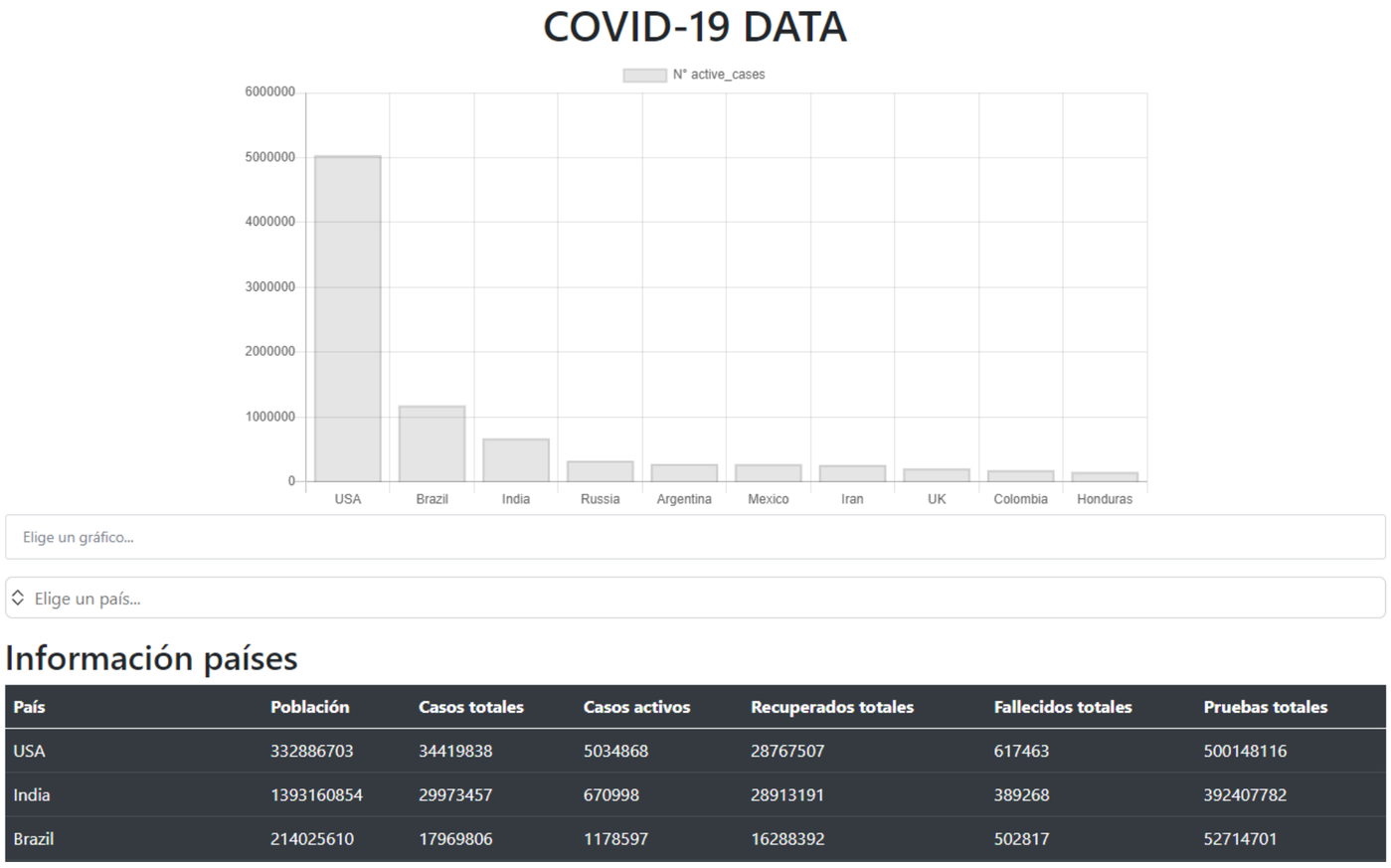
SVELTE

GRUPO 7 - DAHL

Objetivo

Construir una aplicación en base al framework JavaScript Svelte

En nuestro caso, generamos una aplicación de tracking de información relacionada a COVID-19



País	Población	Casos totales	Casos activos	Recuperados totales	Fallecidos totales	Pruebas totales
USA	332886703	34419838	5034868	28767507	617463	500148116
India	1393160854	29973457	670998	28913191	389268	392407782
Brazil	214025610	17969806	1178597	16288392	502817	52714701

