

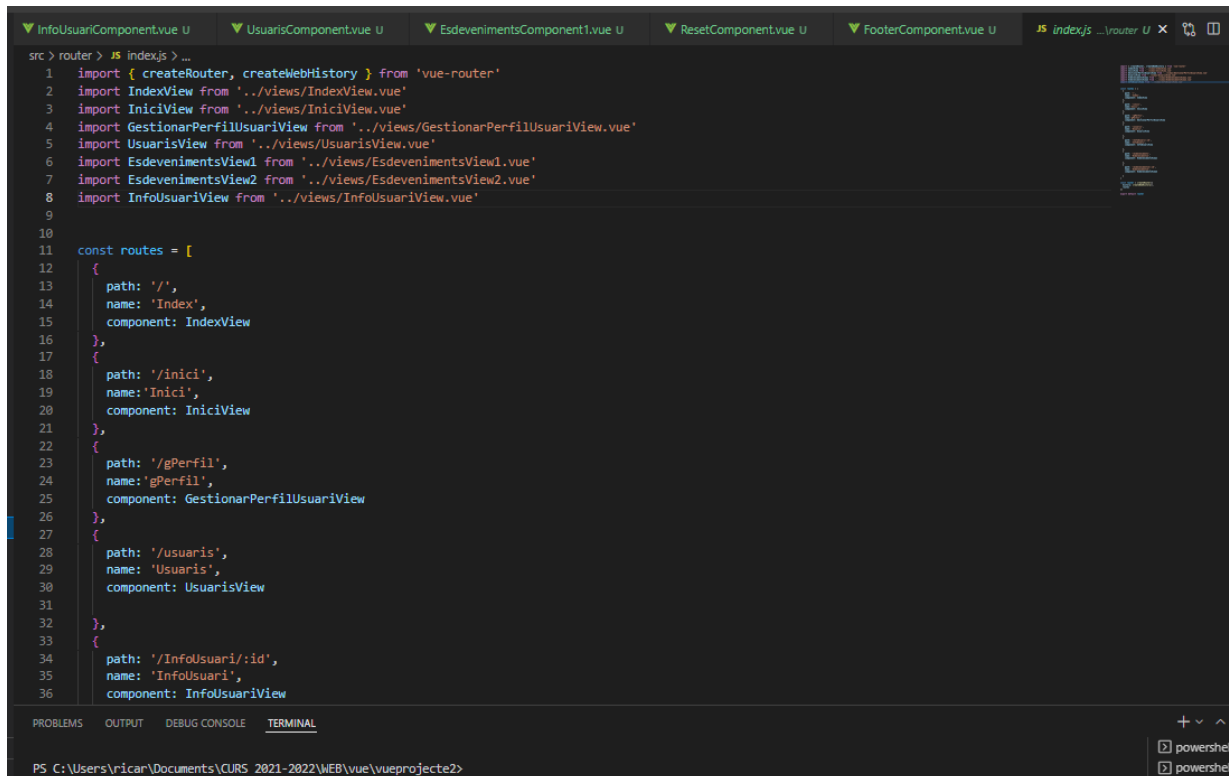
### 1. Introducció

L'objectiu d'aquesta pràctica és realitzar una pàgina web totalment funcional fent servir les tecnologies de JavaScript i Node.js. L'implementació d'aquesta en quan a llenguatge es refereix es demana l'ús de bones pràctiques, aplicar primitives, arrays, objecte, treballar amb el DOM, comunicació amb esdeveniments i callbacks a més de integrar programació funcional com són `forEach`, `map` i `reduce` arrays. També, en quan a la API, saber fer les diferents crides a aquesta (GET/POST/DELETE/PUT/etc...), així com l'ús de API Fetch i l'ús de bases de dades locals. Finalment, també l'ús de diferents llibreries en quan a disseny i demostrar saber fer servir l'eina Vue per aixecar un servidor local amb serveix estàtic de fitxers. Demostrar una capacitat de configuració de un projecte amb `Package.json` i aplicar `npm` per a la instal·lació de dependències.

