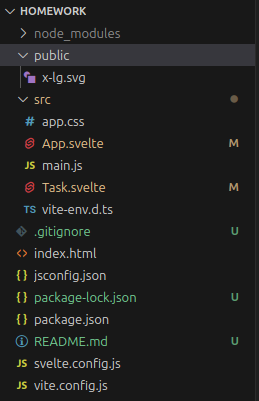
Задание

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Задача

Задачей является написание программы для организации списка дел с использованием неизвестного ранее фреймворка/технологии. В данном случае был выбран фреймворк Svelte, так как он является новым по сравнению с традиционными фреймворками (React, Vue, Angular).

Структура программы



* node\_modules — содержит модули node.js
* public — хранит общедоступные файлы, например изображения
* src — файлы исходного кода
* x-lg.svg — файл с крестом (для удаления задачи)
* index.html — файл html главной страницы, где будет использоваться svelte
* Все остальные файлы в корневой папке проекта конфигурационные
* main.js — JS скрипт, который запускает Svelte
* App.svelte — корневой для Svelte файл. Аналог main.cpp в мире Svelte.
* Task.svelte — компонент Task (Задача), который будет использоваться для задач в приложении
* app.css — файл с описанием стилей для всех компонентов Svelte.

Текст программы

## index.html

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Vite + Svelte ToDo App</title>

</head>

<body>

<!-- Элемент, в котором будет находиться приложение Svelte -->

<div id="app"></div>

<script type="module" src="/src/main.js"></script>

</body>

</html>

## main.js

import './app.css'

import App from './App.svelte'

const app = new App({

target: document.getElementById('app'),

})

export default app

## App.svelte

<script>

import Task from './Task.svelte'

let newTaskName = ''

let tasks = []

// Функция добавления новой задачи

function addNewTask() {

if (!newTaskName.trim())

return

tasks = [...tasks, newTaskName] // Нельзя просто использовать tasks.push(newTaskName), т.к. Svelte не поймет, что нужно обновить DOM

newTaskName = ''

}

</script>

<main>

<h1>Simple TODO App in Svelte</h1>

<!--Форма, в которой пользователь вводит текст новой задачи. Мы используем форму, чтобы при нажатии Enter автоматически нажималась кнопка и выполнялся on click ивент-->

<!--С помощью конструкции on: обрабатываем событие submit, параллельно используя модификатор обработчика событий preventDefault, который

помогает не обновлять страницу при отправке формы. -->

<form id="input-line" on:submit|preventDefault={() => {}}>

<!--Для того, чтобы иметь возможность использовать введеный пользователем текст задачи, используем binding в Svelte.

bind:value = {newTaskName} -- При изменении аттрибута value тега <input> меняется и newTaskname (название новой задачи).

То же самое работает и в обратную сторону. То есть при изменении newTaskName меняется value <input>-->

<input maxlength="30" class="task-input" placeholder="Enter your task here..." bind:value={newTaskName}/>

<button id="add-task-btn" on:click={addNewTask} hidden>Add</button>

</form>

<!--Выводим список задач, используя конструкцию фреймворка Svelte each-->

<ul id="tasks">

{#each tasks as task}

<Task taskText={task}/>

{/each}

</ul>

</main>

<style>

#input-line {

display: flex;

gap: 10px

}

.task-input {

display: inline-block;

width: var(--task-div-length);

height: 50px;

font-size: 40px;

font-family: Helvetica;

font-weight: 500;

border: 1px solid #4d4d4d;

padding: 0.5rem;

background-color: #242424;

}

#add-task-btn {

display: inline-block;

height: 65px;

width: 100px;

border-radius: 3px;

display: none;

}

#tasks {

gap:0;

margin:0;

padding:0;

}

</style>

## app.css

:root {

font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;

line-height: 1.5;

font-weight: 400;

color-scheme: light dark;

color: rgba(255, 255, 255, 0.87);

background-color: #242424;

font-synthesis: none;

text-rendering: optimizeLegibility;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

