elacompil

Opis zadania

Zadanie polega na stworzeniu fragmentu interfejsu webowego systemu typu smart home. Na ten moment system powinien wspierać trzy typy inteligentnych urządzeń: żarówkę, gniazdko elektryczne i czujnik temperatury. W przyszłości może zostać rozbudowany o wsparcie dla kolejnych typów.

Główny ekran powinien zawierać listę urządzeń podpiętych do systemu. Forma prezentacji listy jest dowolna, ale dla każdego urządzenia powinny być widoczne podstawowe informacje jak: typ, nazwa i stan połączenia.

Po kliknięciu na element z listy powinno otworzyć się okno wizualizujące stan urządzenia. Zawartość zależy od typu urządzenia, które zostało wybrane. Poza tym okno:

1. Powinno wspierać opcję dragging. W tym celu zalecana jest biblioteka interact.js.
2. Nie powinno blokować wyboru innego urządzenia z listy.
3. Jeżeli jest już otwarte w momencie wybrania kolejnego urządzenia, powinno pozostać otwarte, a jego zawartość podmieniona.
4. Po ponownym otwarciu powinno się pojawiać w tym samym miejscu, w którym było ostatni raz wyświetlone.

Stan urządzeń powinien być aktualizowany (symulowany) na bieżąco. W tym celu można periodycznie odpytywać REST-owe endpointy w API, ale dodatkowe punkty można otrzymać za aktualizację przy użyciu protokołu WebSocket. Aktualizacje powinny być odwzorowywane w interfejsie użytkownika (zarówno na liście, jak również w oknie ze szczegółami).

Backend API

Należy założyć, że mamy do czyniania z sytuacją, gdzie zespół backendowy nie jest gotowy udostępnić API, ale została ustalona jego specyfikacja (poniżej). Żeby nie czekać, frontend developer powinien zamockować je w dowolny sposób, tak aby móc przygotować i przetestować swoją implementację frontendu.

```
GET /api/v1/devices             => zwraca SmartDevice[];
GET /api/v1/devices/{deviceId}  => zwraca SmartDeviceDetails;
GET /api/v1/refresh             - endpoint typu WebSocket;
```

Format odpowiedzi to JSON

Jako dane w message po WebSocket przychodzi `SmartDeviceDetails` i zawiera informacja o aktualnym stanie urządzenia

elacompil

Format danych jest następujący:

```
SmartDevice:
{
  type: string; // 'bulb', 'outlet' or 'temperatureSensor';
  id: string;
  name: string;
  connectionState: string; // 'connected', 'disconnected' or 'poorConnection'
}

// SmartDeviceDetails can be SmartBulb, SmartOutlet or SmartTemperatureSensor

SmartBulb:
{
  type: 'bulb';
  id: string;
  name: string;
  connectionState: string; // 'connected', 'disconnected' or 'poorConnection'
  isTurnedOn: boolean;
  brightness: number; // <0, 100>
  color: string; // in the CSS formats
}

SmartOutlet:
{
  type: 'outlet';
  id: string;
  name: string;
  connectionState: string; // 'connected', 'disconnected' or 'poorConnection'
  isTurnedOn: boolean;
  powerConsumption: number; // in watts
}

SmartTemperatureSensor:
{
  type: 'temperatureSensor';
  id: string;
  name: string;
  connectionState: string; // 'connected', 'disconnected' or 'poorConnection'
  temperature: number; // in Celsius
}
```

elacompil

Jak dostarczyć zadanie

Prześlij nam link do repozytorium na GitHub gdzie umieściłeś rozwiązanie zadania i link do demo gdzie możemy je przetestować. Zamiast live demo możesz też umieścić w repo informację jak należy je zbudować i przetestować. Jeżeli nie uda Ci się zrobić całego zadania, podeślij nam przynajmniej jego część i daj znać czego nie udało się zrobić i dlaczego. Kompletność rozwiązania jest elementem oceny, ale akceptujemy również rozwiązania niekompletne, które pokazują, że wiesz o co chodzi i widać, że umiałbyś je dokończyć.

Elementy podlegające ocenie

- kompletność wykonania zadania,
- jakość kodu,
- realizacja i sposób realizacji wymagań,
- estetyka zaproponowanego interfejsu użytkownika,
- adekwatność stosowania bibliotek/wzorców do rozwiązywanego problemu

Dodatkowe plusy

- za ciekawy sposób wizualizacji listy urządzeń i szczegółów urządzenia,
- za dodanie opcji resizing dla okna dialogowego,
- za Responsive Web Design dla zawartości okna dialogowego,
- za Responsive Web Design dla listy urządzeń,
- za użycie TypeScript i dobre modelowanie danych.

FAQ

1. **Z jakiego frameworka mam skorzystać?** Nie narzucamy niczego, możesz wybrać to, co Twoim zdaniem sprawdzi się najlepiej. My aktualnie w projektach korzystamy z Vue.js, ale to tylko dlatego, że w naszych zastosowaniach rachunek zysków i strat jest najbardziej korzystny. Możesz też nie korzystać w ogóle z frameworka, jeżeli chcesz nam pokazać, że rozumiesz jak działa JS i umiałbyś takie podstawowe rzeczy zaimplementować sam. Jeżeli wybierzesz np. Reacta, to możesz się spodziewać na rozmowie pytań z nim związanych w kontekście rozwiązań które zastosowałeś w aplikacji.
2. **Jaki jest deadline?** Nie ma. Na dobre rozwiązania i dyskusje na ich temat będziemy otwarci nawet po zakończeniu rekrutacji.

elacompil

Częste błędy

1. Brak możliwości przetestowania reaktywności komponentów (monitorowane urządzenia nigdy nie zmieniają stanu). Idealnie gdyby zmieniał się ‘stan połączenia’ urządzeń, ponieważ wtedy możemy sprawdzić synchronizację stanu pomiędzy listą i oknem ze szczegółami.
2. Niezgodność API ze specyfikacją podaną w zadaniu (nie można swobodnie zmieniać formatu odpowiedzi).