Ergonomie & développement d’une SPA animée

**UnoVinci**

|  |  |
| --- | --- |
| **Auteur 1** | Danloy Dimitri |
| **Auteur 2** | Flisch Arnaud |
| **Auteur 3** | Lefranc Axel |
| **Auteur 4** | Lefranc Sacha |
| **Auteur 5** | Schellekens Arnaud |
| **Date** | 17.12.2023 |
| **Référence** | WEB2-2023-PROJECT-GROUP-27 |
| **Version** | 1.0 |

Contents

[1 Consignes et évaluations 3](#_Toc147308156)

[1.1 Consignes générales 3](#_Toc147308157)

[1.1.1 Création des groupes sur le site du cours 3](#_Toc147308158)

[1.1.2 Création d’un groupe sur GitHub Classroom et du web repo associé 4](#_Toc147308159)

[1.1.3 Projet 4](#_Toc147308160)

[1.2 JavaScript & Node.js : consignes techniques, timing et évaluations 6](#_Toc147308161)

[1.3 Ergonomie : consignes techniques, timing et évaluations 11](#_Toc147308162)

[2 Objectif du projet 11](#_Toc147308163)

[3 Mind map du projet 12](#_Toc147308164)

[4 Persona 12](#_Toc147308165)

[5 Axiomes de Morville 13](#_Toc147308166)

[6 Planning des tâches et cas d’utilisation 13](#_Toc147308167)

[7 Besoins techniques 16](#_Toc147308168)

[7.1 Système 16](#_Toc147308169)

[7.2 Frontend 17](#_Toc147308170)

[7.3 API 17](#_Toc147308171)

[8 Choix technologiques 18](#_Toc147308172)

[8.1 Frontend 18](#_Toc147308173)

[8.2 RESTful API 18](#_Toc147308174)

[8.3 Wireframe 18](#_Toc147308175)

[9 Conception & Implémentation 19](#_Toc147308176)

[9.1 Code repositories 19](#_Toc147308177)

[9.2 Secrets éventuels pour vos API ou base de données 19](#_Toc147308178)

[9.3 Documentation de votre API 20](#_Toc147308179)

[9.4 Déploiement de vos applications 20](#_Toc147308180)

[9.5 Code réutilisé 20](#_Toc147308181)

[10 Analyse des résultats par le groupe 22](#_Toc147308182)

[10.1 Évaluation du résultat par rapport au planning des tâches et des cas d’utilisation 22](#_Toc147308183)

[10.2 Audit ergonomique de votre projet 22](#_Toc147308184)

[10.3 Difficultés techniques rencontrées 23](#_Toc147308185)

[10.4 Conseils pour appliquer cette technologie 23](#_Toc147308186)

[10.5 Quels sont les points positifs à la manière dont s’est déroulée la collaboration au sein du groupe ? 23](#_Toc147308187)

[10.6 Quels sont les points qui seraient à améliorer pour de futures collaborations ? 23](#_Toc147308188)

[11 Analyses individuelles des résultats 24](#_Toc147308189)

[12 Présentation vidéo 24](#_Toc147308190)

[13 Revues de projets par les pairs 25](#_Toc147308191)

# Consignes et évaluations

## Consignes générales

### Création des groupes sur le site du cours

Veuillez former un groupe de 5 étudiants sur le site associé au cours : <https://e-vinci.github.io/web2>. Pour ce faire, veuillez-vous authentifier en cliquant sur l’icône . Rendez-vous sur l’onglet **Projets** (<https://e-vinci.github.io/web2/project-page>). Il est recommandé que l’attribution des **groupes** se fasse par **discussions** entre les **étudiants**. Lorsque 5 étudiants ont **un intérêt commun** pour un **projet**, ils s’inscrivent au sein d’un groupe en cliquant sur l’icône .

Pour aider à la création de groupes, il est aussi possible de vous inscrire :

* **à un groupe vide**. Cela permettra à tous d’identifier les partenaires potentiels.
* **à un groupe où il y a déjà un ou plusieurs étudiants**. Dans ce cas, veuillez-vous entretenir avec ces potentiels partenaires sur le **sujet de votre projet**.

