

## Projectes Web I

3r Curs - Semestre 1 (2021/2022)

Grup 9: Curial Iglesias, Albert Armengol, Jonathan Barragan

### 1. Introducció

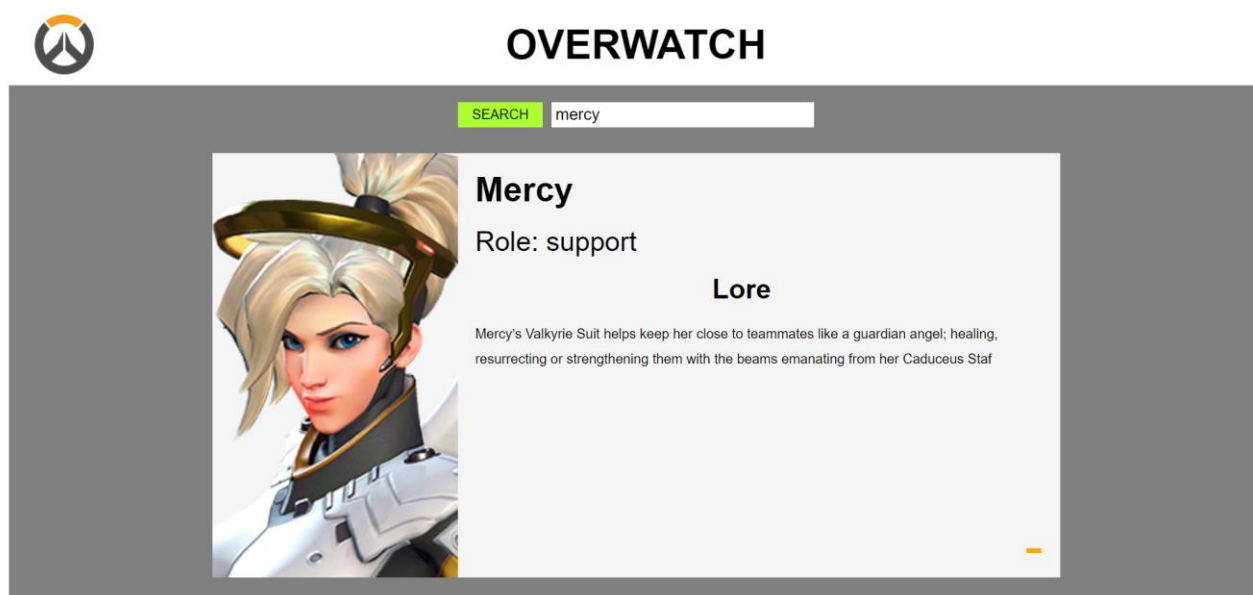
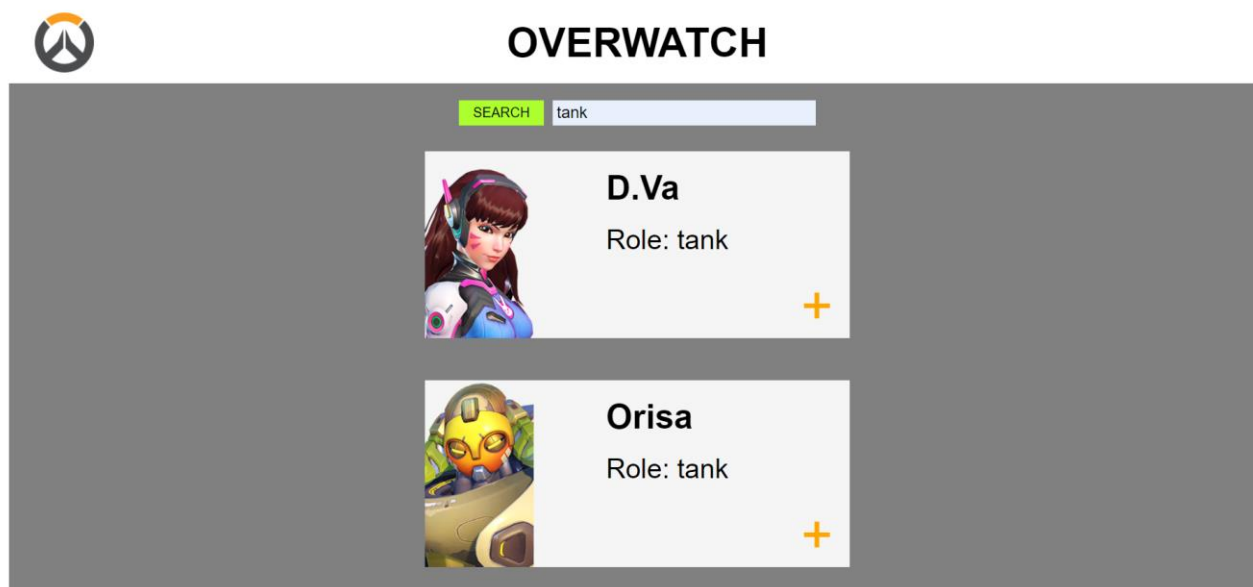
Aquest projecte consisteix en la realització d'una aplicació web que permeti al usuari obtenir informació sobre personatges del videojoc "Overwatch".