--task-div-length: 600px

}

body {

margin: 0;

display: flex;

place-items: center;

min-width: 320px;

min-height: 100vh;

}

h1 {

font-size: 3.2em;

line-height: 1.1;

}

#app {

max-width: 1280px;

margin: 0 auto;

padding: 2rem;

text-align: center;

}

## Task.svelte

<script>

export let taskText // Переменная, которую нужно заполнить при создании экземпляра Task. Это текст задачи.

let taskDone = false // Зачеркнутый ли стиль у текста нашей задачи? (применяется при нажатии на чекбокс).

let hideX = true // Вспомогательная переменная, отвечающая за появления кнопки удалить задачу рядом с задачей.

let deleteSelf = false // Была ли удалена текущая задача?

</script>

<!-- Конструкция class:название\_класса={true/false} позволяет применить определенный класс на какой-либо тег HTML.

В данном случае мы используем его для скрытия задачи, если она была удалена.-->

<main class:hidden={deleteSelf}>

<!-- Так как события on:hover нет в Svelte, то пришлось использовать on:mouseover и mouseout, что выполняет ту же самую работу.

Когда мышка над задачей, то справа от нее должен появиться крест, чтобы пользователь мог удалить ее. Поэтому при наведении делаем

hideX = false, а когда мышь не на элементе -- hideX = true. -->

<div id="task-n-cross" on:mouseover={() => hideX = false} on:mouseout={() => hideX = true}>

<div id="content-in-box">

<!-- Если задача выполнена, то применяем класс strike к <p> -->

<p class:strike={taskDone}>{taskText}</p>

<!-- Если на чекбокс нажали, то задача выполнена => taskDone = true. Если снова нажали, то отменяем действие.

Для обработки этого события используем конструкцию on:событие-->

<input type="checkbox" on:click={() => taskDone = !taskDone}>

</div>

<!--Создаем button, внутри которой находится img. Для обработки события on:click используем button (Иначе нельзя).-->

<button id="transparent-btn" class:hidden={hideX} on:click={() => deleteSelf = true}>

<img id="x-img" src="./public/x-lg.svg" alt="delete task">

</button>

</div>

</main>

<!--Стили для этого компонента-->

<style>

#content-in-box {

display: flex;

justify-content: space-between;

align-items: center;

width: var(--task-div-length);

height: 50px;

font-size: 40px;

font-family: Helvetica;

font-weight: 500;

border-radius: 5px;

border: 1px solid #4d4d4d;

padding: 0.5rem;

background-color: #242424;

margin: 0;

border-radius: 0;

}

#content-in-box:hover {

background-color: #333333;

transition: background-color 0.3;

}

input {

min-width: 50px;

max-width: 50px;

height: 50px;

}

.strike {

text-decoration: line-through;

}

#x-img {

width: 40px;

height: 40px;

}

.hidden {

display: none;

}

#transparent-btn {

background: none;

border: none;

margin: 0;

padding: 0;

position: absolute;

right: -20px;

}

#transparent-btn:hover {

filter: invert(1);

transition: 0.3s all;

}

#task-n-cross {

display: flex;

gap: 15px;

align-items: center;

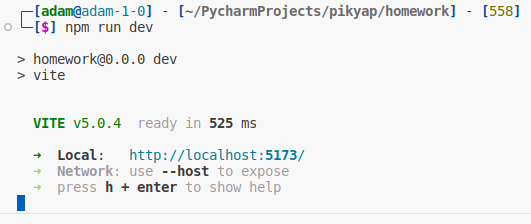
position: relative;

