
	Nom du Module	
	BILAN TP DE PROGRAMMATION	

**Nom : Levent**

**Prénom : Kyllian**

**Nom du TD/TP : Vue.JS Superhero API**

## I. Instructions

### Format

- Le bilan peut être **rédigé** sous forme de document écrit ou présenté sous forme de diaporama si cela est plus adapté.
- Le document final doit être soumis au format **PDF**. Assurez-vous que les fichiers sont bien nommés (évitez les espaces et utilisez des noms explicites, ex. : **bilanTP\_NomPrenom.pdf**).

### Clarté et Précision

- Assurez-vous que toutes les sections sont complétées avec soin. Prenez le temps de rédiger des explications claires et argumentées.
- Présentez les informations dans un ordre logique, avec des titres et sous-titres bien marqués pour faciliter la lecture.

### Respect des délais

- Remettez votre bilan dans les délais impartis pour permettre une évaluation rapide et efficace. Les travaux remis en retard peuvent ne pas être pris en compte.

## II. Fonctionnalités Implémentées

**Liste des fonctionnalités développées et présentez des copies d'écran de votre application finale :**

- Affichage : Affiche la liste de tout les superhero
- Stats : une checkbox pour afficher les stats des super héros
- Recherche : permet de rechercher un superhero avec le nom
- Details : permet d'afficher tout les details d'un superhero
- Power matcher : permet de rechercher des superhero selon 3 critères de statistiques

SuperHeroes

Liste des superhéros






Détail d'un superhero

SuperPower matcher


## Liste des superhéros

☐Afficher les pouvoirs

Reinitialiser

	A-Bomb	1
	Abe Sapien	2
	Abin Sur	3
	Abomination	4
	Abraxas	5

SuperHeroes



### A-Bomb

**Power Stats**

intelligence : 38	strength : 100
speed : 17	durability : 80
power : 24	combat : 64

**Appearance**

gender : Male
race : Human
height : 6'8 / 203 cm
weight : 980 lb / 441 kg
eyeColor : Yellow
hairColor : No Hair

**Biography**

fullName : Richard Milhouse Jones
alterEgos : No alter egos found.
aliases : [ "Rick Jones" ]
placeOfBirth : Scarsdale, Arizona
firstAppearance : Hulk Vol 2 #2 (April, 2008) (as A-Bomb)

**Biography**

fullName : Richard Milhouse Jones
alterEgos : No alter egos found.
aliases : [ "Rick Jones" ]
placeOfBirth : Scarsdale, Arizona
firstAppearance : Hulk Vol 2 #2 (April, 2008) (as A-Bomb)

**Work**

occupation : Musician, adventurer, author; formerly talk show host
base : -

**Connections**

groupAffiliation : Hulk Family; Excelsior (sponsor), Avengers (honorary member); formerly partner of the Hulk, Captain America and Captain Marvel; Teen Brigade; ally of Rom
relatives : Marlo Chandler-Jones (wife); Polly (aunt); Mrs. Chandler (mother-in-law); Keith Chandler, Ray Chandler, three unidentified others (brothers-in-law); unidentified father (deceased); Jackie Shorr (alleged mother; unconfirmed)

Retour à la liste

Hero précédent

Hero suivant

## Power Matcher

Intelligence: 50




Strength: 50

Speed: 50

Tolerance (±): 10



Find Matching Heroes

## Matching Heroes (9)

	Animal Man	28
intelligence: 56 strength: 48 speed: 47 durability: 85 power: 73 combat: 80		
	Arachne	39
intelligence: 50 strength: 48 speed: 50 durability: 70 power: 71 combat: 70		
	Darth Maul	207

## Description détaillée :

- Chaque capture d'écran doit être accompagnée d'une explication claire et détaillée.
- Expliquez le rôle de chaque fonctionnalité (ex. : à quoi elle sert, pour qui elle est destinée).
- Précisez son fonctionnement (ex. : interaction utilisateur, logique derrière la fonctionnalité) et comment vous l'avez programmé.
- Mentionnez les problèmes rencontrés lors de son implémentation et les solutions apportées.

