Тестовое задание

SPA-приложение: Orders & Products.

Тестовое задание необходимо выполнить, используя следующий перечень технологий и подходов:

1. Использовать библиотеки для управления глобальным состоянием.
2. Компонентный подход.
3. Использовать роутинг для навигации.
4. Использование анимационных эффектов при переходе между роутами, при переключении между компонентами (transitions, animate.css, etc).
5. Использование ES6 (arrow functions, spread operators, template strings & interpolation, etc).
6. Использование сторонних плагинов не запрещается если без них нельзя обойтись.
7. Git (выполнить тестовое задание с помощью системы контроля версий Git. По итогу окончания работы, прислать ссылку на git-репозиторий с проектом. В репозитории обязательно должен быть понятно написанный Readme.md с описанием установки всех зависимостей и информацией по запуску проекта).
8. Websocket для создания счетчика активных вкладок приложения (см. Описание задачи).
9. HTML/CSS (верстка компонентов должна быть выполнена структурно в рамках скрина-примера).

Необходимо реализовать:

1. Выполнить верстку компонентов на базе скринов-примеров.
2. Приложение должно включать отдельные страницы.
3. Создать компонент Navigation Menu, в котором будут находится роут-ссылки на Orders и Products.
4. Создать компонент TopMenu, в котором необходимо вывести дату и время, которое будет идти в реальном времени (в правом верхнем углу). Рядом с датой вывести счетчик с помощью Socket.io, который будет показывать количество активных сессий приложения в браузерах в реальном времени.
5. Создать компоненты Orders и Products. В каждом Order есть свои products. Пример полей и отношения Orders к Products можно посмотреть в файле app.js.
6. В компоненте Orders сделать функционал, при котором пользователь кликает на конкретный приход, и он открывается рядом. Блок с информацией о приходе можно закрыть (см. скрины). В компоненте Orders вывести информацию по каждому приходу: Название прихода, количество продуктов в приходе. Даты создания прихода в двух форматах, сумма прихода, равная сумме цен продуктов в двух валютах. Кнопка удаления прихода, по клику на которую будет открываться попап (см. скрины).
7. В компоненте Products вывести все продукты + сделать фильтр по типу продуктов (1 селект) В каждом продукте вывести поля: Название продукта, тип продукта, даты гарантии в разных форматах, цена в разных валютах, Название прихода.

Требования к результату (для всех уровней)

Обязательные инструменты для разработчика

- Vue.js (v3);
- VueX;
- CSS Architecture (БЭМ или изоляция);
- CSS Frameworks (Bootstrap);
- Linters and Formatters;
- REST;
- Git;
- Docker
- WebSocket (WS);

Ожидаемый формат вашего тестового в результате

- **Развёрнуть** (Хостинг / VDS), результат будет проверятся тестировщиком по чек-листу, прежде чем будут смотреть тех.специалисты подразделения.
- **Docker**, упаковать в контейнер приложение со всем окружением и зависимостями
- **Git-репозиторий**, для анализа ветвлений в процессе выполнения тестового.
- **Read.me**, так чтобы по нему было понятно что это за проект и какие функции в нём есть.

Самопроверка

Для самопроверки, перед тем как говорить что всё готово, попробуйте запустить свой проект с нуля по вашему **Read.me** из вашего **Git-репозитория**.

Для уровня **Junior+**

Улучшите решение, путем добавления инструментов предыдущего и текущего уровней:

- TypeScript;
- SSR (Nuxt);
- Unit-тесты;
- i18n;
- JWT;
- Web Storage;
- Lazy Loading;
- Charts;
- Maps;

Для уровня **Middle**

Улучшите решение, путем добавления инструментов предыдущего и текущего уровней:

- Graph (на ваш выбор)
  - GraphQL
- Web Workers
- Event-Driven Architecture (EDA)
- Performance Optimization
- Task Runners (npm scripts)
- Интеграционные и функциональные тесты

Для уровня **Middle+**

Улучшите решение, путем архитектурных изменений:

- Micro Frontend Architecture
- Web Assembly