### 2. Desenvolupament i resultats

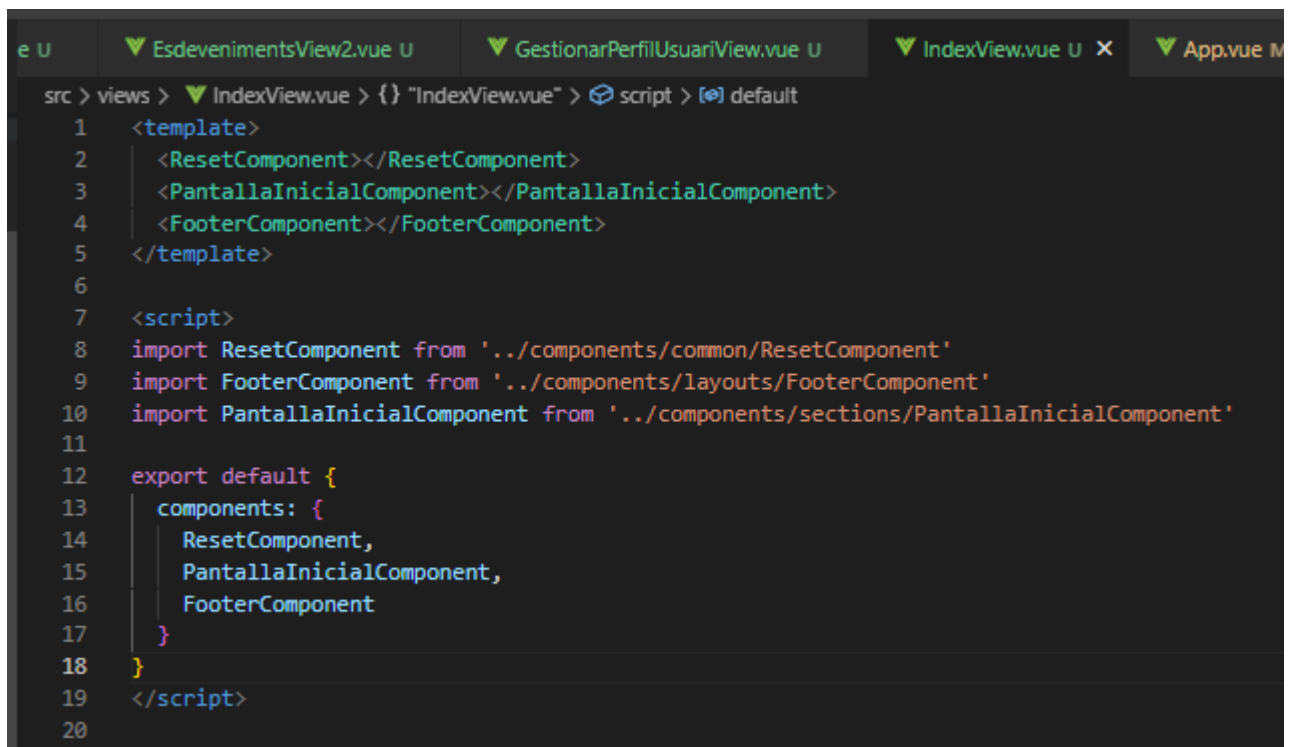
En quan al propi projecte, perquè sigui funcional, necessita interactuar amb la base de dades, per nodrir-se'n i per l'altre emmagatzemar-ne. Aquests dos requisits es veuen coberts gràcies a la disposició d'una API (back-end) que permetrà consultar i emmagatzemar dades fent que l'aplicació web tingui estat. Aquest back-end estarà disponible per a poder desenvolupar el front-end amb tecnologies web HTML, CSS i JavaScript. Per tant, el que s'haurà de dur a terme serà la implementació del front-end així com saber interactuar amb el back-end. Primer de tot, i abans de començar a programar codi, vam pensar com estructurar la nostra pàgina web i quines serien les parts de la que estaria composta. Per això, vam aplicar l'ús de components. Aquests ens permeten treure'ls o posar-los en tan sols una línia de codi i ens permet tenir-ho tot ben estructurat. Dins de la carpeta components vam separar aquests en 3 carpetes, la primera anomenada "common" on hi hauria els components que es repeteixen a tots els arxius, la segona anomenada "layouts" on tindrem el header i el footer i finalment una tercera anomenada "sections" on tindrem les diferents parts de la nostra pàgina web. També a part de la carpeta components, vam fer una altra de views. Aquesta contindrà totes les vistes que cridaran als respectius components per muntar l'aplicació web. També tindrem una carpeta router on s'establiran totes les rutes i les comunicacions dintre de la pròpia

