UNIVERSIDAD
NACIONAL
DE COLOMBIA

Ciclo III

Desarrollo de Software

Capa de Presentación: Componentes Vue

20

Jeisson Andrés Vergara Vargas
Departamento de Ingeniería de Sistemas e Industrial
http://colswe.unal.edu.co/~javergarav/
javergarav@unal.edu.co

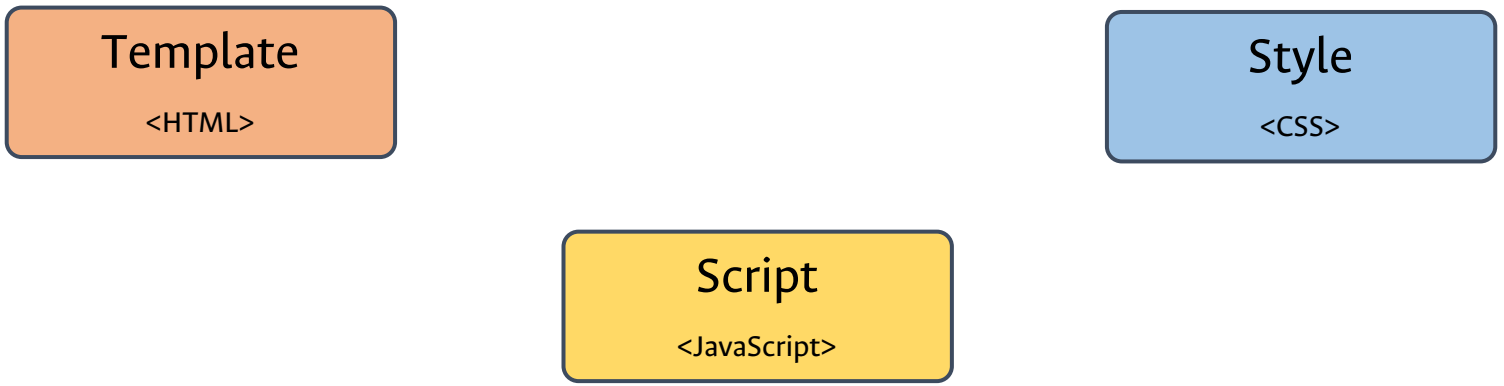
Objetivo de Aprendizaje

Comprender en detalle la estructura de los componentes en Vue.

Conceptos Básicos

Partes de un Componente

Como se estudió anteriormente, un componente en Vue está compuesto por 3 partes.



Las 3 partes son definidas en un archivo **.vue**

Comunicación entre el Script y el Template

La comunicación entre el Script (JavaScript) y el Template (HTML) permite una interacción en tiempo real, algunos ejemplos son:

- Mostrar o no partes del template, dado el valor de una variable.
- Procesar las entradas que el usuario realiza, por ejemplo: escribir un texto o dar un clic.
- Usar contenido dinámico en las etiquetas HTML, es decir que el valor dependa de alguna variable.

Comunicación entre el Script y el Template

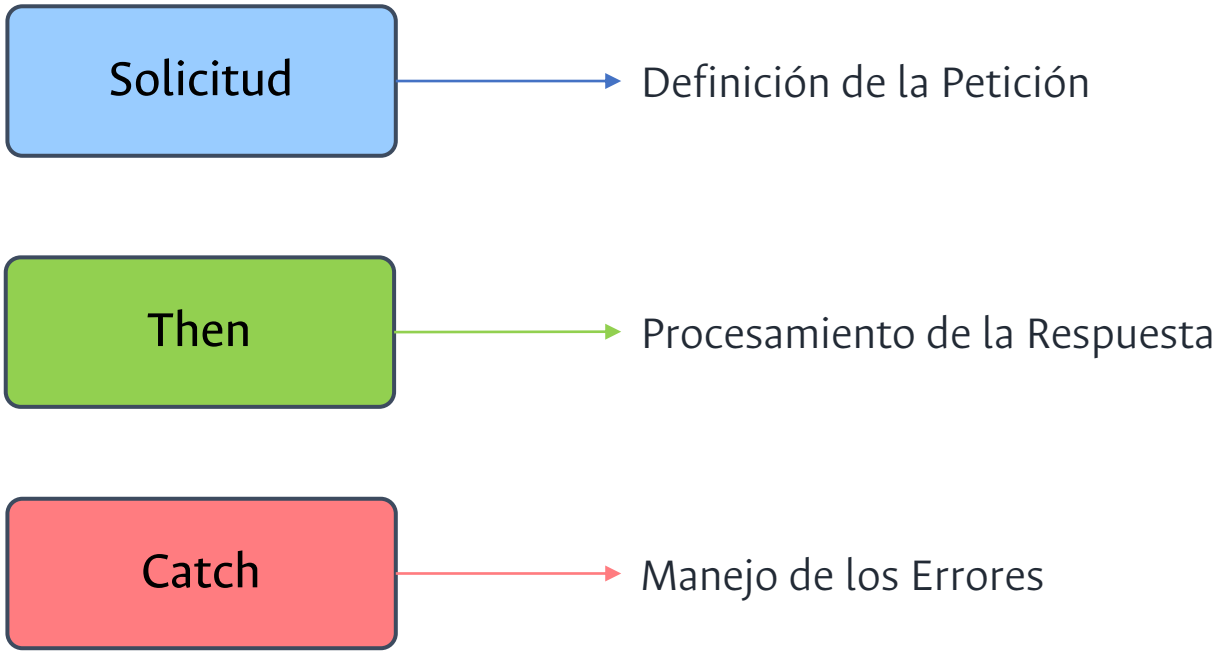
Algunas de las herramientas que ofrece Vue para hacer posible la comunicación entre el Template y el Script son:

- {{variable}}
- v-on
- v-if
- v-for
- v-model

Su uso será mostrado en el código fuente.

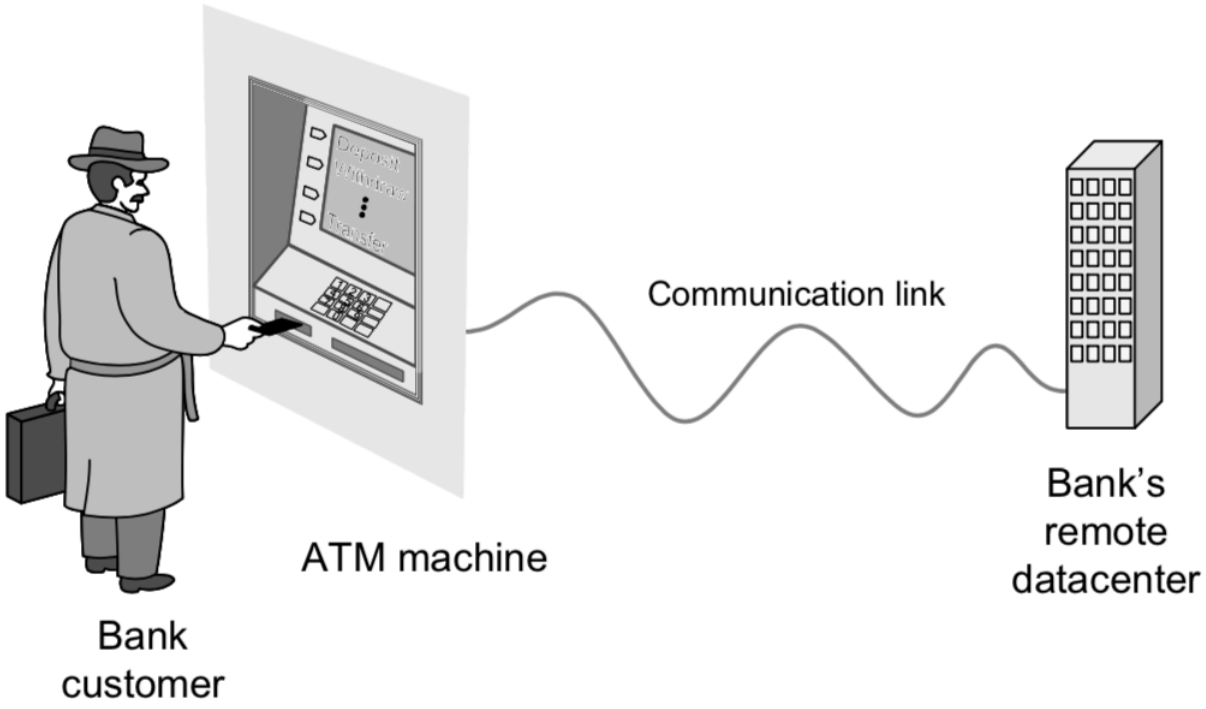
Peticiones HTTP

Con ayuda de AXIOS se realizan las peticiones al BackEnd, dichas peticiones tienen una determinada estructura.