Per realitzar aquest projecte s'utilitzen els llenguatges HTML, CSS i JavaScript.

L'aplicació web es comunica de forma asíncrona amb una API local i aquesta carrega un .json amb tota la informació dels herois.

### 2. Desenvolupament i resultats

Primer de tot mostrem unes imatges per tenir una visió general de com és l'aplicació web:



Els arxius que hem implementat en aquest projecte son:

- HTML: index.html
- CSS: style.css
- JS: index.js, search.js, list.js i info.js

En el **index.html** tenim el header de la web (imatge + logo de Overwatch) i en el main únicament un <div> amb id="App" per poder afegir i eliminar informació utilitzant Vue. També li indiquem que utilitzem un style.css extern, un index.js extern i importem Vue 2.

En el **style.css** apliquem estils a totes les semàntiques HTML, ja estiguin en el propi index.html o introduïdes desde JavaScript utilitzant Vue. A destacar podem dir que hem utilitzat media queries per amplades de pàgina màxima de 1050px, 600px i 460px per així aconseguir una pagina responsive per tots els tamanyes de pantalla. també hem utilitzat un keyframe per animar una rotació.

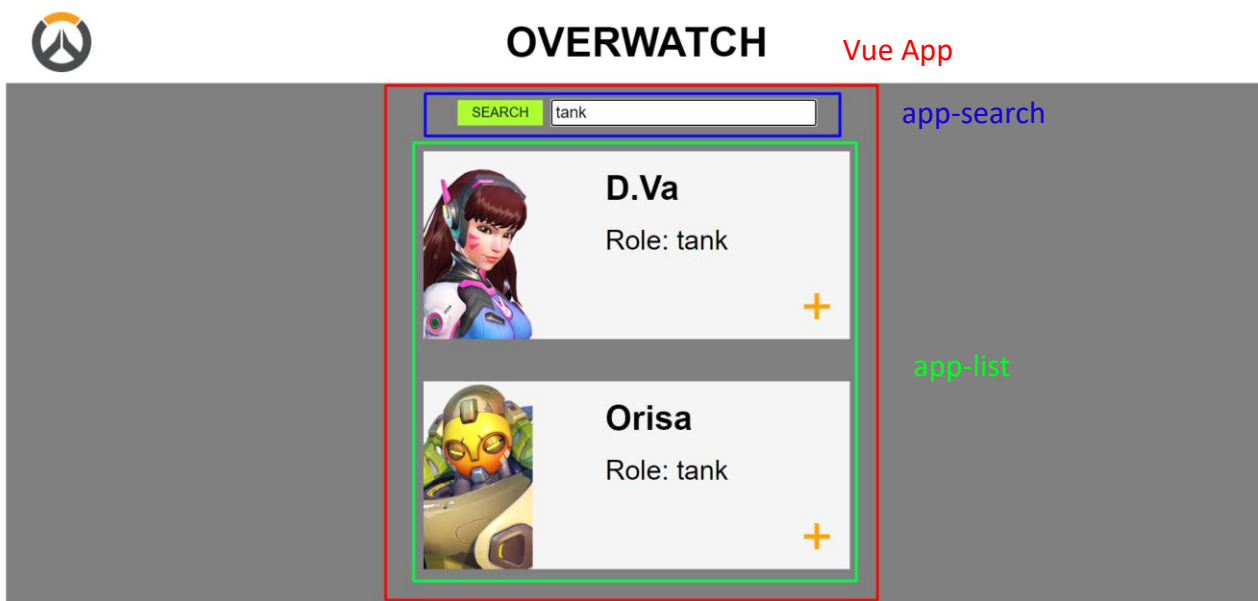
**index.js** es el nostre arxiu JavaScript principal ja que es on creem la instancia Vue com a singleton. Aquest index.js te com a fills search.js, list.js i info.js. Les variables que declarem i podran ser utilitzades per els fills son:

- heroes: Un array d'objectes heroi (key, name, role, portrait, message).
- refresh: Un bool que utilitzem per indicar si s'ha de refrescar els herois que es mostren.

