Để đáp ứng yêu cầu của bạn, bạn có thể sử dụng `rxjs` để theo dõi sự thay đổi dữ liệu từ API và cập nhật biểu đồ một cách realtime.

Đầu tiên, bạn cần cài đặt thư viện `rxjs` bằng cách chạy lệnh sau:

```

npm install rxjs

```

Sau đó, bạn có thể sử dụng `Observable` để theo dõi sự thay đổi dữ liệu từ API và cập nhật biểu đồ tương ứng. Bạn có thể thực hiện như sau:

```

import { Component, Injector, OnInit } from '@angular/core';

import { BaseComponent } from '../../../../../../core/base.component';

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

import \* as echarts from 'echarts';

import { VariablesImportanceService } from './variables-importance.service';

import { FormControl, FormGroupDirective, NgForm, Validators } from '@angular/forms';

import { ErrorStateMatcher } from '@angular/material/core';

import { ActivatedRoute } from '@angular/router';

export class MyErrorStateMatcher implements ErrorStateMatcher {

isErrorState(control: FormControl | null, form: FormGroupDirective | NgForm | null): boolean {

const isSubmitted = form && form.submitted;

return !!(control && control.invalid && (control.dirty || control.touched || isSubmitted));

}

}

@Component({

selector: 'app-variables-importance',

templateUrl: './variables-importance.component.html',

styleUrls: ['./variables-importance.component.scss'],

})

export class VariablesImportanceComponent extends BaseComponent implements OnInit {

chart: echarts.ECharts;

options: echarts.EChartsOption;

public variableImportantChart: any;

items: number[] = [];

keys: string[] = [];

columns = new FormControl('', [Validators.required, Validators.nullValidator]);

data: any[] = [];

sessionId: string;

matcher = new MyErrorStateMatcher();

constructor(

private variablesImportanceService: VariablesImportanceService,

injector: Injector,

private route: ActivatedRoute

) {

super(injector);

}

getShowChart() {

this.variablesImportanceService.getDataPredict('variable\_importance', this.sessionId).subscribe((data) => {

this.data = Object.entries(data?.body)

.sort((a: any, b: any) => a[1] - b[1])

.reverse()

.reduce((acc, [key, value]) => {

acc[key] = value;

return acc;

}, []);

for (let item in this.data) {

this.items = [...this.items, data.body[item]];

}

this.keys = Object.keys(this.data);

this.variableImportantChart = {

title: {

text: '',

},

tooltip: {},

yAxis: {

type: 'category',

data: this.keys.slice(0, 3),

axisLabel: {

textStyle: {

fontSize: 14,

fontWeight: 'bold',

},

},

},

xAxis: {

type: 'value',

axisLabel: {

formatter: '{value}%',

},

},

grid: {

left: '8%',

containLabel: true,

},

series: [

{

name: 'Variable importance',

type: 'bar',

data: this.items.slice(0, 3),

label: {

show: true,

position: 'right',

formatter: function (param: any) {

return param.value.toFixed(2) + '%';

},

},

itemStyle: {

emphasis: {

color: '#7acb7e',

},

normal: {

color: (params: any) => {

const opacity = 1 - params.dataIndex / 5;

return 'rgba(69,94,221,' + opacity + ')';

},

},

},

},

],

};

let chart = echarts.init(document.getElementById('variableImportant'));

chart.setOption(this.variableImportantChart);

// Create an Observable to track changes in data

const dataChanges$: Observable<any> = this.variablesImportanceService.getDataChanges('variable\_importance', this.sessionId);

// Subscribe to the Observable and update the chart when data changes

dataChanges$.pipe(

map((data) => {

this.data = Object.entries(data?.body)

.sort((a: any, b: any) => a[1] - b[1])

.reverse()

.reduce((acc, [key, value]) => {

acc[key] = value;

return acc }, []);

for (let item in this.data) {

this.items = [...this.items, data.body[item]];

}

this.keys = Object.keys(this.data);

this.variableImportantChart.yAxis.data = this.keys.slice(0, 3);

this.variableImportantChart.series[0].data = this.items.slice(0, 3);

chart.setOption(this.variableImportantChart);

})

).subscribe();

});

}

onKeyUp(event: any) {

const selectedColumns = event.target.value;

const arr = Object.entries(this.data).slice(0, selectedColumns);

const keyItem = [];

const valueItem = [];

for (let item in arr) {

valueItem.push(arr[item].slice(0, 1));

keyItem.push(arr[item].slice(1, 2));

}

this.variableImportantChart.yAxis.data = valueItem.flat();

this.variableImportantChart.series[0].data = keyItem.flat();

let chart = echarts.init(document.getElementById('variableImportant'));

chart.setOption(this.variableImportantChart);

}

onSelectionChange(event: any) {

const selectedColumns = event.source.value;

const result = selectedColumns.map((key: any) => {

return this.data[key];

});

this.variableImportantChart.yAxis.data = selectedColumns;

this.variableImportantChart.series[0].data = result;

let chart = echarts.init(document.getElementById('variableImportant'));

chart.setOption(this.variableImportantChart);

}

ngOnInit(): void {

// this.sessionId = this.route.snapshot.paramMap.get('id');

this.sessionId = '3';

this.getShowChart();

}

}

```

Ở đây, chúng ta tạo một `Observable` với phương thức `getDataChanges()` của `variablesImportanceService`, theo dõi sự thay đổi của dữ liệu từ API. Mỗi khi có thay đổi, chúng ta sẽ cập nhật dữ liệu và biểu đồ tương ứng bằng cách sử dụng phương thức `setOption()` của `echarts`.

Lưu ý rằng bạn cần phải sử dụng phương thức `unsubscribe()` để ngừng theo dõi sự thay đổi dữ liệu khi component bị hủy.

Mong rằng điều này sẽ giúp bạn cải thiện hiệu suất và trải nghiệm của ứng dụng của mình.