Ejemplo

Software para un «ATM»



Ejemplo

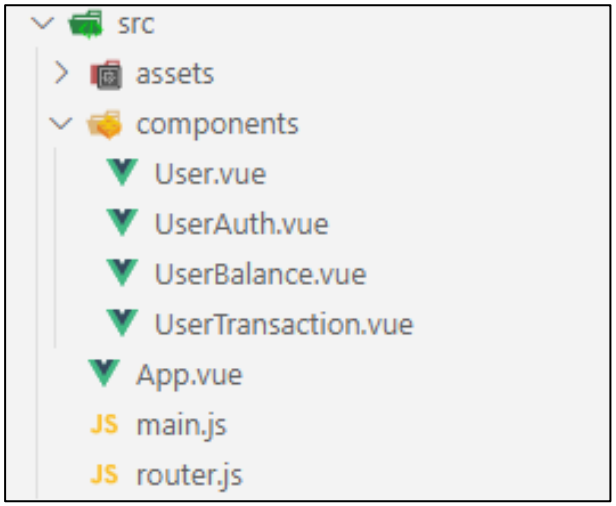
En esta sesión se trabajará sobre el componente construido en el primer acercamiento a Vue y se realizarán las siguientes acciones:

1. Creación de un componente que permita autenticar al usuario.
2. Creación de un componente que permita realizar una transacción.

En la próxima sesión estos componentes serán añadidos al componente principal y se notará su funcionamiento.

Estructura Actual

Para esta sesión se debe tener la siguiente estructura:



Crear 2 archivos vacíos: UserAuth.vue y UserTransaction.vue en el directorio /components.

Implementación del Componente UserAuth

Implementación del Componente UserAuth

Durante la implementación del componente UserAuth se debe tener en cuenta lo siguiente:

- Llamado y procesamiento de la petición HTTP.
- Uso de un formulario para la captura de los datos y del envío.

Componente UserAuth: Template

Componente UserAuth: Template

En el archivo UserAuth.vue definir el siguiente código:

Código: Template – Parte 1:

```
<template>
    <div id="AuthUser" class="auth_user">

        <div class="container_auth_user">
            <h2>Autenticarse</h2>
            <form v-on:submit.prevent="processAuthUser" >
                <input type="text"
                    v-model="user_in.username"
                    placeholder="Username">
                <br>
```

Notar el uso de v-on y de v-model.

Componente UserAuth: Template

En el archivo UserAuth.vue definir el siguiente código:

Código: Template – Parte 2:

```
                    <input type="password"
                           v-model="user_in.password"
                           placeholder="Password">
                    <br>
                    <button type="submit">Iniciar Sesión</button>
                </form>
            </div>
        </div>
    </template>
```

Notar el uso de v-model .

Componente UserAuth: Script

Componente UserAuth: Script

En el archivo UserAuth.vue definir el siguiente código:

Código: Script – Parte 1:

```
<script>
import axios from 'axios';
export default {
    name: "UserAuth",
    data: function(){
        return {
            user_in: {
                username:"",
                password:""
            }
        }
    },
    methods: {
```

Componente UserAuth: Script

En el archivo UserAuth.vue definir el siguiente código:

Código: Script – Parte 2:

```
processAuthUser: function(){
    var self = this

    axios.post("http://127.0.0.1:8000/user/auth/",
                self.user_in, {headers: {}})

        .then((result) => {
            alert("Autenticación Exitosa");
            self.$emit('log-in', self.user_in.username)
        })
```

Notar la definición de la petición y el bloque then.

Componente UserAuth: Script

En el archivo UserAuth.vue definir el siguiente código:

Código: Script – Parte 3:

```
                .catch((error) => {

                    if (error.response.status == "404")
                        alert("ERROR 404: Usuario no encontrado.");

                    if (error.response.status == "403")
                        alert("ERROR 403: Contraseña Erronea.");
                });
        }
    }
}
</script>
```

Notar el manejo de errores con el bloque catch.

Componente UserAuth: Style

Componente UserAuth: Style

En el archivo UserAuth.vue definir el siguiente código:

Código: Style – Parte 1:

```
<style>

    .auth_user{
        margin: 0;
        padding: 0%;
        height: 100%;
        width: 100%;

        display: flex;
        justify-content: center;
        align-items: center;
    }
```

Componente UserAuth: Style

En el archivo **UserAuth.vue** definir el siguiente **código**:

Código: Style – Parte 2:

```
.container_auth_user {
    border: 3px solid  #283747;
    border-radius: 10px;
    width: 25%;
    height: 60%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.auth_user h2{
    color: #283747;
}
```

Componente UserAuth: Style

En el archivo UserAuth.vue definir el siguiente código:

Código: Style – Parte 3:

```
.auth_user form{
    width: 50%;
}

.auth_user input{
    height: 40px;
    width: 100%;
    box-sizing: border-box;
    padding: 10px 20px;
    margin: 5px 0;
    border: 1px solid #283747;

}
```

