PROYECTO DE SOFTWARE

Cursada 2020

TEMARIO

- Microservicios
- Frameworks JS
- Intro Vue

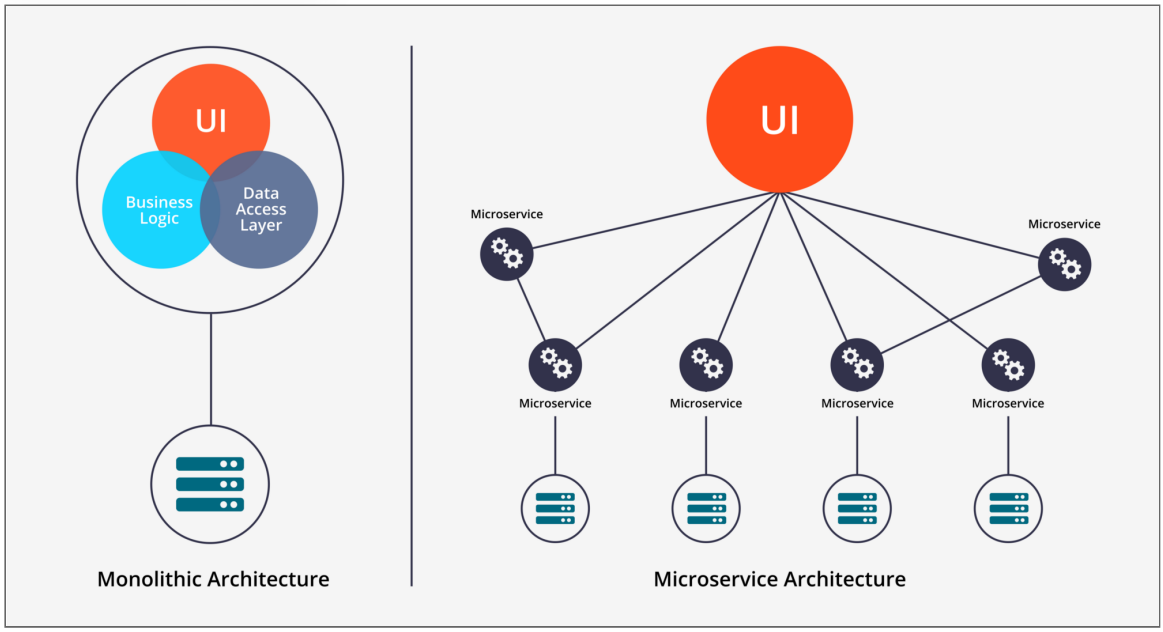
ARRANQUEMOS CON LAS PREGUNTAS DE REPASO

ARQUITECTURA DE MICROSERVICIOS

MICROSERVICIOS

- Las aplicaciones se dividen en sus componentes más pequeños, independientes entre sí.
- A diferencia del enfoque tradicional y monolítico de las aplicaciones, en el que todo se encuentra en una única pieza.
- Los microservicios funcionan en conjunto para llevar a cabo las mismas tareas que la aplicación monolítica.
- Los microservicios facilitan la escalabilidad de todo el sistema, se despliegan según se vayan necesitando.
- Pueden tener distintas tecnologías entre sí.
- Al ser más pequeños, son mas simples de mantener y actualizar.