}

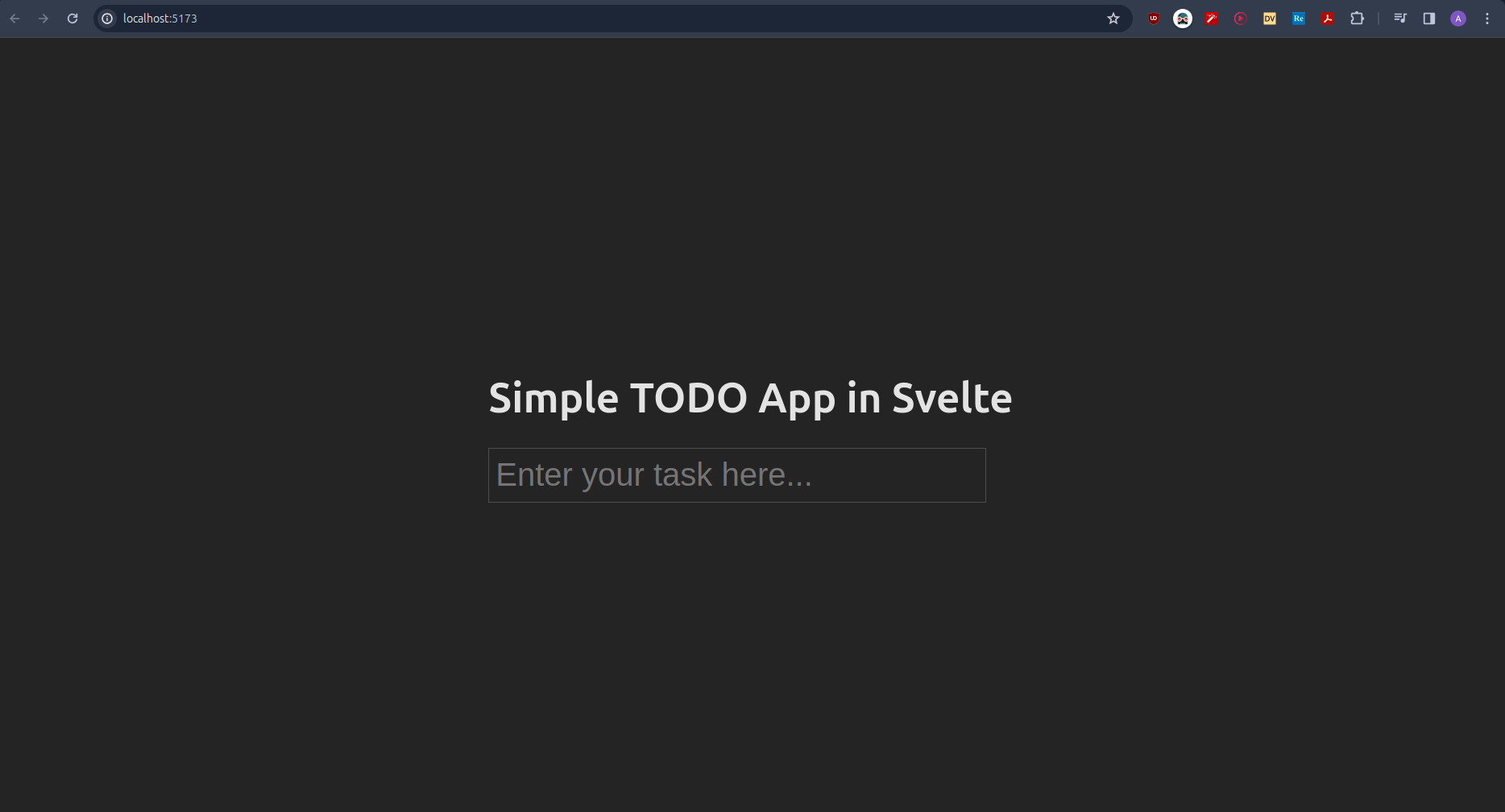
</style>

Экранные формы

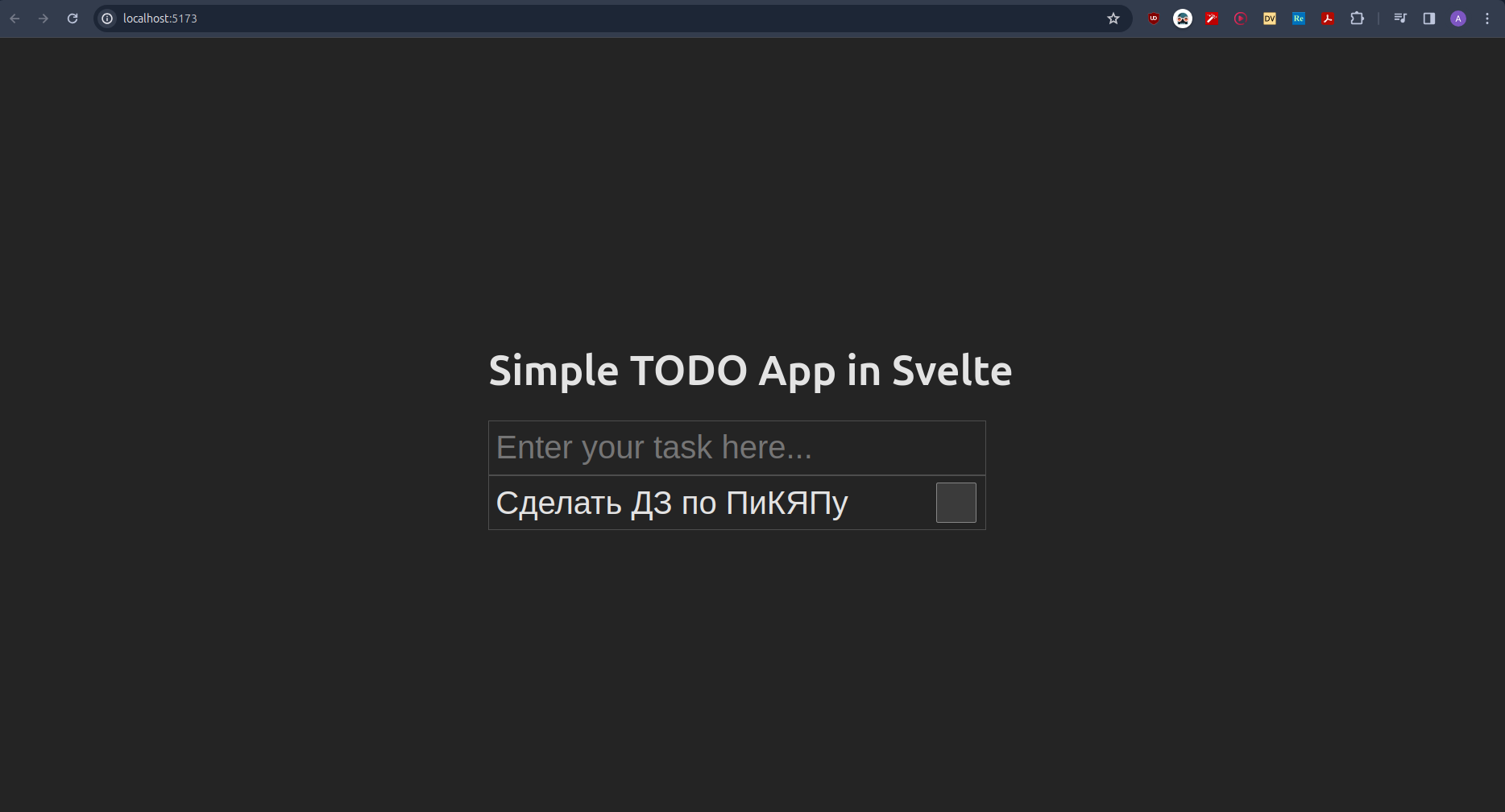
Запуск проекта с помощью Vite — инструмента для разработки фронтэнд приложений.



