

Начало работы над приложением Todo list

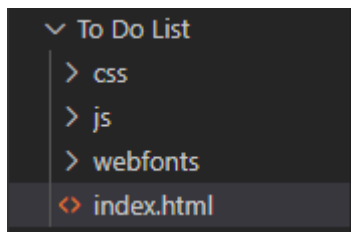
Создать папку проекта.

Файл index.html

Папка css – файлы из bootstrap/css и fontawesome/css

Папка js – файлы из bootstrap/js и vue.js

Папка webfonts из fontawesome



Далее создаем базовый html

```
To Do List > <> index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Todo List</title>
8      <link rel="stylesheet" href="css/bootstrap.min.css">
9      <link rel="stylesheet" href="css/all.min.css">
10 </head>
11 <body>
12     <div id="app"></div>
13
14     <script src="js/bootstrap.bundle.min.js"></script>
15     <script src="js/vue.js"></script>
16 </body>
17 </html>
```

Здесь мы подключили стили bootstrap, иконки fontawesome, скрипты bootstrap и vue.

Также создали элемент #app, куда рендерится приложение.

Зададим body фон с помощью класса bg-light.

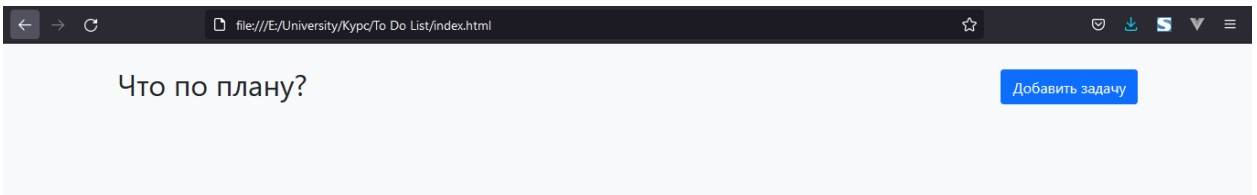
Создаем базовый контейнер с вертикальным отступом, заголовок и кнопку добавить. Используем классы bootstrap:

```

<body class="bg-light">
  <div id="app">
    <div class="container py-4">
      <div class="d-flex justify-content-between align-items-center mb-4">
        <h2>Что по плану?</h2>
        <button class="btn btn-primary">Добавить задачу</button>
      </div>
    </div>
  </div>

  <script src="js/bootstrap.bundle.min.js"></script>
  <script src="js/vue.js"></script>
</body>

```

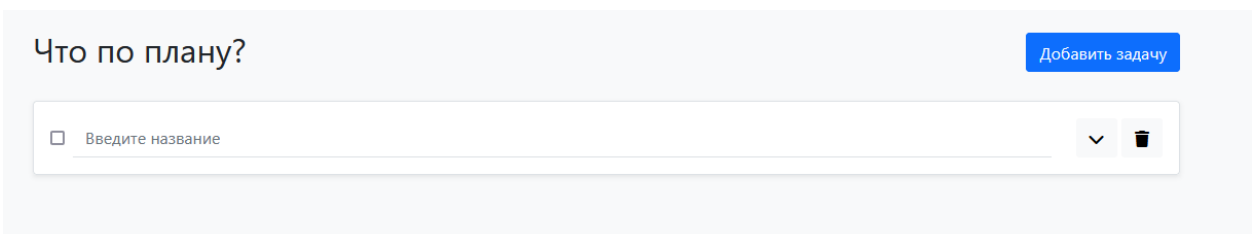


После заголовка и кнопки сделаем элемент задачи. Здесь как раз используем иконки.

```

<div id="app">
  <div class="container py-4">
    <div class="d-flex justify-content-between align-items-center mb-4">
      <h2>Что по плану?</h2>
      <button class="btn btn-primary">Добавить задачу</button>
    </div>
    <div class="border bg-white shadow-sm rounded p-3">
      <div class="row align-items-center">
        <div class="col d-flex">
          <input type="checkbox">
          <input type="text" class="form-control border-0 border-bottom rounded-0 ms-2" placeholder="Введите название">
        </div>
        <div class="col-auto">
          <button class="btn btn-light">
            <i class="fas fa-chevron-down"></i>
          </button>
          <button class="btn btn-light">
            <i class="fas fa-trash"></i>
          </button>
        </div>
      </div>
    </div>
  </div>
</div>