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

- Fonctionnalités visibles : pour chaque capture, précisez les fonctionnalités mises en avant (ex. : barre de navigation, formulaire, tableau de données).
- Aspects techniques : expliquez les choix techniques réalisés (ex. : technologies utilisées, structure du code, organisation des composants) et leur impact sur l'efficacité ou la performance de l'application.
- Commençons par la présentation de l'index :

```

index.html > html > body > div#app.container > div.list-group.my-3
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
7      <script src="https://unpkg.com/axios@latest"></script>
8      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
9      <title>Superheroes vue.js</title>
10 </head>
11 <body>
12     <!-- Add navbar container -->
13     <div id="navbar-container"></div>
14
15     <div id="app" class="container">
16         <h1 class="my-4">Liste des superhéros</h1>
17
18         <input type="checkbox" id="showStats" v-model="printpower">
19         <label for="showStats">Afficher les pouvoirs</label>
20
21         <div class="my-3">
22             <input type="text" class="form-control" placeholder="Rechercher un superhéros" v-model="searchQuery">
23             <button class="btn btn-primary" @click="searchQuery = ''>Reinitialiser</button>
24         </div>

```

- Dans le head on y import les composant vue.js et le bootstrap.
- Le premier element du body est la div qui va contenir la navbar qui se trouve dans un autre fichier html que l'on importe plus bas dans le fichier.
- On démarre ensuite la div app dans laquelle on a le titre, le bouton qui permettra d'afficher les stats puis la barre de recherche avec son bouton de réinitialisation.
- On a ensuite la div qui va afficher les super héros :

```

26     <div class="list-group my-3">
27       <div class="list-group-item" v-for="hero in filteredSuperheroes"
28         :key="hero.id"
29         @click="goToDetail(hero.id)"
30         style="cursor: pointer">
31         <div class="d-flex align-items-center justify-content-between mb-2">
32           <div class="d-flex align-items-center gap-3">
33             
38             <h5 class="mb-0">{{ hero.name }}</h5>
39           </div>
40           <small>{{ hero.id }}</small>
41         </div>
42         <div v-if="printpower" class="d-flex gap-2">
43           <small v-for="(value, key) in hero.powerstats"
44             :key="key">
45             {{key}}: {{value}}
46           </small>
47         </div>
48       </div>
49     </div>
50
51   </div>

```



- La div est organisé de manière a ce que le style fonctionne le mieux possible pour que le site soit propre.
  - Dans la seconde div on a l'element vue.js qui est un for pour afficher les hero dans le filltre que l'ont vera plus bas dans le code.
  - On ouvre ensuite une liste
  - Chaque element de la liste a l'image puis le nom puis l'id et ensuite le stats.
  - Attention les stats on un element vus.js « if » car les stats doivent être afficher uniquement si la checkbox est checked.
  - La div app est ensuite fermé a la fin.
- 
- Voyint maintenant le code vue.js de cette page :

```

53 <script>
54   const app = Vue.createApp({
55     data() {
56       return {
57         superHeros: [],
58         printpower: false,
59         searchQuery: ''
60       },
61     },
62     computed: {
63       filteredSuperheros() {
64         return this.superHeros.filter(hero =>
65           hero.name.toLowerCase().includes(this.searchQuery.toLowerCase())
66         )
67       },
68     },
69     methods: {
70       handleError(e) {
71         e.target.src = 'https://via.placeholder.com/50'
72       },
73       goToDetail(heroId) {
74         window.location.href = `detailhero.html?id=${heroId}`;
75       },
76     },
77     mounted() {
78       // Load navbar
79       fetch('navbar.html')
80         .then(response => response.text())
81         .then(data => {
82           document.getElementById('navbar-container').innerHTML = data;
83         });
84       // Load superheros data
85       axios.get('https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json')
86         .then(response => {
87           this.superHeros = response.data;
88         })
89         .catch(error => {
90           console.error('Error:', error);
91         });
92     },
93   });
94   app.mount('#app');
95 </script>
96
97 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
98
99 </body>
100 </html>