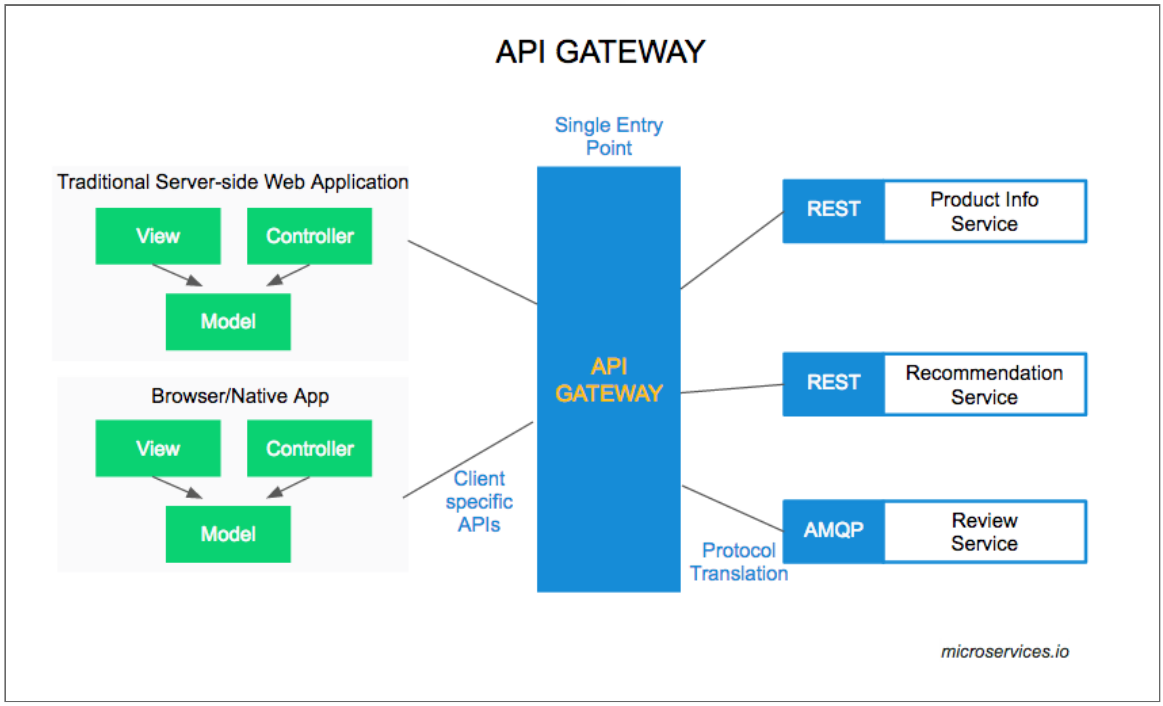
ARQUITECTURA MONOLÍTICA VS ARQUITECTURA DE MICROSERVICIOS



¿CÓMO COMUNICO UN SERVICIO CON OTRO?

- Actualmente la opción más utilizada es mediante APIs HTTP/REST con JSON.
- Incluso puede centralizarse la comunicación utilizando un API Gateway.
- En dicho API Gateway puede implementarse una capa de seguridad, que ante una petisión verifique si el cliente tiene permisos de acceso.

MICROSERVICIOS: API GATEWAY



FRAMEWORKS JS

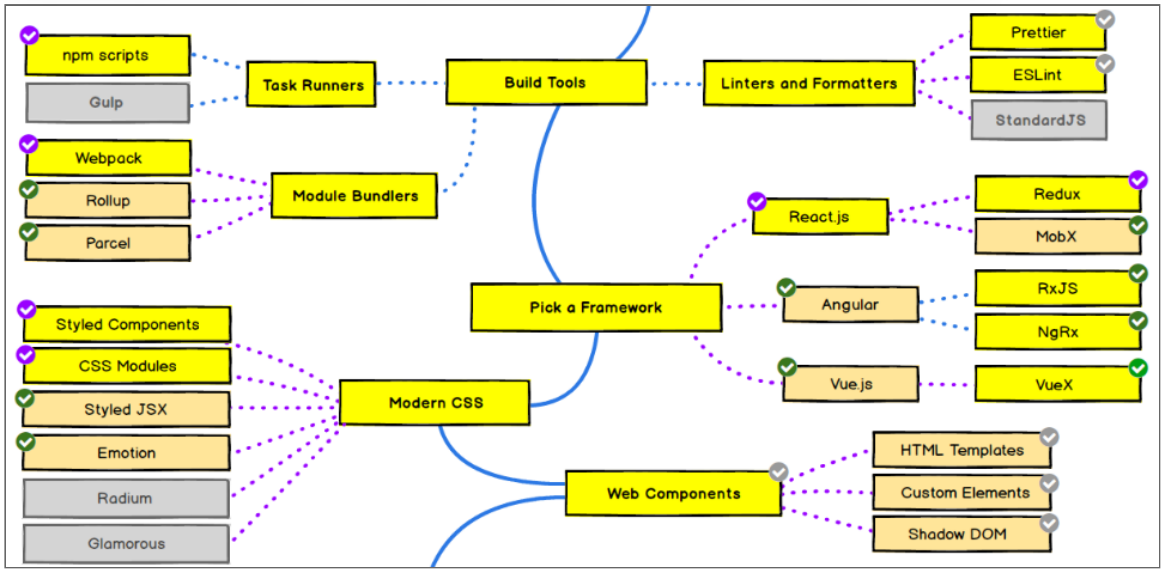
¿QUÉ ÉS UN FRAMEWORK JS?

