Чек-лист по самопроверке №2

В этом документе описаны критерии, которым должна соответствовать работа. Перед отправкой работы на ревью, убедитесь, что она соответствует всем критериям.

Работа отклоняется от проверки

Если не соблюсти хотя бы один из критериев этого блока, ревьюеры не станут проверять работу.

- Пул-реквест не отправлен на проверку.
- Проект не собирается или не запускается.

Работа принимается

В этом блоке собраны требования к итоговой работе, по которым вы можете выявить и самостоятельно исправить частые ошибки:

Общее

- Функциональность, требуемая по заданию предыдущей проектной работы, реализована корректно и продолжает работать. Её описание вы можете найти в чек-листе
- Пул-реквест создан из ветки `sprint-2/step-2` в `main`.
- Сборка и запуск проекта выполняются без ошибок.
- В зависимостях установлены пакеты `redux` , `react-redux` , `redux-thunk`, `react-dnd` , `react-dnd-html5-backend`, а также `@reduxjs/toolkit`, если он используется в проекте.
- В проекте есть:
  - директория `components` с компонентами и стилями: `App`, `AppHeader`, `BurgerIngredients` , `BurgerConstructor`, `Modal`, `ModalOverlay`, `OrderDetails`, `IngredientDetails`;
  - если в проекте используются сторонние шрифты или изображения, то они хранятся в папках `fonts` и `images`;
  - единая директория `services` с вложенными `actions` и `reducers` со всей логикой приложения, либо `services` с `actions` и `reducers` для каждого отдельного компонента;
  - файл `README.md`;
  - файл `.gitignore`.
- Стили портированы как модули. Если есть общие стили, они портированы в глобальную область видимости.
- Код оформлен без ошибок:
  - имена переменных и функций написаны в camelCase;
  - имена классов и функциональных компонент — существительные с прописной буквы;
  - имена переменных — существительные;
  - имя функции отражает то, что она делает.

  Для именования запрещены:
  - транслит;
  - неуместные сокращения.
- Корректно выполняются запросы к API:
  - URL-адрес домена вынесен в отдельную константу;
  - есть проверка успешности выполнения запроса;
  - цепочка обработки промисов завершается блоком `catch`.

React

- Разметка портирована в JSX:
  - разметка заключена в `( )`;
  - разметка вынесена в соответствующие ей компоненты.
- Компоненты написаны корректно:
  - хуки не используются внутри условных блоков;
  - хуки вызываются в основной функции компонента;
  - при использовании классовых компонентов эффекты описаны внутри методов жизненного цикла компонента.
- Для компонентов с пропсами описан `propTypes`. Если в качестве пропсов передается объект или массив объектов, то в `propTypes` описана структура этого объекта.
- Функциональность из брифа реализована корректно:
  - В `BurgerConstructor` может быть только одна булка, которая используется сверху и снизу. Двух разных булок (тип `bun`) быть не может.
  - Стоимость бургера подсчитывается динамически и корректно, в зависимости от тех ингредиентов, которые находятся в конструкторе.
  - При нажатии на кнопку «Оформить заказ» отправляется запрос к API. В теле запроса передается массив из `_id` всех ингредиентов.
  - При успешном запросе отображается модальное окно с компонентом `OrderDetails`, который отбражает данные заказа.
  - При скролле внутри компонента `BurgerIngredients` самый ближний к верхней левой границе заголовок контейнера становится активным.
  - У ингредиентов отображается счётчик с количеством добавленных в бургер ингредиентов этого типа. При добавлении ингредиента в бургер значение счётчика увеличивается, при удалении — значение счётчика уменьшается.
  - Перетаскивание ингредиентов реализовано верно:
    - Пользователь может перенести ингредиент в конструктор.
    - При перетаскивании в конструктор элемента типа `bun`, он заменят текущий элемент булки.
    - Пользователь может сортировать элементы в конструкторе перетаскиванием. Сортировать элементы типа `bun` перетаскиванием нельзя.
    - При перетаскивании элемента вовне конструктора ничего происходить не должно. Элемент удаляется кликом по иконке удаления.
- Работа с Redux выполняется верно:
  - Правильно подключено расширение Redux Devtools, которое работает в режиме разработки.
  - Хранилище имеет начальное состояние из следующих элементов:
    - список всех полученных ингредиентов,
    - список всех ингредиентов в текущем конструкторе бургера,
    - объект текущего просматриваемого ингредиента,
    - объект созданного заказа.
  - Хранилище инициализируется с помощью функции `createStore()` и передается в качестве пропсов в компонент `Provider`.
  - Внедрено хранилище через `Provider`.
  - Для связывания React-компонента с Redux-хранилищем используется `connect` или `useSelector()` и `useDispatch()`.
  - Используются расширители хранилища, корректно применён middleware `thunk` и используется `applyMiddleware` при подключении хранилища.
  - Описаны все экшены и редьюсеры, указанные в задании.

- Для каждого экшена, который связан с запросом к API создан усилитель. Для таких экшенов описан тип `_REQUEST`, тип `_SUCCESS`, `_ERROR`.
- Экшен описывает лишь одно действие. Например, экшена `DECREASE_OR_INCREASE_ITEM` быть не должно.
- Редьюсеры — чистые функции.
- Конструкция `case` внутри редьюсера выполняет только одно действие. Например, удаляет элемент из массива ингредиентов.
- Каждый редьюсер решает только свою задачу: загрузка и хранение ингредиентов, работа конструктора, управление модальным окном ингредиента или оформлением заказа.
- Редьюсеры объединены в один с помощью `combineReducers`.
- Если при выполнении запроса к API в усилителе произошла ошибка, то содержимое хранилища соответствующего элемента приводится к начальному состоянию. Например, если пользователь оформил заказ, а при оформлении следующего у него произошла ошибка — в модальном окне не должен отображаться старый номер заказа. Аналогично при работе со списком ингредиентов.