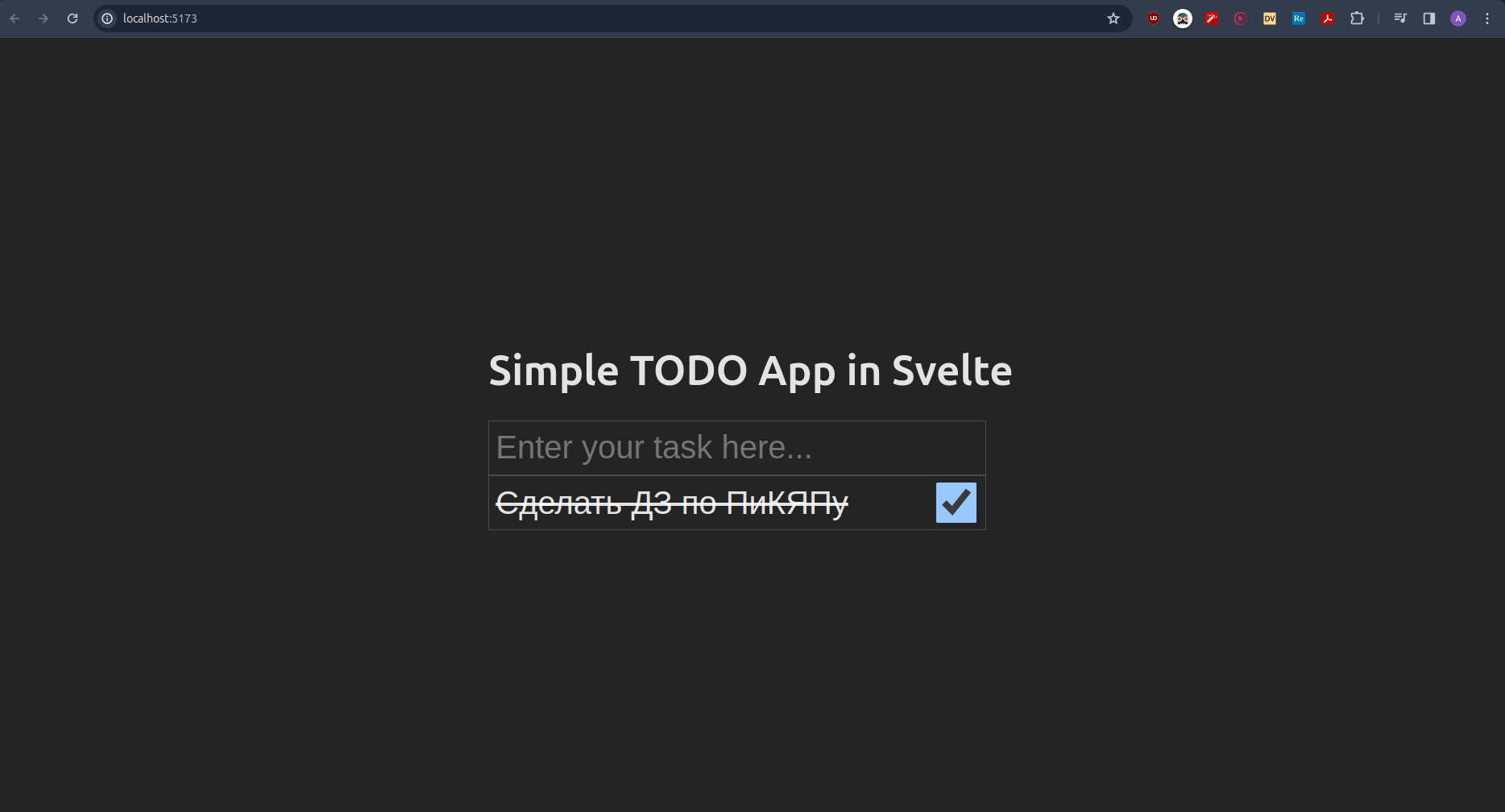
Приложение в браузере:



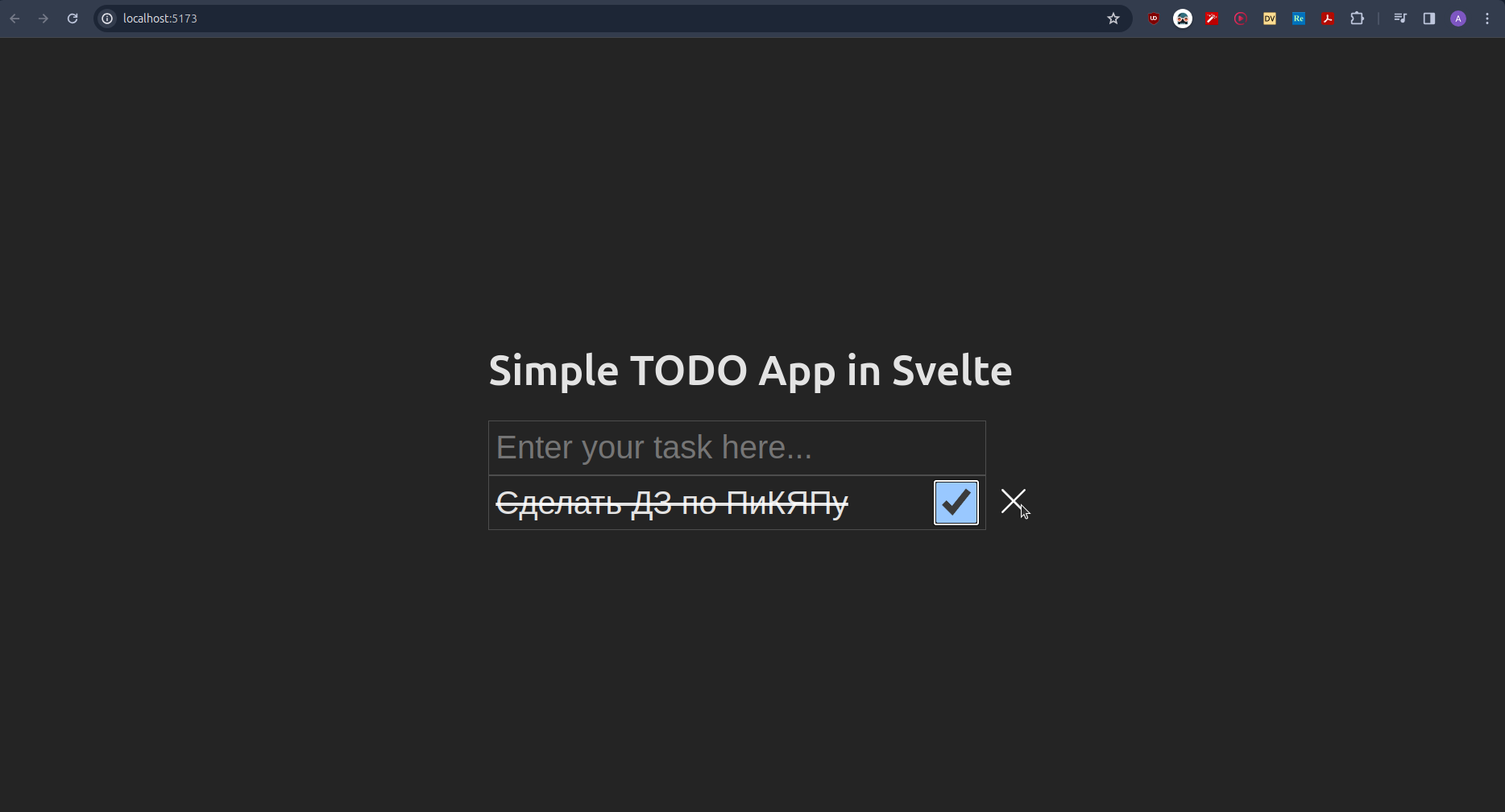
Добавил одну задачу