- Trabajar con **JS pelado** es complejo. Por eso nació **jQuery**(2006) para facilitar el desarrollo.
- jQuery sigue estando **muy extendido**.
- El poder de cómputo de los clientes web **aumenta día a día**.
- Las aplicaciones web implementan **cada vez más funcionalidades y complejidad**, con lo que jQuery se queda corto.

DEMASIADAS HERRAMIENTAS Y FRAMEWORKS JS:

https://en.wikipedia.org/wiki/List_of_JavaScript_libraries

ELECCIÓN DE UN FRAMEWORK JS: FRONT-END PATH



Developer Roadmap

¿POR QUÉ EXISTEN LOS FRAMEWORK JS? LA VERDADERA RAZÓN:

VUE.JS

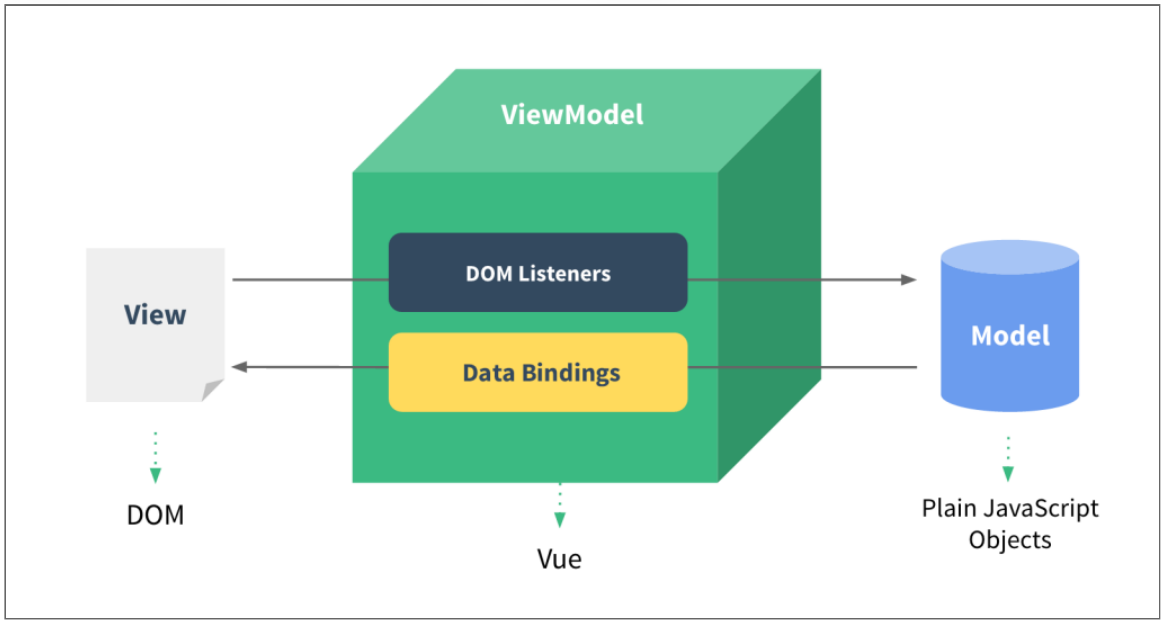
CARACTERÍSTICAS

- Vue.js es un framework de JavaScript para front-end.
- Facilidad de aprendizaje y uso con respecto a otros frameworks como **ReactJS**.
- Mejor rendimiento comparado con **AngularJS**.
- Vue.js es un framework **progresivo**. Tiene la facilidad para usarlo y adaptarlo a proyectos tanto grandes como pequeños.
- Ha tenido un grán crecimiento. Veamos su comunidad en **Github**.