També inclou el següent template html:

```
<div>
  <app-search></app-search>
  <div class="hero-list" v-for="hero in heroes">
    <app-list :hero="hero"></app-list>
  </div>
</div>
```

app-search correspon al vue component de search.js i app-list al vue component de list.js.

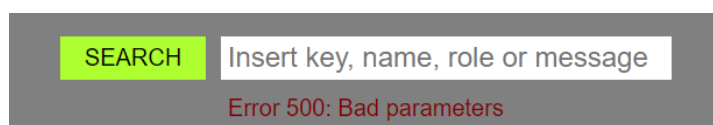


**search.js** és un vue component anomenat app-search que s'encarrega de fer una crida GET a la api a partir d'un String que introdueix l'usuari. Te les següents variables pròpies:

- textToSearch: String que introdueix l'usuari.
- errorMessage: String que es mostrarà en cas que hi hagi error.

També inclou el següent template html:

```
<div>
  <form v-on:submit.prevent="submitForm">
    <button id="search">SEARCH</button>
    <input type="input" v-model="textToSearch" id="textToSearch"
placeholder="Insert key, name, role or message">
  </form>
  <p id="errorMessage">{{ errorMessage }}</p>
</div>
```



The screenshot shows a web form with a green button labeled 'SEARCH'. Next to it is a text input field with the placeholder text 'Insert key, name, role or message'. Below the input field, there is a red error message that reads 'Error 500: Bad parameters'.

Te un formulari el qual consta d'un botó search i un input que assignem a la variable textToSearch.

També un paràgraf que mostra un missatge d'error donat per la variable errorMessage.

Aquest component utilitza un mètode submitForm() que s'activa cada cop que s'apreta el botó search:

```
async submitForm() {
  try {
    const response = await fetch(`http://localhost:3000/api/list?search=${textToSearch.value}`)
    .then((response) => {
      if (response.status !== 200) {
        this.errorMessage = `Error ${response.status}: Bad parameters`;
        throw new Error("Bad parameters");
      }
      return response;
    })
    this.data = await response.json()
    .then((data) => {
      console.log(`Buscant herois amb el terme: ${textToSearch.value}`);
      console.log("Llista d'herois obtinguts:");
      console.log(data);
      this.$root.refresh = true;
      this.$root.$emit('refresh');
      this.$root.refresh = false;
      this.$root.heroes = data;
      this.errorMessage = "";
    });
  } catch (e) {
    console.log(e);
  }
}
```

Aquest mètode es comunica de forma asíncrona amb el mètode GET de la API. La API rep com a paràmetre la variable `textToSearch` i retorna una llista d'herois que coincideixen amb l'string, finalment passem aquesta llista d'herois al array d'objectes `heroes` del pare.

Quan la variable `textToSearch` estigui buida, la API retornarà un error 500 el qual nosaltres detectem i parem l'execució del mètode. En cas que es produeixi aquest error mostrarem tant per consola com per pantalla "Error 500: Bad paràmetres".

Just abans d'enviar la llista d'herois al pare també fem un emit per indicar que s'han de refrescar la llista d'herois que teníem anteriorment.

**list.js** es un vue component anomenat `app-list` que s'encarrega de mostrar la llista d'herois i fer una crida POST a la API sempre que es demani. Te les següents variables pròpies:

- `symbol`: Es el valor que es mostra en el botó d'info i pot tenir els estats "+", "~" o "-".
- `lore`: String que amaga o mostra el terme "Lore".
- `message`: String que amaga o mostra la informació extra d'un heroi.
- `expanded`: Bool que indica si s'ha de mostrar informació extra d'un heroi o no.
- `loaded`: Bool que utilitzem per l'efecte de transició de mostrar més informació.

També rep com a paràmetre un objecte `hero` que no es més que un objecte del array d'objectes `heroes` del pare.