web. En aquest fitxer, només calia establir el path, el nom i el component el qual es dirigia.



```
src > router > JS index.js > ...
1 import { createRouter, createWebHistory } from 'vue-router'
2 import IndexView from '../views/IndexView.vue'
3 import IniciView from '../views/IniciView.vue'
4 import GestionarPerfilUsuariView from '../views/GestionarPerfilUsuariView.vue'
5 import UsuarisView from '../views/UsuarisView.vue'
6 import EsdevenimentsView1 from '../views/EsdevenimentsView1.vue'
7 import EsdevenimentsView2 from '../views/EsdevenimentsView2.vue'
8 import InfoUsuariView from '../views/InfoUsuariView.vue'
9
10
11 const routes = [
12   {
13     path: '/',
14     name: 'Index',
15     component: IndexView
16   },
17   {
18     path: '/inici',
19     name: 'Inici',
20     component: IniciView
21   },
22   {
23     path: '/gPerfil',
24     name: 'gPerfil',
25     component: GestionarPerfilUsuariView
26   },
27   {
28     path: '/usuaris',
29     name: 'Usuaris',
30     component: UsuarisView
31   },
32   {
33     path: '/InfoUsuari/:id',
34     name: 'InfoUsuari',
35     component: InfoUsuariView
36   }
37 ]
```

També un exemple de una vista que crida els diferents components, prèviament amb la corresponent importació:



```
src > views > IndexView.vue > {} "IndexView.vue" > script > default
1 <template>
2   <ResetComponent></ResetComponent>
3   <PantallaInicialComponent></PantallaInicialComponent>
4   <FooterComponent></FooterComponent>
5 </template>
6
7 <script>
8   import ResetComponent from '../components/common/ResetComponent'
9   import FooterComponent from '../components/layouts/FooterComponent'
10  import PantallaInicialComponent from '../components/sections/PantallaInicialComponent'
11
12  export default {
13    components: {
14      ResetComponent,
15      PantallaInicialComponent,
16      FooterComponent
17    }
18  }
19 </script>
20
```

A continuació, ens els diferents components es composaven de una part de “template” on està tota la part de HTML, una altre part de “script” on tindrem tota la lògica que fa funcionar aquell component (amb les diferents crides a la API), i finalment la part de “style” on tindrem tots els estils que s’aplicaran al HTML (CSS). En quan al HTML i CSS vam aplicar diferents classes perquè la nostra pàgina web quedés de la forma que volíem. Cal destacar que en un inici, no veiem que tot estigués al seu lloc i això era perquè en cada component, en el tag de style, s’havia d’aplicar “scoped”. Això, fa que tot el estil que estigui dintre d’aquest tag s’apliqui solament en aquell component i per tant, no generi col·lisions amb altres components que és el que ens estava passant.

```
54
55 <style scoped>
56 a {
57   text-decoration: none;
58 }
59
60 body {
61   background: -webkit-linear-gradient(bottom, #2a0aa0, #07ffc9);
62 }
63
64 html {
65   height: 100%;
66 }
67
68 main {
69   background: -webkit-linear-gradient(bottom, #2a0aa0, #07ffc9);
70   background-repeat: no-repeat;
71   display: flex;
72   align-items: center;
73   min-height: 100vh;
74 }
75
76 label {
77   font-family: "Raleway", sans-serif;
78   font-size: 14pt;
79 }
80 .card {
81   background: #fbfbfb;
82   border-radius: 8px;
83   box-shadow: 1px 2px 8px rgba(0, 0, 0, 0.65);
84   height: 600px;
85   margin: 6rem auto 8.1rem auto;
86   width: 1000px;
87 }
```