```



Добавим форму редактирования даты и времени

```

<div class="border bg-white shadow-sm rounded p-3">
  <div class="row align-items-center">
    <div class="col d-flex">
      <input type="checkbox">
      <input type="text" class="form-control border-0 border-bottom rounded-0 ms-2" placeholder="Введите название">
    </div>
    <div class="col-auto">
      <button class="btn btn-light">
        <i class="fas fa-chevron-down"></i>
      </button>
      <button class="btn btn-light">
        <i class="fas fa-trash"></i>
      </button>
    </div>
  </div>
  <div>
    <div class="row pt-4">
      <div class="col-6">
        <input type="date" class="form-control">
      </div>
      <div class="col-6">
        <input type="time" class="form-control">
      </div>
    </div>
  </div>
</div>

```

Что по плану? Добавить задачу

☐ Введите название
▼
🗑

ДД . ММ . ГГГГ

--:--

А теперь сделаем, чтобы эта форма была раскрывающейся. Модифицируем код кнопки со стрелкой так, добавив класс `collapsed` и атрибуты `data-bs-toggle="collapse"` и `data-bs-target="#task-1"`.

```

<div class="col-auto">
  <button class="btn btn-light collapsed" data-bs-toggle="collapse" data-bs-target="#task-1">
    <i class="fas fa-chevron-down"></i>
  </button>
  <button class="btn btn-light">
    <i class="fas fa-trash"></i>
  </button>
</div>

```

И блока с формой – добавили класс `collapse` и `id="task-1"`:


```

<div class="collapse" id="task-1">
  <div class="row pt-4">
    <div class="col-6">
      <input type="date" class="form-control">
    </div>
    <div class="col-6">
      <input type="time" class="form-control">
    </div>
  </div>
</div>

```

Здесь использовали Bootstrap collapse. Атрибут data-target указывает, какой элемент триггерит кнопка. В данном случае это элемент #task-1. Подробнее, как это работает, можно почитать по ссылке <https://getbootstrap.com/docs/5.0/components/collapse/>

Добавим стиль для стрелки, чтобы она крутилась. Создадим файл custom.css в папке css и подключим. Добавим такой код:

```
To Do List > css > # custom.css >  button:not(.collapsed) .fa-chevron-down
1  button .fa-chevron-down {
2      transition: .4s;
3  }
4  button:not(.collapsed) .fa-chevron-down {
5      transform: rotate(-180deg);
6  }
```

Кнопка при открытии блока теряет класс collapsed. За этим можно понаблюдать, переключая ее.

С версткой пока закончили.

Создаем объект Vue:

```
<script src="js/bootstrap.bundle.min.js"></script>
<script src="js/vue.js"></script>
<script>
  var app = new Vue({
    el: '#app',
    data: {
      tasks: []
    }
  })
</script>
body>
```

Пока из данных у нас пустой массив задач. Скоро задач прибавится.

Сразу создадим методы добавления и удаления, потом приделаем их к кнопкам:

```

var app = new Vue({
  el: '#app',
  data: {
    tasks: []
  },
  methods: {
    addTask: function() {
      this.tasks.unshift({
        completed: false,
        title: '',
        date: '',
        time: ''
      })
    },
    deleteTask: function(index) {
      this.tasks.splice(index, 1)
    }
  }
})

```

Метод addTask добавляет В НАЧАЛО МАССИВА объект с полями completed, title, date, time.

Метод deleteTask удаляет из массива по индексу.