Si nécessaire, vous pouvez vous désinscrire d’un groupe où vous n’avez pas trouvé de sujet commun dans le but de rejoindre un autre groupe. Il suffit de cliquer sur l’icône .

A la date ultime de création de groupe (Séance 9 du cours de JS, 16/10 ou 17/10 selon la série), pour les étudiants toujours en recherche de partenaires, nous faciliterons (ou imposerons si nécessaire) la création des groupes, mais pas des sujets de projet.

Une fois tous les groupes de 5 étudiants remplis, il restera maximum 4 étudiants non liés à un projet. Si nécessaire un ou plusieurs groupes de 4 étudiants seront créés.

### Création d’un groupe sur GitHub Classroom et du web repo associé

Pour chaque groupe de projet, vous allez héritez d’un web repository contenant un boilerplate via GitHub classroom.

**Veuillez passer à cette étape qu’une fois votre groupe déjà finalisé sur le site du cours.**

#### Création de l’équipe associée à un projet

Veuillez identifier le membre qui créera votre équipe sur GitHub.

Ce membre accédera à l’assignement via : <https://classroom.github.com/a/zJz7A4kY>

Ce membre devra créer une équipe reprenant le numéro de projet donné sur <https://e-vinci.github.io/web2/project-page> : si le nom de projet indiqué est **Projet N°4 : …** , il créera une équipe portant le nom **group-04** puis cliquera sur **Create team**.

Ce membre devra encore cliquer par la suite sur **Accept this assignment**.

Un web repository aura été créé pour votre équipe.

#### Joindre une équipe existante

Une fois l’équipe d’un projet créée, les autres membres accéderont aussi à l’assignement via : <https://classroom.github.com/a/zJz7A4kY>.

Ces membres joindront l’équipe existante en cliquant sur **Join** au sein de la bonne équipe. Par exemple, pour les membres du **Projet N°4**, ils cliqueront sur **Join** dans l’équipe **group-04**.

Si vous le souhaitez, vous pouvez visualiser cette vidéo qui montre [comment Joindre un GitHub Classroom Group Assignment](https://youtu.be/3TIVVCkne_0).

### Projet

Vous allez créer une SPA mettant en œuvre :

* Des sujets et technologies qui vous tiennent à cœur ;
* Une RESTful API tournant sous Node.js & Express ;
* Un frontend animé ;
* Un frontend consommant votre RESTful API et éventuellement des APIs tierces ;
* Au moins une librairie JS non vue en cours pour le frontend (anime.js ou phaser.io sont autorisées) ainsi qu’une librairie non vue pour l’API.

Pour votre frontend animé, l’animation peut être 2D, 3D, sous forme de jeux ou de simples effets visuels…

Au niveau de la présentation de votre projet, veillez à :

* Prendre en compte l’expérience utilisateur dès le début
* Optimiser le choix de vos technologies en fonction de l’expérience utilisateur
* Appliquez un maximum de théorème psycho-marketing
* Respectez les règles de Usability et auditez votre projet
* Respectez le GDPR