El que més temps ens va dur a terme, va ser la part de “script” on vam haver de fer les diferents crides a la API. Tot i que en un inici no sabíem molt bé com fer-ho poc a poc vam anar aprenent sobre com es feien les crides, les respostes d’aquestes, així com les promeses (async/await), i també la part de beforeCreate() referent a quin és el procés que segueix una pàgina web per crear-se. Així doncs, poc a poc vam anar realitzant les diferents crides. És important destacar el fet de per totes les crides es necessitava un accessToken. Aquest, és un token que ens dona la informació que l’usuari ja s’ha loguejat correctament i per tant pot veure la informació de la nostra pàgina web. Per tenir aquest accessToken, quan vam fer la crida a la API del login, en la resposta d’aquesta ens proporcionava aquest. Per tenir-lo en tot moment, vam guardar-lo a localStorage. Així doncs, en cada crida recuperàvem aquest accessToken i li passàvem com a Header. També vam controlar diferents errors

que podia tenir la resposta de la nostra API com el 400, 500, etc... així com també mostrar a l'usuari diferents errors que pot estar cometent ell a la hora de la entrada de informació en la nostra pàgina web. Per fer un petit resum de tot lo que està composta de la nostra web primer tenim un component anomenat PantallaInicialComponent el qual tindrà el fet de loguejar-se i registrar-se. Per poder fer-ho vam tenir que aplicar un form on l'usuari podia introduir les seves dades i nosaltres agafàvem cada camp el qual omplia l'usuari i el guardàvem en un objecte anomenat en el nostre cas usuarioLogin i usuarioRegister. El fer click al botó corresponent es realitzava la crida a la API passant-li com a body tot aquest objecte com a JSON utilitzant el stringify. Si la resposta era correcta retornàvem aquesta i guardàvem el accessToken en el cas de Login al localStorage.

```
async login() {
  this.error = [];

  await fetch("http://puigmal.salle.url.edu/api/v2/users/login", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(this.usuarioLogin),
  })
  .then((res) => {
    if (res.status == 200) {
      return res.json();
    } else {
      throw new Error("Incorrect credentials. Please try again!");
    }
  })
  .then((data) => {
    localStorage.setItem("accessToken", data.accessToken);
    this.getUsers();
  })
  .catch((error) => {
    console.log(error);
    if (!this.error.includes(error.message)) {
      this.error.push(error.message);
    }
  });
},
```

Un cop l'usuari es loguejava correctament el redirigim a IniciComponent on l'usuari podrà escollir anar a 3 seccions diferents: Gestionar Perfil Usuari, Usuaris o Esdeveniments. En cada cas, s'utilitzarà el <router-link> to="nom\_component", per anar al component desitjat. En el cas de Gestionar Perfil d'Usuari vam mostrar per pantalla la informació de l'usuari (nom, cognom i correu) així com les diferents estadístiques personals. Per fer això cal dir primerament que cal aplicar el beforeCreate() comentat anteriorment, el qual permet fer les crides i mostrar tot per pantalla abans de que l'usuari pugui fer res. Així doncs, quan l'usuari vegi per primer cop la pantalla en qüestió veurà tot lo que s'ha fet en el beforeCreate(). També cal comentar que per fer aquestes crides, necessitàvem passar la ID de usuari loguejat. Per fer-ho en PantallaInicialComponent vam fer la crida a la API de getUsers(), el qual ens retornava la informació dels diferents usuaris dintre la qual es trobava la id. Per trobar la ID exacte de l'usuari que s'havia loguejat vam haver de filtrar per cada usuari el correu fins trobar el que era igual i guardar la ID en el localStorage. Així doncs, en GestionarPerfilComponent recuperàvem aquesta ID i podien fer les diferents crides a la API esmentades. També vam fer la funcionalitat d'eliminar compte. En quan a UsuariComponent, vam mostrar per pantalla tots els usuaris i també la funcionalitat de poder veure el perfil més detalladament fent click a un botó com

també enviar i eliminar peticions d'amistat. També vam implementar en el cas de les amistats mostrar per pantalla diferents alertes indicant a l'usuari si havia enviat o no la petició determinada. Finalment, en quan a Esdeveniments, vam mostrar tots els esdeveniments amb la seva respectiva informació com era la imatge, el nom, entre altres. També vam implementar un timeline el qual mostrava el Top 5 d'esdeveniments segons el "avg\_score", una dada que depenia de la puntuació que donaven els diferents usuaris a cada esdeveniment. En la part superior d'aquesta pantalla també teníem dos botons els quals si li donaven es desplegava diferent informació respectivament. Un botó servia per buscar a través de un cercador els esdeveniments que l'usuari volgués. En aquest cas, es pot filtrar per data, per nom de l'esdeveniment o per paraula clau. L'altre botó obre un desplegable per crear un esdeveniment. En aquest cas, l'usuari ha d'omplir tots els camps i donar-li al botó de crear esdeveniment. Finalment, si es fa click en un esdeveniment en concret ens redirigeix a una altre pantalla on es pot veure el esdeveniment en qüestió amb més informació detallada d'aquest.

