22. Учебный проект: меняй-удаляй (часть 1)

Рабочая ветка **module7-task1**

Задача

Пришло время расширить функциональность нашего приложения. Сегодня нам предстоит решить сразу несколько задач: реализовать фильтрацию и разобраться, как создавать и удалять задачи.

Модель данных

Прежде, чем мы начнём реализовывать основную функциональность, нам нужно ввести в наше приложение модель данных для синхронизации задач между различными частями приложения.

1. Создайте директорию `/src/models` с новым файлом `tasks.js`, в котором опишите класс `Tasks`.

2. Добавьте в класс 2 метода: один для получения задач, другой для их записи.

3. Добавьте ещё один метод для обновления конкретной задачи. Он должен принимать два параметра: `id` обновляемой задачи (если для этого понадобится добавить поле `id` в моковые данные, то сделайте это) и обновлённую задачу. Реализацию метода

заберите из `BoardController`.

4. В `main.js` создайте экземпляр модели и передайте в неё, с помощью созданного на втором шаге метода записи, моковые данные.

5. В `main.js` передайте в конструктор `BoardController` модель, а передачу моковых данных в метод `render` — удалите.

6. В `BoardController` замените работу с массивом задач на работу с моделью: для получения и обновления данных используйте соответствующие методы модели.

Фильтрация

В этой части задания мы запрограммируем фильтры.

1. На первом шаге классика — компонент фильтров у нас уже есть, нужен контроллер для него. С конструктором и методом `render`. Конструктор принимает контейнер и модель данных. После создания экземпляра контроллера в `main.js`, вызовите у него метод `render` для отрисовки.

2. Для реализации фильтрации модернизируйте модель: добавьте в неё метод для активации фильтра. Измените метод модели для получения данных, чтобы он учитывал

активный фильтр. Активируйте фильтр в модели при взаимодействии пользователя с интерфейсом.

3. Но как `BoardController` будет узнавать об изменении активного фильтра? Подход уже вам известен:

   - в модель добавьте метод для установки обработчика изменения активного фильтра;

   - при активации фильтра в модели вызовите установленный обработчик;

   - в `BoardController` передайте ей обработчик, который будет получать данные от модели и вызывать перерисовку.

4. Осталось реализовать обновление счётчика у фильтров при изменении, а в будущем добавление и удаление, данных:

   - в модель добавьте метод для установки обработчика изменения данных;

   - в контроллере фильтров передайте в модель обработчик, пересчитывающий количество задач по каждому фильтру и вызывающий перерисовку

компонента фильтров.

Обратите внимание, при переключении фильтров должны сбрасываться сортировка и счётчик показанных задач («Load more»). Подробности в техническом задании.

Удаление и добавление данных

Удаление и добавление задач можно реализовать разными способами. Предлагаем один из самых простых. Наша доска уже умеет перерисовываться при изменении данных. Значит для того, чтобы удалить задачу с доски, достаточно изменить данные (удалить из них конкретную задачу).

1. Для удаления научим функцию `onDataChange` принимать в качестве обновлённых данных `null`. Логика следующая: если вместо новых данных пришёл `null`, то старые данные нужно удалить.

2. Дальше добавим в компонент формы редактирования обработчик события `click` по кнопке удаления, где вызовем функцию `onDataChange` и передадим ей в качестве новых данных `null`.

3. Теперь провернём всё то же самое, только для добавления — то есть `null` должен приходить в `onDataChange` вместо старых данных, а новые данные должны добавляться в модель (реализуйте в модели

соответствующие методы).

4. Затем нам нужно добавить обработчик на кнопку «Add new task», по клику на которую нужно показать форму добавления новой задачи. В качестве формы добавления нужно использовать форму редактирования, поэтому должно сохраниться всё поведение контролов и валидации. Кроме того, добавьте условие, чтобы можно было открыть только одну форму добавления.

Обратите внимание, при попытке пользователя добавить новую задачу, должны сбрасываться фильтры, сортировка и счётчик показанных задач («Load more»); скрываться без сохранения любая показанная форма редактирования. Подробности в техническом задании.

5. В конце останется лишь обработать отправку формы.

Обратите внимание, при закрытии формы добавления (любым способом) введённая информация не сохраняется.

Безопасность превыше всего

Мы отлично поработали! Данные в нашем приложении создаются, удаляются, изменяются... пользователями. А где есть пользовательский ввод, там потенциальная дыра в безопасности. Поэтому пора подумать о том, как обезопасить приложение и добропорядочных пользователей от «хакеров».

Установите из npm и подключите в проект библиотеку для превращения в строку возможного HTML или JavaScript кода в пользовательском вводе. Мы рекомендуем библиотеку `he`.

Обработайте с помощью этой библиотеки описание задачи.