DEMO

API

```
import axios from "axios";

async function getAll() {
    const options = {
        method: 'GET',
        url: 'https://worldometers.p.rapidapi.com/api/coronavirus/all/',
        headers: {
          'x-rapidapi-key': '531a08cec7msh94a6d9b4e4a844ep15f8fbjsnb6421a2820ce',
          'x-rapidapi-host': 'worldometers.p.rapidapi.com'
        }
      };

    const response = await axios.request(options);
    return response;
}

const API = {getAll};

export default API;      Fernanda, 2 days ago • api call
```

```
import { writable } from "svelte/store";

function createStore() {
  const { subscribe, set, update } = writable({
    graphCountries: [],
    countriesNames: [],
    data: [],
    selectedChart: "active_cases",
    fetching: null,
    items: [
      {value: 'active_cases', label: 'Casos Activos'},
      {value: 'total_cases', label: 'Casos Totales'},
      {value: 'population', label: 'Población'},
      {value: 'total_recovered', label: 'Total de Recuperados'},
      {value: 'total_deaths', label: 'Total de Fallecidos'},
      {value: 'total_tests', label: 'Total de Testeados'}
    ],
  });
  return {
    subscribe,
    addData: (data) => {
      update((state) => {
        state.data = data;
        return state;
      }
    )},
    addNames: (countriesNames) => {
      update((state) => {
        state.countriesNames = countriesNames;
        return state;
      }
    )},
    updateGraphCountries: (newCountry) => {
      console.log("updateGraphCountries");
      update((state) => {
        state.graphCountries = newCountry;
        return state;
      });
    },
    changeChart: (selectedChart) => {
      console.log("changeChart");
      update((state) => {
        state.selectedChart = selectedChart;
        return state;
      });
    },
    updateFetching: (fetching) => {
      console.log("updateFetching");
      update((state) => {
        state.fetching = fetching;
        return state;
      });
    },
  };
}
```

Store

- Simplificada.
- Sintaxis simple.
- Acceso global elementos del store.
- Rápida implementación a través del método *writable*.

App components

```
<div id="main">
  <div class="container">
    <div class="row mt-5">
      <div class="col">
        <h1 class="text-center">COVID-19 DATA</h1>
      </div>
    </div>
    <div style="width:100%">
      <Chart {...commonProps}/>
    </div>
    <div style="width:100%">
      <MultiSelect/>
      <Table {...commonProps}/>
    </div>
  </div>
</div>

<style>

  #main {
    display: block;
    align-items: center;
  }


  .container {
    margin: 0 auto;
    border: 5px;
  }
</style>
```

MultiSelect

```
{#if $store.data.length > 0}
  <Select items={$store.items} on:select={handleSelect}></Select>
  <MultiSelect bind:selected {name} {placeholder} options={$store.countriesNames} {required} />
{:else}
  <div class="container-spinner">
    <div style="margin: 0 auto;">
      <Circle size="60" color="#FF3E00" unit="px" duration="1s"></Circle>
    </div>
  </div>
{/if}
```

Table

```
{#if $store.data.length > 0}
  <SvelteTable
    columns="{columns}"
    rows="{selection}"
    classNameTable={['table table-dark']}
    classNameThead={['thead-light']}
    classNameSelect={['custom-select']}>
  </SvelteTable>

{:else}
  <div class="container-spinner">
    <div style="margin: 0 auto;">
      <Circle size="60" color="#FF3E00" unit="px" duration="1s"></Circle>
    </div>
  </div>
{/if}
```

Chart

```
{#if $store.data.length > 0}
  <MDBRow>
    <MDBCol md="8" class="mx-auto">
    <Bar {data} {options} />
    </MDBCol>
  </MDBRow>
{:else}
  <div class="container-spinner">
    <div style="margin: 0 auto;">
      <Circle size="60" color="#FF3E00" unit="px" duration="1s"></Circle>
    </div>
  </div>
{/if}

<style>
  beayancan, 8 minutes ago | 1 author (beayancan)
  .container-spinner {
    display: flex;
  }
</style>
```

Reactividad

```
$: {
  if ( $store.graphCountries.length > 0) {
    selection = _.orderBy($store.data, [$store.selectedChart], ["desc"]);

    let filteredData = $store.data.filter(
      (x) => $store.graphCountries.includes(x.name)
    );
    selection.push.apply(selection, filteredData);
    selection = _.uniqBy(selection, "name");
    selection = _.orderBy(selection, [$store.selectedChart], ["desc"]);
  }
  else {
    selection = _.orderBy($store.data, [$store.selectedChart], ["desc"]);
  }
}
```

Conclusiones

- Framework sencillo y amigable para desarrollar
- Store más compacta y simple.
- Se escribe menos código.
- Menos boilerplate.
- Al tener componentes reactivos Svelte actualiza solamente el nodo afectado cada vez que el estado de un app cambia.
- No necesita virtual DOM.
- No es muy popular, por lo tanto, no existen tantos componentes a diferencia de sus competidores como Vue o React.

MUCHAS GRACIAS

GRUPO 7 - DAHL