CARACTERÍSTICAS

- Es un framework "reactivo" que implementa **"two way data-binding"**: enlace de datos en dos direcciones (entre la vista y el modelo) de una manera muy eficiente y rápida.
- Se basa en el patrón **Model-View-Viewmodel**
- Vue.js, está más enfocado hacia la vista, y puede ser implementado en el HTML de cualquier proyecto web sin requerir cambios drásticos.
- Los navegadores modernos poseen la extensión **Vue.js devtools** que nos asiste en el desarrollo.
- Vue.js soporta todos los browsers que sean compatibles con **ES5-compliant**. (Ver **Versiones JS**.)

MODEL-VIEW-VIEWMODEL EN VUEJS



INSTALACIÓN VUEJS:

- Descargando el js e incluyéndolo directamente en un tag **<script>**.
- Linkear directamente desde una **CDN** (Content Delivery Network).

```
<script src="https://unpkg.com/vue">
</script>
```

- Intalar Vue CLI via **npm** (manejador de paquetes por defecto para Node.js):

```
$ npm install vue
```

o

```
npm install -g @vue/cli
# OR
yarn global add @vue/cli
```

Este último es la opción recomendada para para proyectos más grandes.

INCLUYENDOLO DE UNA CDN:

Veamos **hello_world**.

```
  <title>Mi primera aplicación Vue</title>
<script src="https://unpkg.com/vue"></script>

  <div id="app">
    <h2>{{ message }}</h2>
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        message: 'Hola Vue!!!'
      }
    })
  </script>
```

Los datos y DOM están ahora relacionados utilizando **{{}}**. Modifiquemos **app.message**.

DIRECTIVAS VUE

- Son atributos específicos de Vue que comienzan con *v-*.
    - **v-text**, **v-once**, **v-html**.
    - **v-bind**, **v-model**.
    - Condicionales: **v-if**, **v-else**, **v-else-if**.
    - Bucles: **v-for**.
    - Eventos: **v-on**.
    - **v-show**.

DIRECTIVA V-BIND:

- La **intepolación {{}}** no funciona para atributos, se utiliza **v-bind** para relacionar atributos con datos de Vue.
- Veamos **v-bind**.

```
  <title>Directivas Vue</title>
<script src="https://unpkg.com/vue"></script>

  <div id="app">
    <span v-bind:title="message">
      Hover your mouse over me !!
    </span>
  </div>
  <script>
  var app = new Vue({
    el: '#app',
    data: {
      message: 'Fecha ' + new
Date().toLocaleString()
    }
  })
  </script>
```

DIRECTIVA CONDICIONAL V-IF:

- Veamos **v-if**.

```
  <title>Directivas Vue</title>
  <script src="https://unpkg.com/vue">
</script>

  <div id="app">
    <span v-if="seen">Now you see me</span>
    <span v-else="">Oh no 😢</span>
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        seen: true
      }
    })
  </script>
```

BUCLES V-FOR:

- Veamos **v-for**.

```
  <title>Lista de compras</title>
<script src="https://unpkg.com/vue"></script>

  <div id="app">
    <ul>
      <li v-for="product in products">
        {{ product }}
      </li>
    </ul>
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        products: [
          'Harina',
          'Arroz',
          'Yerba'
        ]
      }
    })
  </script>
```

- Modifiquemos **app.products**, agregando: **app.products.push("Manteca")** y eliminando: **app.products.pop()**.