Теперь привяжем массив к элементам верстки. Используем директиву v-for:

```

<div class="border bg-white shadow-sm rounded p-3" v-for="(task, index) in tasks" :key="index">
  <div class="row align-items-center">
    <div class="col d-flex">
      <input type="checkbox" v-model="task.completed">
      <input type="text" class="form-control border-0 border-bottom rounded-0 ms-2" placeholder="Введите название" v-model="task.title">
    </div>
    <div class="col-auto">
      <button class="btn btn-light collapsed" data-bs-toggle="collapse" :data-bs-target="`#task-${index}`">
        <i class="fas fa-chevron-down"></i>
      </button>
      <button class="btn btn-light">
        <i class="fas fa-trash"></i>
      </button>
    </div>
  </div>
  <div class="collapse" :id="`task-${index}`">
    <div class="row pt-4">
      <div class="col-6">
        <input type="date" class="form-control" v-model="task.date">
      </div>
      <div class="col-6">
        <input type="time" class="form-control" v-model="task.time">
      </div>
    </div>
  </div>
</div>
</div>

```

Здесь главному блоку добавили вывод по циклу, который пробегается по всем элементам массива:

```
v-for="(task, index) in tasks" :key="index"
```

полям ввода сразу назначили v-model:

```

v-model="task.completed"
v-model="task.title"
v-model="task.date"

```

```
v-model="task.time"
```

И подвязали collapse через динамические атрибуты. У нас в примере было task-1, здесь мы динамически подставим индекс, чтобы кнопка соответствовала блоку.

```
data-bs-toggle="collapse" :data-bs-target="#task-${index}"  
class="collapse" :id="task-${index}"
```

***можно было лепить строки и так:*

```
:id="'task-' + index"
```

Но мы будем использовать шаблон строк, так как он более удобный и меньше вероятность ошибиться с кавычками. Синтаксис получается такой:

```
let str = `три плюс два равно ${3 + 2}`
```

Подробнее <https://dev-gang.ru/article/-sposoba-obedinenija-strok-v-javascript-prgtacj4yl/>

Сейчас у нас ни одного элемента нет, ведь массив пуст. Правда, в начале может мелькать на странице. Добавим блоку #app атрибут v-cloak:

```
<div id="app" v-cloak>
```

И в custom.css добавим:

```
*[v-cloak] {  
  display: none;  
}
```

Теперь элемент #app будет видим только после отрисовки контента.

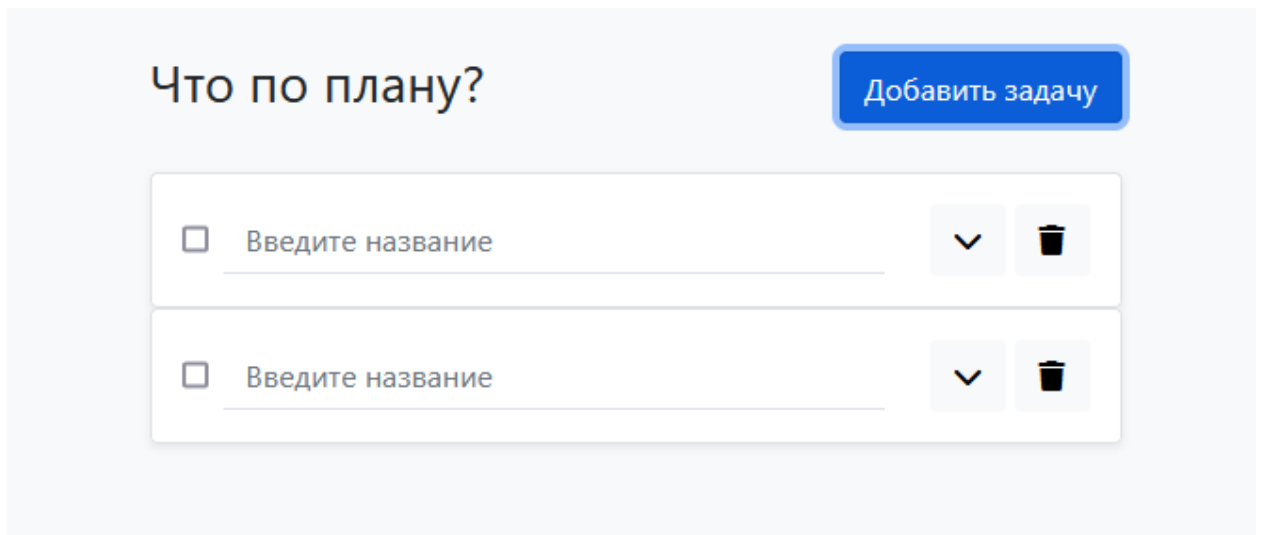
Вывод списка есть, надо научить добавлять задачи. Повесим обработчик на кнопку:

```
<div class="d-flex justify-content-between align-items-center mb-4">  
  <h2>Что по плану?</h2>  
  <button class="btn btn-primary" @click="addTask">Добавить задачу</button>  
</div>
```