## 

## JavaScript & Node.js : consignes techniques, timing et évaluations

| **Tâche** | **Compétences** | **Critères** | **Deadline** | **Pt** | **Consignes** |
| --- | --- | --- | --- | --- | --- |
| Objectif du projet | C6) Documenter et présenter en vidéo le développement d'une SPA |  | **Séance 9 : 16/10 ou 17/10** |  | Donnez un nom à votre projet et décrire l’objectif de votre projet au §2 de ce document ainsi que sur <https://e-vinci.github.io/web2/project-page>, complétez :   * Le nom du projet : Projet N°X : Nom de votre projet * le champs « Description ».   Discuter de votre objectif avec un enseignant et assurer vous que cet objectif soit validé avant d’aller plus loin dans votre projet. |
| Planning des tâches et cas d’utilisation | C6) |  | **Séance 12 : 24/10 ou 27/10** |  | Décrire le planning des tâches et cas d’utilisation selon les instructions données au §6.  Présenter votre planning à un enseignant, afin qu’il puisse vous aider à bien prioriser les tâches. |
| Indiquer l’URL de votre code repository | C6) |  | **Séance 12 : 24/10 ou 27/10** |  | Votre code doit être accessible par tout le monde via un web repository public qui vous sera assigné par GitHub Classroom. Cela permettra notamment aux enseignants de suivre vos avancées tout au long de votre projet. Veuillez indiquer votre URL sur <https://e-vinci.github.io/web2/project-page>.  Plus d’information aux §1.1.2 et §9.1. |
| Choix technologiques | C6) |  | **Séance 15 : 13/11 ou 14/11** |  | Compléter le §8.  Discuter de vos choix technologiques avec un enseignant. |
| Rapports individuels d’activités | C6) | Rapports de qualité *Indicateurs : formulation de qualité, analyse de qualité, respect des consignes* | **12/11**  **19/11**  **26/11**  **3/12**  **10/12**  **17/12** | **1 solo** | Des sessions individuelles de feedback sont organisées via TEAMMATES permettant à chacun de répondre à des questions dont les réponses sont confidentielles ou anonymisées au sein d’un groupe. Des e-mails seront envoyés vous invitant à compléter un formulaire hebdomadaire, à compléter pendant le WE.  À partir de la 2ème soumission, tout formulaire hebdomadaire non complété amènera à une pénalité individuelle de 0.5 point.  Si vous manquez deux soumissions, vous aurez l’obligation de montrer que vous êtes actif sur le projet sous risque d’être écarté du projet. |
| Soumission du rapport de groupe | C6) | Idem | **17/12** | **1** | Compléter le §10 ainsi que tous les paragraphes qui n’auraient pas été finalisés de ce document.  Soumettre ce document, via Moodle (un devoir sera créé) ainsi que dans le répertoire **/report** de votre repo.  Effacer toutes les consignes mises *en grisé* dans ce document avant de soumettre ce rapport sur Moodle. |
| Soumission de la vidéo | C6) | Vidéo de qualité  *Indicateurs : présentation du projet de qualité, analyse de qualité, respect des consignes* | **17/12** | **2** | Présenter votre projet selon les exigences du §12. |
| Soumission du code du frontend | C2) Création d'IHM pour SPA  inclus :  C5 : Intégrer au développement d'une SPA des technologie non vues en cours  inclus si nécessaire :  C3) Sécurisation de SPA | Qualité de l’IHM produite  *Indicateurs : esthétique, fonctionnel, codage de qualité, respect des consignes, ambitieux & original, utilisation d’une librairie pour l’IHM non vue en cours* | **17/12** | **8** | Réaliser un frontend et un backend de Qualité : Code bien structuré, UI et UX de qualité, API bien documentée (documentation des opérations de votre API, requêtes permettant de tester votre API…).  Être ambitieux et original.  Démontrer une appropriation personnelle du code (via commentaires dans le code, discussion lors des cours...).  Respecter les spécifications techniques décrites dans ce document.  Déployer votre frontend et votre backend chez un provider gratuit.  NB : votre RESTful API doit être un minimum différente des APIs fournies dans les démos du cours de JS. |
| Soumission du code du backend | C1 : Création de services web  inclus :  C5 : Intégrer au développement d'une SPA des technologie non vues en cours  inclus si nécessaire :  C3 : Sécurisation de SPA | Qualité du web service produit  *Indicateurs : fonctionnel, codage de qualité, respect des consignes, ambitieux & original, utilisation d’une librairie pour le service web non vue en cours* | **17/12** | **5** |
| Déploiement tant de votre frontend que backend | C4) Déploiement d'applications web | Déploiement de la SPA sur le cloud  *Indicateurs : fonctionnel, performances de chargement acceptables* | **17/12** | **2** |  |
| Réaliser un minimum de 5 revues sur le site web | C7) Analyser le développement de SPA faites par des pairs | Revues de projets compréhensibles & constructives  *Indicateur : présence d’un minimum de 5 revues* | **Avant examen de 1ère session** | **1**  **solo** | Via <https://e-vinci.github.io/web2/my-reviews-page>, vous devez revoir les vidéos de présentation de 5 groupes (sauf le vôtre), exécuter leurs applications, et fournir votre critique de chacun de ces projets. Vous pourrez fournir la critique d’autant de projets que vous le souhaitez. Plus d’info sur la revue de projet au §13. |
|  | **TOTAL POINTS** |  |  | **20** | Il est à noter que des membres d’un même groupe pourront être côtés différemment en fonction de leur engagement sur le projet. L’engagement d’un étudiant est visible via les rapports individuels d’activités (outil TEAMMATES)via GitHub (GitHub Project, Issues, Milestones, commits…) et lors des sessions de cours.  Les étudiants non actifs risquent d’être écarté du projet, spécialement s’ils ne soumettent pas leurs rapports individuels.  Les étudiants n’ayant pas réalisé au moins un use case significatif seront considérés inactifs.  Les étudiants n’ayant pas participés significativement au projet recevront d’office une lourde pénalité au niveau de leurs points, voire un 0/20. |