També inclou el següent template HTML:

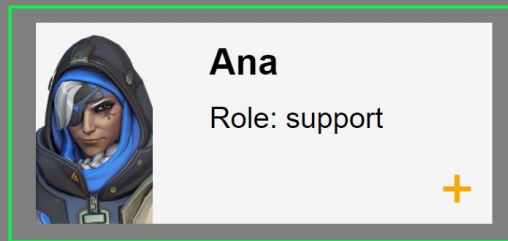
```
<div class="hero" v-bind:class="{ 'heroExpand': expanded }">
  
  <div>
    <h2 class="name"> {{ this.hero.name }} </h2>
    <p class="role"> Role: {{ this.hero.role }} </p>
    <app-info :message="message" :lore="lore" class="message"></app-info>
  </div>
  <button class="btn-more" v-bind:class="{ 'anim': loaded }" v-
on:click.prevent="infoButton">{{ symbol }}</button>
</div>
```

Te un div que mostra informació del hero. Aquest div te 2 estats segons la class que tingui assignada:

- `hero`: Mostra imatge, nom i rol.
- `heroExpand`: mostra la informació anterior + el message

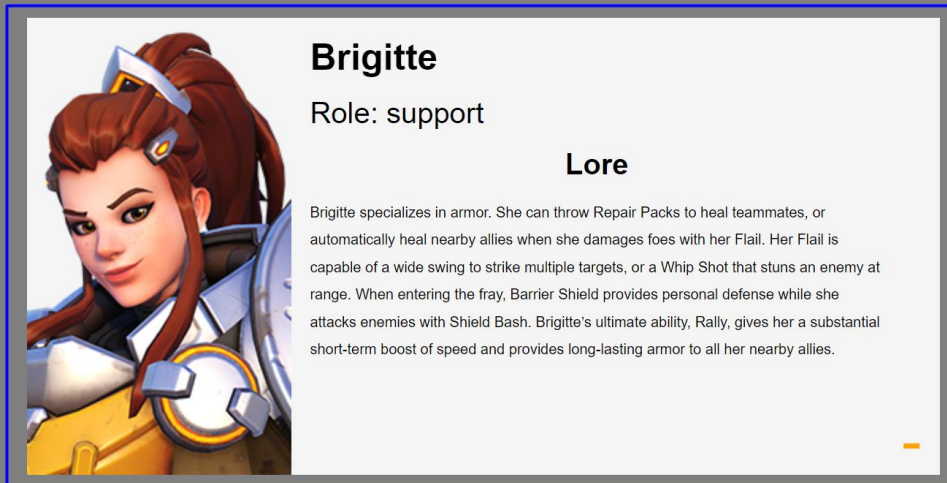
També te un botó `btn-more` amb 3 estats:

- informació reduïda: El botó es +
- carregant informació: El botó es ~
- informació ampliada: El botó es -



hero

heroExpand



Aquest component utilitza un mètode `infoButton()` que s'activa cada cop que s'apreta el botó `btn-more` o quan rep un `emit` de que s'ha de refrescar la llista d'herois:

```
async infoButton() {
  if (this.symbol == "-" || this.$root.refresh == true) {
    this.expanded = false;
    this.hideMessage();
  } else {
    try {
      this.loading();
      const parameters = {
        method: "POST",
        headers: {"Content-type": "application/json"},
        body: JSON.stringify({key: this.hero.key})
      };
      const response = await fetch(`http://localhost:3000/api/detail`, parameters)
      .then((response) => {
        if (response.status != 200) {
          this.errorDetect = true;
          throw new Error("Bad parameters");
        }
        return response;
      })
      this.data = await response.json()
      .then((data) => {
        this.hero.message = data[0].message;
        console.log(`Carregant informació extra de ${this.hero.name}:`);
        console.log(this.hero.message);
        setTimeout(() => {this.showMessage();}, 1200);
        this.$root.refresh = false;
      });
    } catch (e) {
      console.log(e);
    }
  }
}
```

Aquest mètode es comunica de forma asíncrona amb el mètode POST de la API. La API rep com a paràmetre la variable key de l'objecte hero i retorna el message del hero que li hem introduït, finalment assignem aquest message a l'atribut message del nostre objecte hero.

Quan la variable key estigui buida, la API retornarà un error 500 el qual nosaltres detectem i parem l'execució del mètode. En cas que es produeixi aquest error mostrarem per consola "Error 500: Bad paràmetres".