Добавили

```
@click="addTask"
```

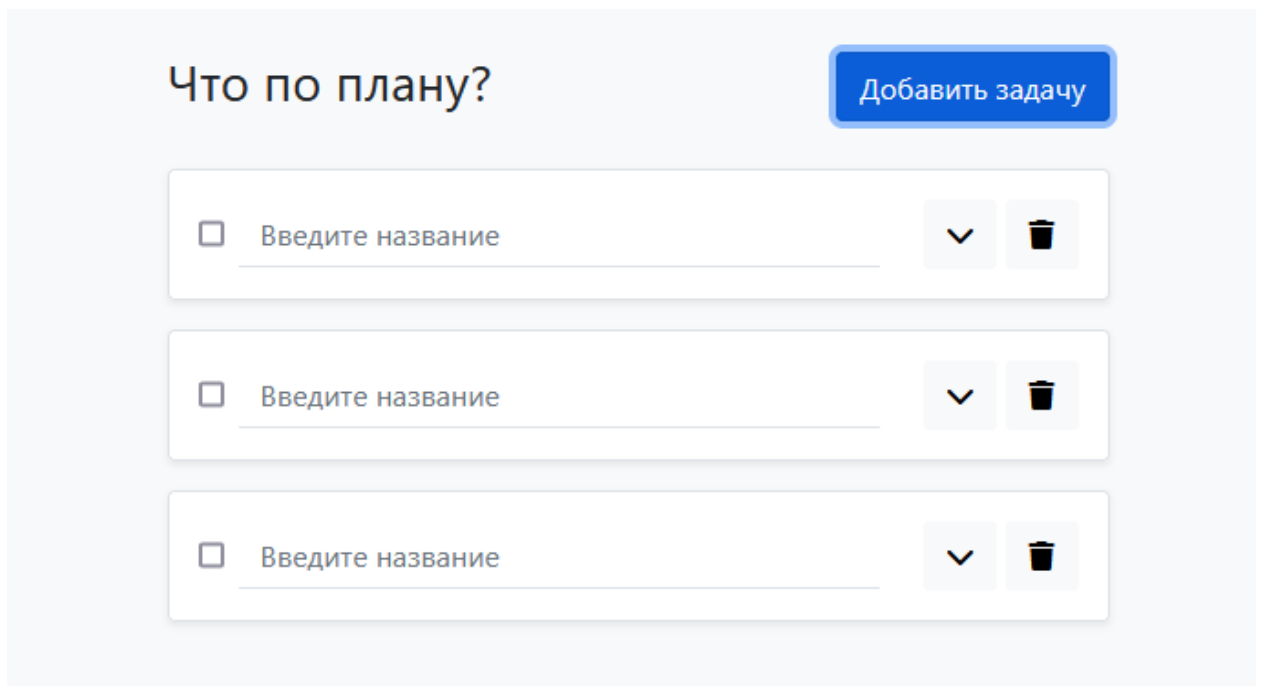
Теперь при нажатии на кнопку добавляются новые пункты:



Сделаем, чтобы не прилипали. Добавим класс `.mb-3`:

```
<div class="border bg-white shadow-sm rounded p-3 mb-3" v-for="(task, index) in tasks" :key="index">
```

Стало лучше:



Теперь реализуем удаление по кнопке корзины:

```
<button class="btn btn-light" @click="deleteTask(index)">
  <i class="fas fa-trash"></i>
</button>
```

Добавили

```
@click="deleteTask(index)"
```

Теперь можно добавлять и удалять.

