Programming Evaluation for:

Frontend (VUE, Nuxt)

1. Цель задания

Создать приложение согласно макету, в котором пользователь может:

- просматривать список заявок;
- редактировать список товаров внутри заявки;
- проходить валидацию данных при сохранении;
- отправлять данные на сервер;
- управлять отображением навигационного меню.

Макет:
https://www.figma.com/design/hWvnhnxENItvaW8XmHeUIa/%D0%A2%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B5-%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5?node-id=2-11998&t=XfK5s9TEXfPNqQvi-0

2. Общие требования

1. Приложение построено на **Nuxt 4** (Nuxt 3).
2. Приложение написано на **TypeScript**.
3. Используется **Composition API.**
4. Использовать компонентный подход (Каждый компонент содержит **только свою логику** — без глобальных side-effects).
5. Стили оформляются в **SCSS-модулях**
6. Логика компонентов, асинхронные операции и работа с API реализованы через **кастомные хуки.**
7. Источником данных является **Nuxt Server API** с возможностью замокать данные.
8. Используется **localStorage** для временного хранения данных заявок.
9. UI должен включать:
    - таблицы для отображения заявок и товаров;
    - формы редактирования с валидацией;
    - левое навигационное меню (скрываемое).
10. При отправке данных localStorage очищается.
11. Должны использоваться асинхронные запросы.
12. Приложение не должно использовать какие-либо внешние библиотеки компонентов, стилей, валидации.

3. Структура интерфейса

3.1. Главная страница `/`

Элементы интерфейса:

- Левое **навигационное меню** (отображается только на определённых страницах).
  - Возможность **сворачивания/разворачивания** по клику на кнопку.
- **Таблица заявок**, содержащая следующие колонки:
  - `Номер`
  - `Статус`
  - `Результат проверки`
  - `Дата создания`
  - `Действие` (кнопки управления)
- Для каждой строки отображаются две кнопки:
  - **Редактировать** — переход на страницу `/edit?id=<id>`.
  - **Отправить** — отправка данных на сервер.

Логика отображения кнопок:

- Если в localStorage есть данные по заявке, кнопка должна быть **“Отправить”**.
- Если данных нет — **“Редактировать”**.

3.2. Страница редактирования `/edit`

Элементы интерфейса:

- Заголовок страницы.
- Список товаров (получается с сервера с задержкой 2 с).
- Каждая позиция содержит поля:
  - `Название` (readonly)
  - `Количество` (input, редактируемое поле)
  - `Цена` (input, редактируемое поле)
  - `Цвет` (select с ограниченным набором опций)

Требования к загрузке данных:

- При первом открытии страницы запрос к `/api/products` (задержка 2 секунды).
- После получения данных — сохранение в **localStorage**.
- При повторном открытии страницы данные берутся из **localStorage**, без запроса к серверу.

Валидация при сохранении:

- `Input` — допускаются только **цифры и точка** (`^\d+(\.\d+)?$`).
- `Select` — допускаются только **опции из списка**.
- Проверка всех полей происходит **после нажатия кнопки “Сохранить”**.
- При ошибке валидации соответствующие поля подсвечиваются (например, `border-red`).
- Если все поля валидны:

- данные сохраняются в localStorage;
- выполняется переход обратно на страницу `/`.

3.3. Отправка данных

- При нажатии кнопки **“Отправить”** на главной странице:
  - данные заявки отправляются на сервер (`POST /api/send`).
  - после успешного ответа localStorage очищается.
  - таблица обновляется, статусы сбрасываются.

4. API

4.1. `GET /api/table-data`

Описание: Возвращает список заявок для главной страницы.
Пример ответа:

```
[
  { "id": 1, "name": "Заявка 1", "status": "draft" },
  { "id": 2, "name": "Заявка 2", "status": "draft" }
]
```

4.2. `GET /api/products`

Описание: Возвращает список товаров с задержкой 2 секунды.
Пример ответа:

```
[
  { "id": 1, "name": "Товар 1", "price": 123.45 },
  { "id": 2, "name": "Товар 2", "price": 678.90 }
]
```

4.3. `POST /api/send`

Описание: Получает отредактированные данные заявки и сохраняет (или логирует).
Тело запроса:

```
{
  "id": 1,
  "products": [
    { "id": 1, "name": "Товар 1", "price": 120.00 },
    { "id": 2, "name": "Товар 2", "price": 500.00 }
  ]
}
```

Ответ:

```
{ "success": true }
```

5. Критерии приёмки

1. При запуске отображается таблица заявок с кнопками.
2. При нажатии **Редактировать** → переход на `/edit` → отображаются товары (через 2 сек).
3. После редактирования и успешного сохранения → возврат на `/` → кнопка меняется на **Отправить**.
4. При нажатии **Отправить** → данные отправляются → localStorage очищается → страница обновляется.
5. Меню слева разворачивается и сворачивается по нажатию.
6. Валидация корректно подсвечивает поля при ошибках.
7. Срок выполнения задания 7 дней.