Però aquest mètode no sempre fa la crida a la API, depèn del estat en el que estigui el botó btn-more en el moment d'apretar-lo. Si esta en estat "+" fa la crida a la API i seguidament assigna al <div class="hero"> una nova classe "heroExpand" que fa que augmenti el tamany de la targeta del heroi i també mostri el message retornat per la API. En cas que el btn-more estigui en estat "-" el que farà serà treure-li la class "heroExpand" i per tant amagar la informació del message.

El mètodes loading(), showMessage() i hideMessage() que es veuen a la captura simplement els utilitzem per assignar nous valors a les variables que controlen el símbol del botó i si el heroi esta expandit o no:

```
loading() {
  this.symbol = "~";
  this.loaded = true;
},

showMessage() {
  this.loaded = false;
  this.symbol = "-";
  this.lore = "Lore";
  this.message = this.hero.message;
  this.expanded = true;
},

hideMessage() {
  this.symbol = "+";
  this.lore = "";
  this.message = "";
},
```

Finalment tenim un mounted() que fa que s'executi el mètode infoButton() que acabem d'explicar en cas que es detecti un senyal de refresh, activat per la funció submitForm() en el component app-search.

```
mounted(){
  const thisInstance = this
  this.$root.$on('refresh', function(){
    thisInstance.infoButton();
  })
}
```

Això ho fem per a que quan s'apreti el botó search es minimitzin tots els herois que estiguessin despleats en aquell moment per evitar que es sobreescrigui informació dels nous herois.

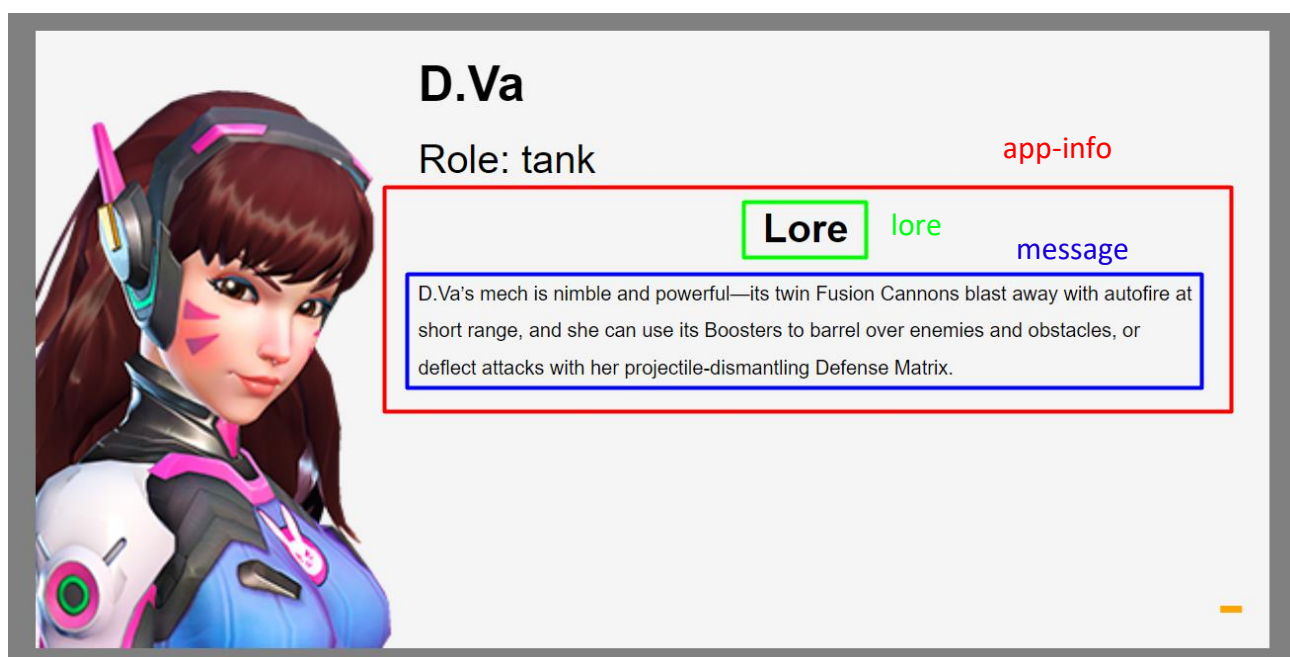
**info.js** es l'últim vue component anomenat `app-info` i simplement s'encarrega de mostrar un títol Lore i el missatge de l'heroi en cas que es demani. Té com a propietats `lore` i `message`.