Добавим иконку для кнопки Добавить:

```
<button class="btn btn-primary" @click="addTask">
  <i class="fa-solid fa-plus"></i>
  Добавить задачу
</button>
```

После блока .col добавим блок .col-auto для отображения даты и времени, если они указаны. Дата приводится к нормализованному виду.

```
<div class="row align-items-center">
  <div class="col d-flex">
    <input type="checkbox" v-model="task.completed">
    <input type="text"
      v-model="task.title"
      class="form-control border-0 rounded-0 border-bottom ms-2"
      placeholder="Введите название">
  </div>
  <div class="col-auto v-if="task.date || task.time">
    До
    <span v-if="task.date">
      {{ new Date(task.date).toLocaleDateString() }}
    </span v-if="task.time">
    <span>{{ task.time }}</span>
  </div>
  <div class="col-auto">
    <button class="btn btn-light collapsed" data-bs-toggle="collapse" :data-bs-target="#task-${index}">
      <i class="fa-solid fa-chevron-down"></i>
    </button>
    <button class="btn btn-light" @click="deleteTask(index)">
      <i class="fa-solid fa-trash-can"></i>
    </button>
  </div>
</div>
```

Мои задачи + Добавить задачу

☐ Написать диплом

До 12.05.2023

▼

🗑

☐ Сдать практику

▼

🗑

☐ Сделать задание

До 27.02.2022 12:00

▼

🗑

☒ Подготовить презентацию

До 18.02.2022 20:20

▼

🗑

☒ Составить курс

До 03.02.2022 00:00

▼

🗑

Теперь сделаем текст красным, если задача просрочена. Используем динамические классы:

```
<div class="col-auto v-if="task.date || task.time" :class="{ 'text-danger': checkTime(task)}">
  До
  <span v-if="task.date">
    {{ new Date(task.date).toLocaleDateString() }}
  </span v-if="task.time">
  <span>{{ task.time }}</span>
</div>
```


Класс text-danger будет добавлен, если checkTime возвращает true. Напишем этот метод:

```
methods: {
  addTask: function() {
    this.tasks.unshift({
      title: 'Новая задача',
      date: '',
      time: '',
      completed: false
    })
  },
  deleteTask: function(index) {
    this.tasks.splice(index, 1)
  },
  checkTime: function(task) {
    if(task.completed) return false
    let taskDate = task.date ||
    (
      new Date().getFullYear() + '-' +
      (new Date().getMonth() + 1) + '-' +
      new Date().getDate()
    )
    let dateStr = (taskDate + ' ' + task.time).trim()
    return new Date(dateStr).getTime() < Date.now()
  }
}
```

Он сверяет дату дедлайна по задаче с текущей. При этом, если дата не назначена, берется текущая. Обратите внимание, getMonth() возвращает номера месяцев, начиная с нуля.

☐ Составить курс

До 03.02.2022 10:00

▼

🗑

Теперь сделаем автосохранение. Напишем метод-наблюдатель:

```
data: {
  tasks: []
},
watch: {
  tasks: {
    deep: true,
    handler: function() {
      localStorage.setItem('tasks', JSON.stringify(this.tasks))
    }
  }
},
methods: {
```

При изменении даже внутренних свойств данные сериализуются в JSON и сохраняются в локальное хранилище браузера.

И загрузка при первоначальной отрисовке:

```
    },  
    mounted() {  
      if(localStorage.getItem('tasks')) {  
        this.tasks = JSON.parse(localStorage.getItem('tasks'))  
      }  
    }  
  }  
})
```

Теперь при перезагрузке страницы данные должны оставаться.