## Ergonomie : consignes techniques, timing et évaluations

Les « deadlines » données ci-dessous sont les dates où au plus tard l’avancement des tâches doivent être présentables à un enseignant pendant le cours.

| **Compétence** | **Tâches** | **Deadline** | **Points** | **Consigne** |
| --- | --- | --- | --- | --- |
| Reporting & présentation | Objectif du projet | **17/10** |  | Décrire l’objectif de votre projet au §2 de ce document.  Discuter de votre objectif avec un enseignant et assurer vous que cet objectif soit validé avant d’aller plus loin dans votre projet. |
| Conception | Définir la vision marketing | **5/11** | 6 | Décrire le Mind map du projet. Créer le persona de (s) l’utilisateur (s) ciblé (s) par le projet. Répondre aux axiomes de Morville. |
| Analyse d’applications web | Architecture UX | **5/11** | 4 | Construire les wireframes détaillés de votre application. |
|  | Analyse des résultats et rapport associé | **17/12** | 2 | Auditez votre projet et vérifiez le respect des règles GDPR. |
|  | Présentation vidéo | **17/12** | 8 | Présenter votre projet en intégrant l’expérience utilisateur. |
|  | **TOTAL** |  | **20** | Il est à noter que différents membres d’un groupe pourront être côtés différemment en fonction de leur engagement sur le projet visible lors des sessions de cours. |

# Objectif du projet

*Décrire le but du projet que vous souhaitez développer. Cette description (~15 lignes) doit notamment répondre à ces questions : quel type d’organisation (équipe sportive, bibliothèque, amis, famille…) sera prise en charge par votre application ?*

*Pourquoi cela vous tient à cœur ?*

*Vous pouvez envisager n’importe quel type d’application respectant les consignes du §1.1.*

*Veuillez résumer l’objectif de votre projet sur* [*https://e-vinci.github.io/web2/project-page*](https://e-vinci.github.io/web2/project-page) *dans le champ « Description ».*

*Veuillez aussi donner un nom à votre projet que vous compléterez sur* [*https://e-vinci.github.io/web2/project-page*](https://e-vinci.github.io/web2/project-page) *ainsi que sur la première page de ce rapport.*

*Pour décider de l’objectif de votre projet, vous pouvez vous inspirer de ce qui a été fait les années précédentes par les étudiants de web 2 en consultant la vitrine de projets offerte sur le site du cours.*

*Voici quelques autres exemples d’applications que vous pourriez réaliser :*

* *Une gestion de critiques de livres,*
* *Une gestion d’une équipe sportive (gestion des achats ou autres de l’équipe),*
* *Un jeux vidéo (multi-joueurs ou pas) dont les utilisateurs et leurs scores sont gérés par une RESTful API…*
* *Un jeu de devinette d’un mot sur base du dessin d’un des joueurs*
* *…*
* *Pour les étudiants qui n’ont toujours pas d’idées, voici ici la liste des projets proposés il y quelques années :*