### 3. Tecnologies i eines

Les tecnologies usades principalment són Visual Studio Code i Vue juntament amb Node.js. Visual Studio Code és un editor de codi, el qual ens ha permès codificar tota la pràctica. També el Node.js s'ha utilitzat per el back-end de la API.

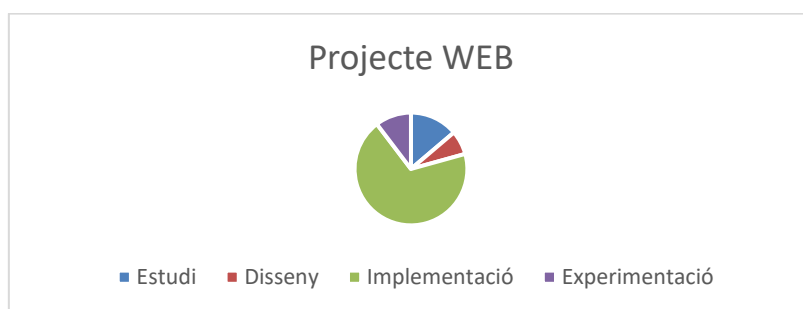
### 4. Costos

Estudi: Aquesta categoria comprèn tot allò que té a veure amb l'aprenentatge previ a la pràctica. És la part teòrica. Temps aproximat de: 8h.

Disseny: Comprèn els pensaments i esbossos previs a l'execució i la implementació de la pràctica. Saber com s'estructurarà, quines parts tindrem i quins són els diferents passos per dur a terme. Temps aproximat: 4h.

Implementació: És tota la part pràctica del projecte. Comprèn el HTML, CSS i els diferents scripts que contenen tota la lògica de les APIS. Temps aproximat: 40h.

Experimentació: És la part on hem invertit en buscar noves formes d'estils, de diferents maneres de codificar els diferents components per tal de que l'aspecte sobretot fos de la manera que desitjàvem. Temps invertit aproximat: 6h.



## 5. Conclusions

Amb aquesta pràctica he après a implementar una pàgina web des de zero. El fet de separar-ho amb components ha fet que la implementació de la pràctica sigui més entenedora i més fàcil per saber que s'està fent en tot moment. També, en quan a que un component és tot un bloc el qual pots implementar o no a través de una sola línia i també es pot reutilitzar en altres llocs cosa molt important per no repetir codi. També he après a donar-li estils a la pàgina web amb el CSS, a través de la implementació de diferents classes. A més a més, he après la comunicació amb la API (back-end) i a com utilitzar la informació que rebo per implementar diferents funcionalitats. No només la informació rebuda, sinó també com agafar informació que l'usuari entra per teclat i utilitzar aquesta per passar-li a les diferents crides de la API. Finalment, dir que el fet de ser un projecte en grup ens ha permès tant el meu company com a mi, fer pair-programming, és a dir, treballar alhora mentre un feia un component de la pàgina web, l'altre anava programant un altre i així avançar més ràpid. El fet de ser un projecte d'aquest àmbit, ha provocat que hem tingut que buscar diferent informació a internet i per tant, hem après molt també d'aquesta vessant, ja que és un àmbit que canvia constantment i per tant, hem d'estar informats en tot moment de qualsevol canvi que pugui haver.

## 6. Competències

### Competències generals

CG01	Capacitat d'anàlisi i síntesi.
CG06	Presa de decisions.

### Competències específiques

CE2	Dissenyar, implementar i mantenir sistemes informàtics a través de l'aplicació apropiada dels paradigmes, entorns i llenguatges de programació, així com de les eines de suport afins.
CE4	Desenvolupar i utilitzar components software intercanviables, reutilitzables i definits per interfícies que permetin la composició i cooperació de sistemes informàtics.
CE6	Aplicar fonaments del disseny gràfic, la usabilitat i accessibilitat en el desenvolupament de solucions informàtiques que incrementin el grau de satisfacció i l'experiència d'ús de l'usuari.