Теперь сверстаем форму поиска после блока с заголовком и кнопкой:

```
<div class="d-flex justify-content-between align-items-center mb-4">  
  <h2>Мои задачи</h2>  
  <button class="btn btn-primary" @click="addTask">  
    <i class="fa-solid fa-plus"></i>  
    Добавить задачу  
  </button>  
</div>  
<div class="mb-4">  
  <input type="text" class="form-control" placeholder="Поиск по задачам..." v-model="searchText">  
</div>
```

И добавим свойство searchText, которое связано с этим полем:

```
data: {  
  tasks: [],  
  searchText: ''  
},
```

Поиск реализуем с помощью вычисляемого свойства filteredTasks. Создадим его:

```
data: {  
  tasks: [],  
  searchText: ''  
},  
computed: {  
  filteredTasks: function() {  
    return this.tasks.filter(task => task.title.toLowerCase().includes(this.searchText.toLowerCase()))  
  },  
}
```

Это массив, отфильтрованный по условию: название содержит строку запроса, при этом сравнение без учета регистра.

Еще переделаем логику с ключами, чтобы индексы работали правильно. Заведем поле id при создании задачи. Чтобы оно было уникальным, будем присваивать ему текущую дату в мс).

ПРЕДВАРИТЕЛЬНО УДАЛИМ ВСЕ ЗАДАЧИ и перепишем метод addTask и метод deleteTask:

```

methods: {
  addTask: function() {
    this.tasks.unshift({
      id: Date.now(),
      title: 'Новая задача',
      date: '',
      time: '',
      completed: false
    })
  },
  deleteTask: function(id) {
    this.tasks.splice(this.tasks.findIndex(task => task.id == id), 1)
  },
}

```

В шаблоне заменим tasks на filteredTasks и заменим index на task.id:

```

<div class="border bg-white shadow-sm p-3 mb-3" v-for="task in filteredTasks" :key="task.id">
  <div class="row align-items-center">
    <div class="col d-flex">
      <input type="checkbox" v-model="task.completed">
      <input type="text"
        v-model="task.title"
        class="form-control border-0 rounded-0 border-bottom ms-2"
        placeholder="Введите название">
    </div>
    <div class="col-auto v-if="task.date || task.time" :class="{ 'text-danger': checkTime(task)}">
      До
      <span v-if="task.date">
        {{ new Date(task.date).toLocaleDateString() }}
      </span>
      <span v-if="task.time">
        <span>{{ task.time }}</span>
      </span>
    </div>
    <div class="col-auto">
      <button class="btn btn-light collapsed" data-bs-toggle="collapse" :data-bs-target="#task-${task.id}">
        <i class="fa-solid fa-chevron-down"></i>
      </button>
      <button class="btn btn-light" @click="deleteTask(task.id)">
        <i class="fa-solid fa-trash-can"></i>
      </button>
    </div>
  </div>
  <div class="collapse" :id="`task-${task.id}`">
    <div class="row pt-4">
      <div class="col-6">
        <input type="date" class="form-control" v-model="task.date">
      </div>
      <div class="col-6">
        <input type="time" class="form-control" v-model="task.time">
      </div>
    </div>
  </div>
</div>

```

Теперь можно искать задачи по названию:

Мои задачи

+ Добавить задачу

☐ Сдать практику

▼

🗑

☐ Сделать задание

До 27.02.2022 12:00

▼

🗑

Сделаем отдельный блок для выполненных задач. Добавим еще 2 вычисляемых свойства `activeTasks` и `completedTasks`:

```
computed: {
  filteredTasks: function() {
    return this.tasks.filter(task => task.title.toLowerCase().includes(this.searchText.toLowerCase()))
  },
  activeTasks: function() {
    return this.filteredTasks.filter(task => !task.completed)
  },
  completedTasks: function() {
    return this.filteredTasks.filter(task => task.completed)
  }
},
```

Это будут активные и выполненные задачи.