MÉTODOS Y EVENTOS V-ON:

- La directiva **v-on** nos permite actuar cuando se produzca algún **evento DOM**.
- Dentro de la sección **methods** ponemos el método que se va a disparar cuando el evento se produzca.
- Veamos **v-on**.

```
  <title>Directivas Vue</title>
  <script src="https://unpkg.com/vue">
</script>

<div id="app-5">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse
Message</button>
</div>
<script>
var app = new Vue({
  el: '#app-5',
  data: {
    message: 'Bienvenidos a Proyecto de
Desarrollo!!'
  },
  methods: {
    reverseMessage: function () {
      this.message =
this.message.split('').reverse().join('')
    }
  }
})
</script>
```

EVENTOS V-ON:

- Incluso es posible "colgarse" de múltiples eventos:

```
<div v-on="
  click   : onClick,
  keyup   : onKeyup,
  keydown : onKeydown
">
</div>
```

- Notar que se modifica el estado de nuestra applicación sin tocar el DOM, todo eso lo hace Vue.
- El código queda simplificado y enfocado en la lógica de lo que hay que resolver.

DIRECTIVA V-MODEL:

- Hace la relación bidireccional entre un input y los datos de la aplicación Vue.
- Veamos **v-model** y **v-models**.

```
  <title>Directivas Vue</title>
  <script src="https://unpkg.com/vue">
</script>


<div id="app-6">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
<script>
var app = new Vue({
  el: '#app-6',
  data: {
    message: 'Hello Vue!'
  }
})
</script>
```

PROPIEDADES COMPUTADAS:

- Veamos **propiedades-computadas**.
- Nos evita poner demasiada lógica en la visualización.

```
  <title>Lista de compras</title>
  <meta charset="UTF-8">
  <script src="https://unpkg.com/vue">
</script>

  <div id="app">
    <ul>
      <li v-for="product in products">
        {{ product }}
      </li>
    </ul>
    Cantidad de elementos: {{ countProducts }}
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        products: [
          'Harina',
          'Arroz',
          'Yerba'
        ]
      },
      computed: {
        // a computed getter
        countProducts: function () {
          return this.products.length
        }
      }
    })
  </script>
```

WATCHERS:

- Veamos **watcher**.
- Para reaccionar cuando un dato cambia.

```
  <title>Lista de compras</title>
  <meta charset="UTF-8">
  <script src="https://unpkg.com/vue">
</script>

  <div id="app">
    <ul>
      <li v-for="product in products">
        {{ product }}
      </li>
    </ul>
    Cantidad de elementos: {{ countProducts }}
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        products: [
          'Harina',
          'Arroz',
          'Yerba'
        ],
        countProducts: 3
      },
      watch: {
        products: function () {

this.countProducts=this.products.length
        }
      }
    })
  </script>
```

CONSUMIENDO UNA API CON VUEJS

EJEMPLO BÁSICO CONSULTANDO UNA API CON FETCH

- Veamos **lista-api**
- En este caso utilizamos el hook **created** dentro del ciclo de vida de la instancia Vue.

```
  <div id="app">
    <ul>
      <li v-for="municipio in municipios">
        {{ municipio.id }} - {{ municipio.name
}} (Fase: {{ municipio.phase }})
      </li>
    </ul>
  </div>
  <script>
    var app = new Vue({
      el: '#app',
      data: {
        municipios: []
      },
      created() {
        fetch('https://api-
referencias.proyecto2020.linti.unlp.edu.ar/mun
icipios')
          .then(response => response.json())
          .then(json =>{
            this.municipios = json.data.Town
          })
      }
    })
  </script>
```

UTILIZANDO EL CLIENTE HTTP AXIOS

- Realiza los **XMLHttpRequests** del cliente (browser).
- Realiza las peticiones HTTP en node.js (servidor).
- Soporta API de **Promesas de JS**.
- Intercepta y transforma los datos de requerimientos y respuestas.
- Transforma automáticamente a JSON.
- Soporte del lado del cliente para protección contra **CSRF**.