També inclou el següent template html:

```
<div>
  <h3>{{ lore }}</h3>
  <p>{{ message }}</p>
</div>
```

Aquestes propietats les rep desde el component `app-list` que en el seu template recordem que tenia el següent:

```
<app-info :message="message" :lore="lore" class="message"></app-info>
```

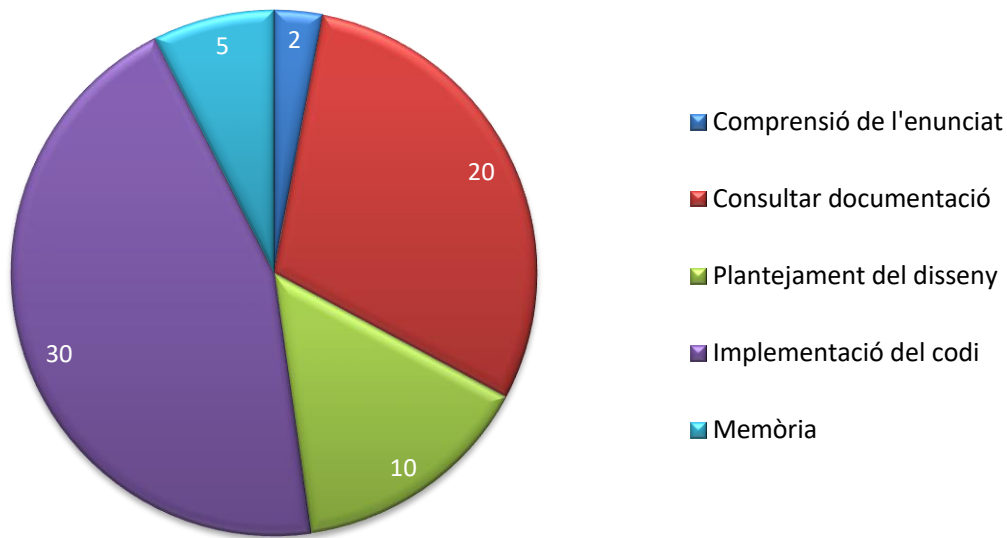


### 3. Tecnologies i eines

- Bitbucket: Es l'eina que hem utilitzat com a repositori per poder passar-nos el projecte entre els membres del grup.
- HTML: Hem utilitzat aquest llenguatge per indicar-li al navegador quins elements ha de carregar a la aplicació web.
- CSS extern: Hem utilitzat CSS extern per aplicar estils als elements de semàntica del HTML, tant del propi .html com dels introduïts des de template de Vue.
- Vue: Hem utilitzat el framework Vue, el qual utilitzem com a singleton per activar reactivitat entre els Vue components.
- API: Hem fet una comunicació `async/await` amb la API utilitzant `fetch()` per fer un HTTP request, tant pel mètode GET com pel POST.

## 4. Costos

### Cost - Estimació d'hores



- Comprensió de l'enunciat: Comprendre i entendre tots els apartats de l'enunciat que inclou la finalitat de la web, estructura data.json, la api, els components a realitzar i els requeriments de l'entrega.
- Consultar documentació: Tota la documentació que hem necessitat consultar per la realització del projecte sobre JavaScript, Vue i APIs. La majoria de pàgines que hem consultat són MDN, vue.org, stack overflow i explicacions de youtube.
- Plantejament del disseny: Aquesta part consisteix en planificar com estructurar el codi així com quins components Vue utilitzar i les funcions de cada un.
- Implementació del codi: Implementació del codi HTML, CSS i JavaScript. Amb diferència el llenguatge al que li hem dedicat més temps és JavaScript ja que és el que més es requeria per aquest projecte i també el que dominem menys.
- Memòria: La realització d'aquesta memòria / portfoli.



## 5. Conclusions

Aquest projecte posa a prova la capacitat dels alumnes que han d'utilitzar tots els conceptes apresos a classe, i també buscar-ne de nous, per poder resoldre satisfactòriament tots els problemes que planteja aquest projecte.

Hem après a entendre el funcionament de les APIs i com comunicar-nos amb elles, així com controlar els possibles errors que poden retornar. A utilitzar el framework Vue i com fer la comunicació entre components per poder mostrar informació a l'aplicació web de forma dinàmica.

També ens ha ajudat a millorar el nostre nivell amb el llenguatge JavaScript i hem reforçat els coneixements que ja teníem sobre HTML i CSS.

Criem que hem assolit aquests coneixements satisfactòriament i estem contents amb el resultat d'aquest projecte.