Componente UserAuth: Style

En el archivo UserAuth.vue definir el siguiente código:

Código: Style – Parte 4:

```
.auth_user button{
    width: 100%;
    height: 40px;

    color: #E5E7E9;
    background: #283747;
    border: 1px solid #E5E7E9;

    border-radius: 5px;
    padding: 10px 25px;
    margin: 5px 0;
}
```

Componente UserAuth: Style

En el archivo UserAuth.vue definir el siguiente código:

Código: Style – Parte 5:

```
.auth_user button:hover{
    color: #E5E7E9;
    background: crimson;
    border: 1px solid #283747;
}

</style>
```

Implementación del Componente UserTransaction

Componente UserTransaction

Durante la implementación del componente UserTransaction se debe tener en cuenta lo siguiente:

- Llamado y procesamiento de la petición HTTP.
- Uso de un formulario para la captura de los datos y del envío.
- Captura de parámetros a través del Path.

Componente UserTransaction: Template

Componente UserTransaction: Template

En el archivo UserTransaction.vue definir el siguiente código:

Código: Template – Parte 1:

```
<template>
    <div id="UserTransaction">
        <div class="continer_user_transaction">
            <h2> Transacción {{username}}</h2>

            <form v-on:submit.prevent="processTransaction" >
                <input type="tetx" v-model="value" placeholder="Valor">
                <br>
                <button type="submit">Hacer Transacción</button>
            </form>

        </div>
    </div>
</template>
```

Componente UserTransaction: Script

Componente UserTransaction: Script

En el archivo UserTransaction.vue definir el siguiente código:

Código: Script – Parte 1:

```
<script>

import axios from 'axios';
export default {
    name: "UserTransaction",
    data: function(){
        return{
            username: "none",
            value: ""
        }
    },
```

Componente UserTransaction: Script

En el archivo UserTransaction.vue definir el siguiente código:

Código: Script – Parte 2:

```
methods:{
    processTransaction: function(){
        var self = this
        let transaction_in = {
            username: this.username,
            value: this.value
        }
```

Componente UserTransaction: Script

En el archivo UserTransaction.vue definir el siguiente código:

Código: Script – Parte 3:

```
axios.put("http://127.0.0.1:8000/user/transaction/",
            transaction_in,  {headers: {}})
     .then((result) => {
        alert("Transaction Correcta, Saldo Restante: " +
              result.data.actual_balance);
     })
     .catch((error) => {
         alert("ERROR Transaction Incorrecta: Saldo Insuficiente");
     });
   }
},
```

Componente UserTransaction: Script

En el archivo UserTransaction.vue definir el siguiente código:

Código: Script – Parte 4:

```
        created: function(){
            this.username = this.$route.params.username
        }

    }
    </script>
```

Notar la captura del atributo pasado por el path.

Componente UserTransaction: Style

Componente UserTransaction: Style

En el archivo UserTransaction.vue definir el siguiente código:

Código: Style – Parte 1:

```
<style>
    #UserTransaction{
        margin: 0;
        padding: 0%;
        height: 100%;
        width: 100%;

        display: flex;
        justify-content: center;
        align-items: center;

    }
```

Componente UserTransaction: Style

En el archivo UserTransaction.vue definir el siguiente código:

Código: Style – Parte 2:

```
.continer_user_transaction {
    border: 3px solid  #283747;
    border-radius: 10px;

    width: 25%;
    height: 60%;

    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

Componente UserTransaction: Style

En el archivo UserTransaction.vue definir el siguiente código:

Código: Style – Parte 3:

```
#UserTransaction h2{
    color: #283747;
}
#UserTransaction form{
    width: 50%;
}
#UserTransaction input{
    height: 40px;
    width: 100%;
    box-sizing: border-box;
    padding: 10px 20px;
    margin: 5px 0;
    border: 1px solid #283747;
}
```

Componente UserTransaction: Style

En el archivo UserTransaction.vue definir el siguiente código:

Código: Style – Parte 4:

```
#UserTransaction button{
    width: 100%;
    height: 40px;

    color: #E5E7E9;
    background: #283747;
    border: 1px solid #E5E7E9;

    border-radius: 5px;
    padding: 10px 25px;
    margin: 5px 0;
}
```

Componente UserTransaction: Style

En el archivo UserTransaction.vue definir el siguiente código:

Código: Style – Parte 5:

```
    #UserTransaction button:hover{
        color: #E5E7E9;
        background: crimson;
        border: 1px solid #283747;
    }
</style>
```

Referencias

- **[Vue.js]** (2020). Retrieved 6 December 2020, from https://es.vuejs.org/