EJEMPLO BÁSICO CONSULTANDO UNA API CON AXIOS

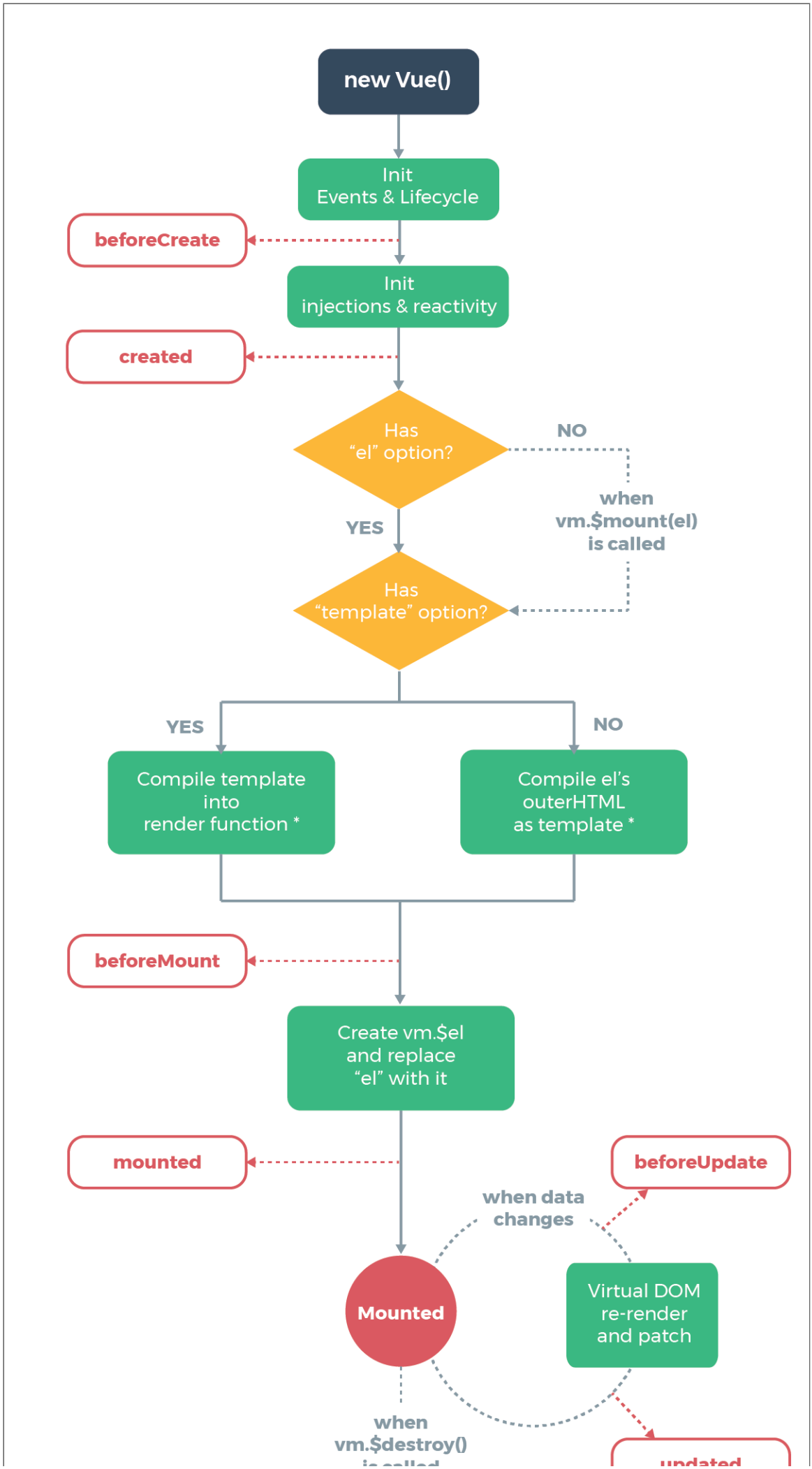
- Veamos **lista-api-axios**.
- Ahora, abramos directamente el archivo local en el navegador.