```

- Le script démarre avec l'élément data qui initialise des variables.
- On commence avec le tableau de superhero qui est vide puis l'affichage des state sur false et enfin la recherche vide.
- On a ensuite l'élément computed avec la fonction de filtre qui est effectuée la recherche.
- Ensuite les methods.
- La première method gère les erreurs d'affichage des images
- La deuxième est la fonction qui va envoyer vers la page detail quand on clique sur un superhero
- Ensuite les éléments mounted
- Le premier est la fonction qui récupère la navbar pour l'afficher.
- Le deuxième est l'élément qui récupère l'API super hero et ajoute les données dans le tableau qu'on initialise au début.
- Après ça on monte l'app et on ferme le script.
- On ouvre ensuite un script bootstrap pour faire fonctionner le menu burger de la navbar.

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

- Voyont ensuite le code de la navbar :



```

1 <nav class="navbar navbar-expand-lg navbar-dark bg-primary mb-4">
2   <div class="container">
3     <a class="navbar-brand">SuperHeroes</a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
5       <span class="navbar-toggler-icon"></span>
6     </button>
7     <div class="collapse navbar-collapse" id="navbarNav">
8       <ul class="navbar-nav ms-auto">
9         <li class="nav-item">
10           <a class="nav-link" href="index.html">Liste des superhéros</a>
11         </li>
12         <li class="nav-item">
13           <a class="nav-link" href="detailhero.html?id=1">Détail d'un superhero</a>
14         </li>
15         <li class="nav-item">
16           <a class="nav-link" href="powermatcher.html">SuperPower matcher</a>
17         </li>
18       </ul>
19     </div>
20   </div>
21 </nav>

```

- Le code comment par un titre qui sera sur la droite puis le bouton burger qui ne s'activera que si la page est trop petite
- On a ensuite simplement la liste avec les 3 pages.

- Voyont ensuite le code de la page de detail des super héros :

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

```

13 <div id="navbar-container">
14 </div>
15
16 <div id="app" class="container my-4">
17   <div v-if="hero" class="row">
18     <div class="col-md-4">
19       
20     </div>
21     <div class="col-md-8">
22       <h1>{{ hero.name }}</h1>
23
24       <h3 class="mt-4">Power Stats</h3>
25       <div class="progress mb-2" v-for="(value, stat) in hero.powerstats" :key="stat">
26         <div class="progress-bar" :style="{ width: value + '%' }">
27           {{ stat }} : {{ value }}
28         </div>
29       </div>
30
31       <h3 class="mt-4">Appearance</h3>
32       <ul class="list-group">
33         <li class="list-group-item" v-for="(value, key) in hero.appearance" :key="key">
34           <strong>{{ key }} :</strong> {{ Array.isArray(value) ? value.join(' / ') : value }}
35         </li>
36       </ul>
37
38       <h3 class="mt-4">Biography</h3>
39       <ul class="list-group">
40         <li class="list-group-item" v-for="(value, key) in hero.biography" :key="key">
41           <strong>{{ key }} :</strong> {{ value }}
42         </li>
43       </ul>
44
45       <h3 class="mt-4">Work</h3>
46       <ul class="list-group">
47         <li class="list-group-item" v-for="(value, key) in hero.work" :key="key">
48           <strong>{{ key }} :</strong> {{ value }}
49         </li>
50       </ul>
51
52       <h3 class="mt-4">Connections</h3>
53       <ul class="list-group">
54         <li class="list-group-item" v-for="(value, key) in hero.connections" :key="key">
55           <strong>{{ key }} :</strong> {{ value }}
56         </li>
57       </ul>
58
59       <div class="mt-4 d-flex gap-3">
60         <button class="btn btn-primary" @click="goBack">Retour à la liste</button>
61         <button class="btn btn-primary" @click="previousHero">Hero precedent</button>
62         <button class="btn btn-primary" @click="nextHero">Hero suivant</button>
63       </div>
64     </div>
65   </div>
66 </div>

```

- Cette première moitié du code n'est que l'affichage donc le front end complet est present ici.
- On commence par récupérer la navbar
- Ensuite on ouvre la div app qui commence par afficher ll'image en large.
- On affiche ensuite le nom du hero
- Apres on a les stats du hero qui sont afficher sous forme de bar de progression comme une bar de chargement et on utilise un element for car ca raccourcit le code. Le for va donc prendre tous les éléments du json sui sont dans la directive stats pour les afficher l'un après l'autre jusqu'à ce qu'il n'y en ai plus.
- On fait ensuite pareil avec les autres éléments donc on ouvre un for qui prend tout les éléments d'un categori et les affoches a la suite
- Donc on utilise cette metode pour l'apparence, la biographie, le travail puis ses connections.





- On a ensuite un div qui va contenir trois bouton, le premier reviens a la liste des hero, le deuxième va afficher le hero suivant et le troisième affiche le hero suivant. Le code de gestion de cest bouton et fonction est afficher dans le script qui va suivre.
- Voici donc le script qui va gérer nos fonction :

```

68 <script>
69   const app = Vue.createApp({
70     data() {
71       return {
72         hero: null,
73         allHeroes: []
74       }
75     },
76
77     methods: {
78       goBack() {
79         window.location.href = 'index.html';
80       },
81
82       previousHero() {
83         const currentIndex = this.allHeroes.findIndex(h => h.id === this.hero.id);
84         if (currentIndex > 0) {
85           const prevHero = this.allHeroes[currentIndex - 1];
86           window.location.href = `detailhero.html?id=${prevHero.id}`;
87         }
88       },
89
90       nextHero() {
91         const currentIndex = this.allHeroes.findIndex(h => h.id === this.hero.id);
92         if (currentIndex < this.allHeroes.length - 1) {
93           const nextHero = this.allHeroes[currentIndex + 1];
94           window.location.href = `detailhero.html?id=${nextHero.id}`;
95         }
96       },
97
98       loadHero() {
99         const urlParams = new URLSearchParams(window.location.search);
100         const id = urlParams.get('id');
101
102         axios.get('https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json')
103           .then(response => {
104             this.allHeroes = response.data;
105             this.hero = this.allHeroes.find(h => h.id === parseInt(id));
106           })
107           .catch(error => {
108             console.error('Error:', error);
109           });
110       }
111     },

```

- Le script demare dans data a ou on a les variable d'initialisation vide.
- On a ensuite les 4 fonction qui sont donc les suivant :
- Le bouton retour qui renvoie juste sur la page d'accueil
- Le bouton hero precedent qui va simplement changer l'index de l'id dans l'URL
- Le bouton hero suivant qui fait pareil mais +1 a l'id de l'index dans l'url
- Ensuite on a l'element qui va charger les hero de l'api puis ensuite restreindre a l'hero dont l'id est sélectionné.
- Le reste du code est le suivant :

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

```

113     mounted() {
114         fetch('navbar.html')
115             .then(response => response.text())
116             .then(data => {
117                 document.getElementById('navbar-container').innerHTML = data;
118             });
119
120         this.loadHero();
121     }
122 });
123
124 app.mount('#app');
125 </script>
126
127 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
128 </body>
129 </html>
130

```



- On y charge la navbar, on y monte l'app puis on charge le script bootstrap et on ferme le tout.
- Ouvrent ensuite le code de la page qui power matcher :

```

1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
7      <script src="https://unpkg.com/axios@latest"></script>
8      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
9      <title>Power Matcher</title>
10 </head>
11 <body>
12     <div id="navbar-container"></div>
13
14     <div id="app" class="container my-4">
15         <h1>Power Matcher</h1>
16         <div class="card p-4 mb-4">
17             <div class="mb-3">
18                 <label class="form-label">Intelligence: {{ selectedPowers.intelligence }}</label>
19                 <input type="range" class="form-range" min="0" max="100"
20                     v-model="selectedPowers.intelligence">
21             </div>
22
23             <div class="mb-3">
24                 <label class="form-label">Strength: {{ selectedPowers.strength }}</label>
25                 <input type="range" class="form-range" min="0" max="100"
26                     v-model="selectedPowers.strength">
27             </div>
28
29             <div class="mb-3">
30                 <label class="form-label">Speed: {{ selectedPowers.speed }}</label>
31                 <input type="range" class="form-range" min="0" max="100"
32                     v-model="selectedPowers.speed">
33             </div>
34
35             <div class="mb-3">
36                 <label class="form-label">Tolerance (±): {{ tolerance }}</label>
37                 <input type="range" class="form-range" min="0" max="30"
38                     v-model="tolerance">
39             </div>
40
41             <button class="btn btn-primary" @click="findMatches">Find Matching Heroes</button>
42         </div>

```

- Ici le code démarre par la navbar puis ouvre la div app.
- On commence par afficher un titre puis on ouvre la div pour les input.
- La première input est un range avec le v-model pour l'intelligence.
- La deuxième est identique mais est pour la force
- Pareil pour la vitesse.
- Comparé au TP j'ai ajouté un élément de tolérance qui va permettre de faire des recherches donc les éléments ne sont pas 100% Exacte par exemple sur une tolérance sélectionnée de 5 on aura des résultats qui peuvent être jusqu'à 5 points de plus ou de moins que la recherche.
- On a ensuite le bouton pour valider la recherche.
  
- Le reste du code est l'affichage des héros qui correspondent à la recherche :

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

```

44     <div v-if="matchingHeroes.length > 0">
45         <h2>Matching Heroes ({{ matchingHeroes.length }})</h2>
46         <div class="list-group my-3">
47             <div class="list-group-item" v-for="hero in matchingHeroes"
48                 :key="hero.id"
49                 @click="goToDetail(hero.id)"
50                 style="cursor: pointer">
51                 <div class="d-flex align-items-center justify-content-between mb-2">
52                     <div class="d-flex align-items-center gap-3">
53                         
58                         <h5 class="mb-0">{{ hero.name }}</h5>
59                     </div>
60                     <small>{{ hero.id }}</small>
61                 </div>
62                 <div class="d-flex gap-2">
63                     <small v-for="(value, key) in hero.powerstats"
64                         :key="key">
65                         {{key}}: {{value}}
66                     </small>
67                 </div>
68             </div>
69         </div>
70     </div>
71 </div>

```

- Dans ce code on commence par afficher le nombre de hero qui correspondent à la recherche.
- Le reste du code d'affichage est exactement le même que celui de la page d'accueil mais le v-for change de nom puisque c'est une fonction différente.
- Voyons maintenant le script et tout le reste du code :

```



73 <script>
74   const app = Vue.createApp({
75     data() {
76       return {
77         allHeroes: [],
78         matchingHeroes: [],
79         selectedPowers: {
80           intelligence: 50,
81           strength: 50,
82           speed: 50
83         },
84         tolerance: 10
85       },
86     },
87   },
88   {
89     methods: {
90       goToDetail(heroId) {
91         window.location.href = `detailhero.html?id=${heroId}`;
92       },
93       findMatches() {
94         this.matchingHeroes = this.allHeroes.filter(hero => {
95           const intelligenceMatch = Math.abs(hero.powerstats.intelligence - this.selectedPowers.intelligence) <= this.tolerance;
96           const strengthMatch = Math.abs(hero.powerstats.strength - this.selectedPowers.strength) <= this.tolerance;
97           const speedMatch = Math.abs(hero.powerstats.speed - this.selectedPowers.speed) <= this.tolerance;
98
99           return intelligenceMatch && strengthMatch && speedMatch;
100         });
101       },
102     },
103   },
104   {
105     mounted() {
106       fetch('navbar.html')
107         .then(response => response.text())
108         .then(data => {
109           document.getElementById('navbar-container').innerHTML = data;
110         });
111
112       axios.get('https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json')
113         .then(response => {
114           this.allHeroes = response.data;
115         })
116         .catch(error => {
117           console.error('Error:', error);
118         });
119     },
120   },
121   app.mount('#app');
122 </script>
123
124 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
125 </body>
126 </html>
127

```

- On commence par initialiser les valeurs des héros et des sélecteurs ainsi que de la tolérance.
- Dans la méthode on a l'élément qui permet de rediriger vers la page détail si on clique sur un héros de l'affichage.
- Il y a ensuite la méthode qui va chercher les héros qui match avec nos sélecteurs donc en gros le code prend le héros et compare leurs stats avec les valeurs des sélecteurs tout en prenant en compte la tolérance.
- Le reste est identique aux autres pages donc on récupère la navbar puis on récupère les héros via l'API, on monte l'app et on charge le script bootstrap.

### Critères d'évaluation :

- Fonctionnalités listées avec précision.
- Explications approfondies et compréhensibles.
- Explications précises et bien argumentées pour chaque capture.
- Captures claires et pertinentes.
- Explications du code

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

### III. Difficultés Rencontrées et Solutions Apportées

#### Difficultés :

- Le cours il est ou ?
- Aucune idée de comment charger l'API pour en récupérer les données
- Le bootstrap ??? Je dois deviner ????
- Le code du powermatcher pareil je dois deviner ???

#### Solutions :

- Bah merci copilot pour avoir gérer tout le bootstrap parceque j'avais pas envie des passer la semaine a apprendre la docu et pareil pour récupérer l'api quoi

#### Critères d'évaluation :

- Difficultés bien identifiées et pertinentes.
- Solutions claires, adaptées et bien expliquées.

### IV. Commentaires et Suggestions

#### Remarques générales :

- Le TP est impossible a faire sans l'IA plus le temps passe plus j'ai l'impression que vous nous forcez a s'en servir.

#### Suggestions d'amélioration :

- Fournir un cours complet et centrer sur ce TP avec tout les detail etape par etape sans truc inutile.



#### Focus sur les Apprentissages

**Question :** qu'avez-vous appris au cours de ce TP, sur le plan technique et méthodologique ?  
J'ai appris a me servir de bootstrap et a manipuler une API en vue.js

- **Instructions supplémentaires :**
  - Le git rien a dire et l'affichage est parfait.
  - Utilisation de l'IA de manière intelligente.

#### Critères d'évaluation :

- Observations pertinentes et réflexion approfondie.
- Suggestions concrètes et réalistes.

	Nom du Module	
	BILAN TP DE PROGRAMMATION	

## V. Auto-évaluation

### Évaluation de votre travail :

- Projet terminé et fonctionnel avec des éléments en plus du TP donc une bonne note en soit.

### Objectifs pour les futurs projets :

- Aucune, je ne compte jamais me servir de vue.js.

### Critères d'évaluation :

- Auto-évaluation objective et bien argumentée.
- Objectifs pertinents et réalistes.

## VI. Code

### Lien GitHub :

- Github : [Furoshaa/Vue.JS-SuperHeroes-API](https://github.com/Furoshaa/Vue.JS-SuperHeroes-API)
- Lien hébergé : [superapi.furosha.com](https://superapi.furosha.com)

### README :

- Incluez un fichier README complet et structuré contenant :
  - Une description du projet.
  - Les instructions d'installation et d'utilisation.
  - Les prérequis et outils utilisés.

### Critères d'évaluation :

- Lien GitHub fonctionnel.
- README bien élaboré.
- Code propre, bien organisé et documenté