Добавим в шаблон блок сразу под циклом с задачами:

```
<template v-if="completedTasks.length">
  <hr class="mt-4">
  <h3 class="mb-4">Выполненные</h3>
  <div class="border bg-white p-3 mb-3 text-decoration-line-through text-muted" v-for="task in completedTasks" :key="task.id">
    <div class="row align-items-center">
      <div class="col d-flex">
        <input type="checkbox" v-model="task.completed">
        <div class="ms-2">{{ task.title }}</div>
      </div>
      <div class="col-auto v-if="task.date || task.time">
        до
        <span v-if="task.date">
          {{ new Date(task.date).toLocaleDateString() }}
        </span>
        <span v-if="task.time">
          <span>{{ task.time }}</span>
        </span>
      </div>
      <div class="col-auto">
        <button class="btn btn-light" @click="deleteTask(task.id)">
          <i class="fa-solid fa-trash-can"></i>
        </button>
      </div>
    </div>
  </div>
</template>
```

Теперь, если есть выполненные задачи, то они будут отображаться ниже:

Мои задачи

+ Добавить задачу

Поиск по задачам...

☐ Добиться успешного успеха

▼

🗑

☐ Не отчислиться

▼

🗑

Выполненные

☒ Поступить в вуз

🗑

Доработаем UX. Обработаем случай, когда ни одной задачи нет. В этом случае отображаем какой-нибудь текст. Добавим его после блоков с задачами:

```
<div class="border bg-white shadow-sm p-3 mb-3" v-for="task in activeTasks" :key="task.id">...
</div>
<template v-if="completedTasks.length">...
</template>
<div v-if="!tasks.length" class="p-5 text-center h4 bg-white shadow-sm">
  Ура, делать ничего не надо 🎉
</div>
```

Мои задачи

+ Добавить задачу

Поиск по задачам...

Ура, делать ничего не надо 🎉

Мои задачи

+ Добавить задачу

Ура, делать ничего не надо 🎉

Можно, в принципе, и не показывать поле поиска, если ничего нет. Добавим v-if:

```
<div class="mb-4" v-if="tasks.length">
  <input type="text" class="form-control" placeholder="Поиск по задачам..." v-model="searchText">
</div>
```

Мои задачи

+ Добавить задачу

Ура, делать ничего не надо 🎉

Теперь что если задачи есть, но под условия поиска ни одна не подходит?
Добавим блок с условием, что задачи есть, но нет (`tasks.length > 0` и `filteredTasks == 0`). Шаблон примет следующий вид:

```
<div v-if="!tasks.length" class="p-5 text-center h4 bg-white shadow-sm">
  Ура, делать ничего не надо 🎉
</div>
<div v-else-if="!filteredTasks.length" class="p-5 text-center h4 bg-white shadow-sm">
  Мы искали, но не нашли 😞
</div>
```

Мои задачи

+ Добавить задачу

ты точно ничего не найдешь

Мы искали, но не нашли 😞

Еще сделаем стирание строки поиска при добавлении нового элемента. Потому что сейчас, если у нас задано условие поиска и мы добавляем новый элемент, мы его не видим, что неудобно юзеру.

```
addTask: function() {
  this.tasks.unshift({
    id: Date.now(),
    title: 'Новая задача',
    date: '',
    time: '',
    completed: false
  })
  this.searchText = ''
},
```

Теперь добавим анимацию списка. Обернем цикл по блокам в элемент `<transition-group>` и сделаем ему атрибуты `name="appear"` и `tag="div"`:

```
<transition-group name="appear" tag="div">
  <div class="border bg-white shadow-sm p-3 mb-3" v-for="task in activeTasks" :key="task.id">...
</div>
</transition-group>
```

И добавим стили в CSS:

```
.appear-enter, .appear-leave-to {  
  transform: translateX(-100%);  
  opacity: 0;  
}  
.appear-enter-active, .appear-leave-active {  
  transition: .4s;  
}
```

Теперь добавление и удаление должно быть анимировано. Аналогично пропишем анимацию списку выполненных задач:

```
<template v-if="completedTasks.length">  
  <hr class="mt-4">  
  <h3 class="mb-4">Выполненные</h3>  
  <transition-group name="appear" tag="div">  
    <div class="border bg-white p-3 mb-3 text-decoration-line-through text-muted" ...  
  </div>  
  </transition-group>  
</template>
```