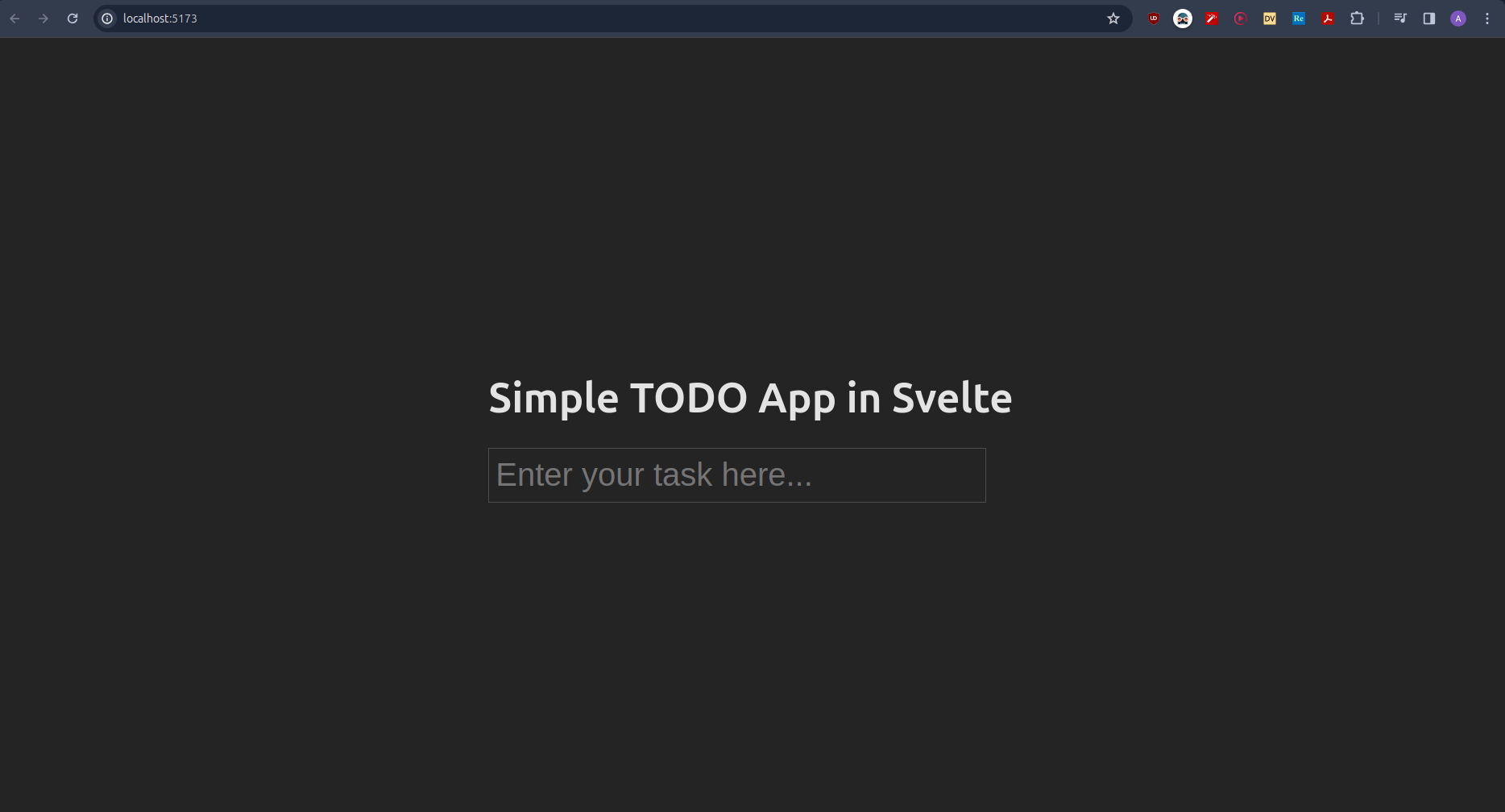
Пометил задачу как выполненную:



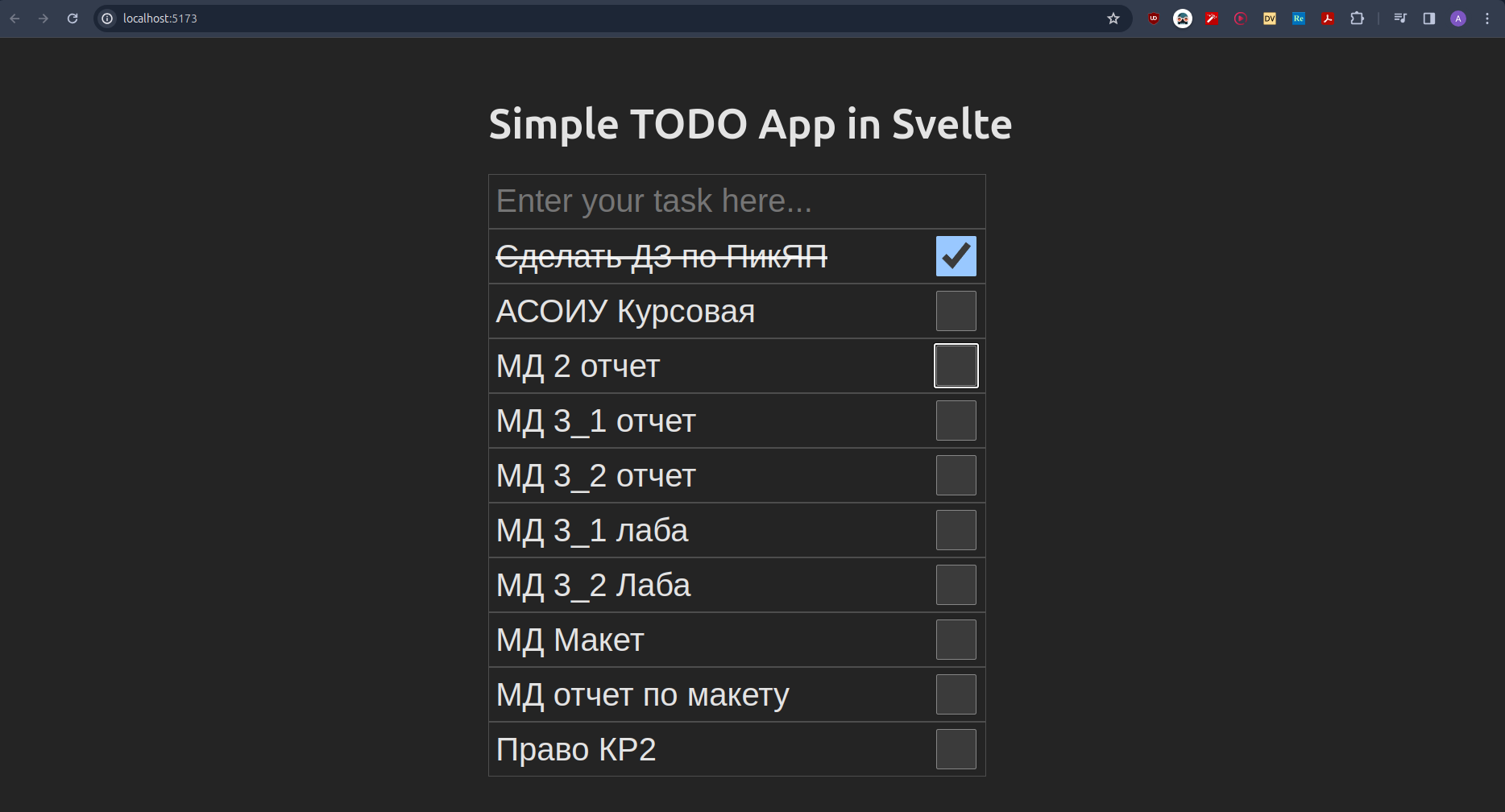
Удаление задачи:



Результат удаления:



Удаление с несколькими задачами:



Результат:

