**Nom :** Levent

**Prénom :** Kyllian

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

# ****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

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquementUne image contenant texte, capture d’écran, Site web, Page web

Description générée automatiquementUne image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

## ****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.
  + **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 part la presentation de l’index :

Une image contenant texte, capture d’écran, menu

Description générée automatiquement

* + 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 demarre ensuitre la div app dans laquel on a le titre, le bouton qui permettra d’afficher lles stats puis la barre de recherche avec son bouton de reinitialisation.
  + On a ensuite la div qui va afficher les super heros :

Une image contenant texte, capture d’écran

Description générée automatiquement

* + 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 :

Une image contenant texte, capture d’écran

Description générée automatiquement

* + Le script demarre avec l’element data qui anitialize des variables.
  + On commence avec le tableau de superhero qui est vide puis l’affichage des state sur false et enfin lla recherche vide.
  + On a ensuite l’element computed qavec la fonction de filtre qui est enfaite la recherche.
  + Ensuite les methods.
  + La premiere method gere les erreur d’affichage des images
  + La deuxième est la fonction qui va envoyer vers la page detail quand on clique sur un superhero
  + Ensuite les elements mounted
  + Le premier est la fonction qui recupere la navbar pour l’afficher.
  + Le deuxième est l’element qui recupere l’api super hero et ajoute les données dans le tableau qu’on initialllize au début.
  + Apres ca on monte l’app et on ferme le script.
  + On ouvre ensuite un script bootstrap pour faire fonctionner lle menu burger de la navbar.
  + Voyont ensuite le code de la navbar :

Une image contenant texte, Logiciel multimédia, logiciel, capture d’écran

Description générée automatiquement

* + 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 :

Une image contenant texte, capture d’écran

Description générée automatiquement

* + 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 ellement for car ca raccourcit le code. Le for va donc prendre touts 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 :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

* + 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 retoure qui renvoie juste sur la page d’acceuil
  + 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 séléctionné.
  + Le reste du code est le suivant :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* + On y charge la navbar, on y monte l’app puis on charge le script bootstrap et on ferme le tout.
  + Ouvront ensuite le code de la page qui power matcher :

Une image contenant texte, capture d’écran

Description générée automatiquement

* + Ici le code demarre par la navbar puis ouvre la div app.
  + On commence par afficher un titre puis on ouvre la div pour les input.
  + La premiere input est un range avec le v-model pour l’intelignece.
  + La deuxième est identique mais est pour la force
  + Pareil pour la vitesse.
  + Comparé au TP j’ai ajouté un élément de tolerence qui va permettre de faire des recherche donc les éléments ne sont pas 100% Exacte par exemple sur une tolerence séléctionné de 5 on aura des resultat qui peuvent être jusqu’à 5 point 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 hero qui cirrespondent a la recherche :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* + Dans ce code on commence par afficher le nombre de hero qui correspondant a la recherche.
  + Le reste du code d’affichage est exactement le même que celui de la page d’acceuil mais le v-for change de nom puisque c’est une fonction differente.
  + Voyons maintenant le script et tout le reste du code :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

* + On commence par initialiser les valeurs des héros et des sélécteurs ainsi que de la tolerence.
  + Dans la methode on a l’element qui permet de rediriger vers la page detail si on clique sur un hero de l’affichage.
  + Il y a ensuite la methode qui va chercher les hero qui match avec nos selecteurs donc en gros le code prend le hero est compare leurs stats avec les valeurs des sélécteur tout en prenant en compte la tolerence.
  + Le reste est identique aux autres page donc on recupere la navbar puis on recupere 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

# ****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 boostrap 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’affochage est parfait.
  + Utilisation de l’IA de manière intelligente.

### ****Critères d’évaluation :****

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

# ****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 hebergé : [superapi.furosha.com](http://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