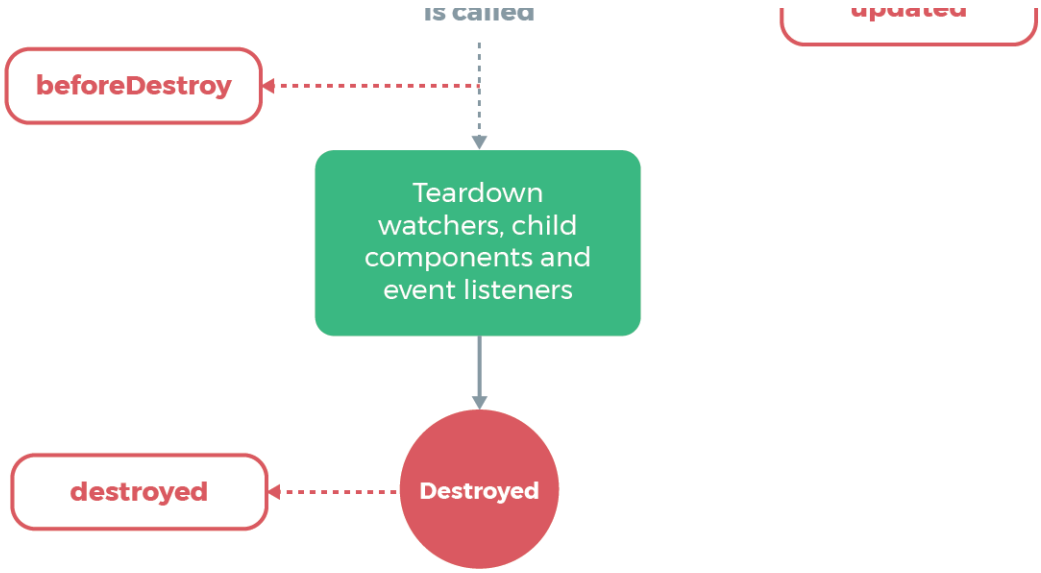
```
  <title>Provincias Argentinas</title>
  <meta charset="UTF-8">
  <script src="https://unpkg.com/vue">
</script>
  <script
src="https://unpkg.com/axios/dist/axios.min.js
"></script>


  <div id="app">
    <div v-if="provincias &&
provincias.length">
      <h1>Provincias Argentinas:</h1>
      <ul>
        <li v-for="p in provincias">
          {{ p.id }} - {{ p.nombre }}
        </li>
      </ul>
    </div>
    <div v-if="errors && errors.length">
      <h1>Errores:</h1>
      <ul>
        <li v-for="error of errors">
          {{error.message}}
        </li>
      </ul>
    </div>
  Fuente: <a href="https://datos.gob.ar/"
target="blank">https://datos.gob.ar/</a>
  </div>
  <script>
    var app = new Vue({
      el: '#app',
      data: {
        provincias: [],
        errors: []
      },
      created() {
```

```
axios.get('https://apis.datos.gob.ar/georef/ap
i/provincias')
              .then(response => {
              // JSON responses are
automatically parsed.
              this.provincias =
response.data.provincias
              })
              .catch(e => {
                this.errors.push(e)
              })
      }
      })
  </script>
```

CICLO DE VIDA DE UNA INSTANCIA VUE:

Diagrama


new Vue()
Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO
when
vm.$mount(el)
is called
YES
Has
“template” option?
YES
NO
Compile template
into
render function *
Compile el’s
outerHTML
as template *
beforeMount
Create vm.$el
and replace
“el” with it
mounted
beforeUpdate
when data
changes
Mounted
Virtual DOM
re-render
and patch
when
vm.$destroy()
updated



* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

GUIA VUE

- https://vuejs.org/v2/guide

TUTORIALES VUE:

- Tutorial paso a paso: **https://coursetro.com/courses/23/Vue-Tutorial-in-2018---Learn-Vue.js-by-Example** **<< Tarea para el hogar**
- Vue.js fundamentals: **https://www.youtube.com/playlist?list=PLwAKR305CRO_1yAao-8aZiQnBqJeyng4O**

PARA SEGUIR LEYENDO: JS FRAMEWORKS

- **https://carlosazaustre.es/frameworks-de-javascript/**
- Comparacíon de Frameworks JS: **https://tinyurl.com/comparacionLibsJS**
- **https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/**
- **https://medium.com/dailyjs/the-deepest-reason-why-modern-javascript-frameworks-exist-933b86ebc445**
- **https://javascriptreport.com/the-ultimate-guide-to-javascript-frameworks/**

¿DUDAS?

SEGUIMOS CON VUEJS LA PRÓXIMA SEMANA...