**

# Mind map du projet

*Veuillez créer le Mind map de votre application en allant le plus en détails. N’oubliez pas de regrouper en catégories pertinentes comme vu en cours.*

*Veuillez soumettre votre mind map au sein du répertoire* ***/ergonomics/marketing-view*** *de votre repo.*

# Persona

*Veuillez créer au moins 2 personas de votre audience cible. Le but étant de représenter au mieux les personnes qui seront le plus engagés par votre application.*

*Veuillez soumettre vos persona au sein du répertoire* ***/ergonomics/marketing-view*** *de votre repo.*

# Axiomes de Morville

*Veuillez décrire chaque axiome de Morville de manière assez complète vis-à-vis de votre application. Ne pas donner de réponse basique mais allez plus loin dans les besoins.*

*Exemple : App de la ville de Bruxelles*

* *Valable : “parce que les gens en ont besoin et ça les aidera”*

*--> Une application aide toujours mais en quoi cela rajoute de la plus-value ?*

*--> En quoi l’application va*

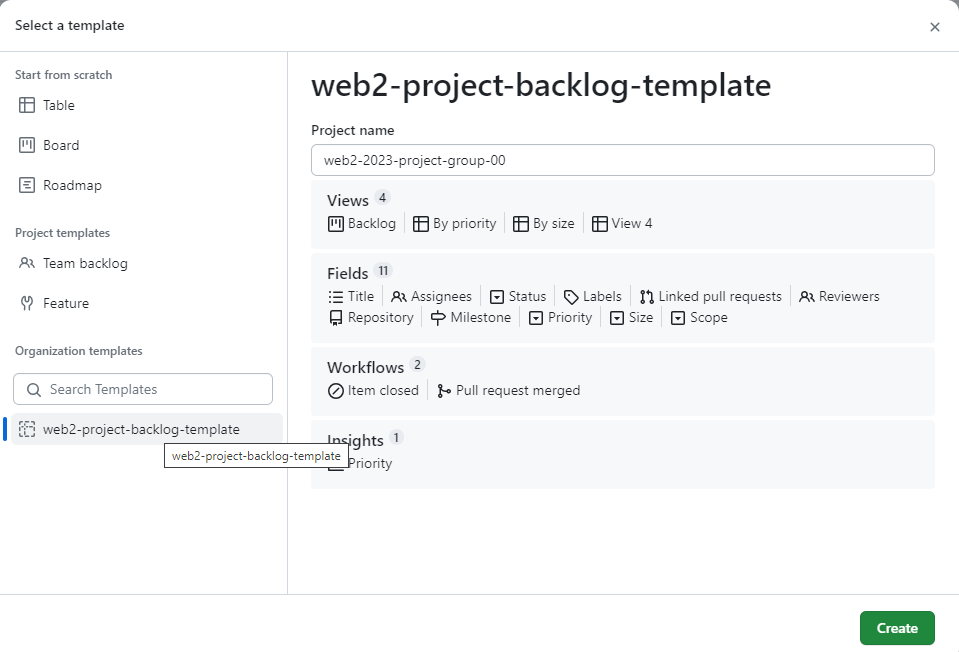
*réellement changer l’expérience utilisateur déjà existante pour ce type de projet ? Et si cela ne change pas, il faudra aussi le justifier. Ne pas recréer la roue n’est pas un défaut en soi, ça s’appelle de l’efficience.*

*Veuillez soumettre vos axiomes au sein du répertoire* ***/ergonomics/marketing-view*** *de votre repo.*

# Planning des tâches et cas d’utilisation

*Au sein du repository GitHub de votre projet, vous devez créer un* ***Project*** *pour planifier les tâches, allouer les responsables, documenter vos avancées, visualiser vos* ***Milestones****...*

*Veuillez regarder la vidéo qui suit pour mettre en place votre GitHub Project. Cette année, si vous souhaitez accélérer la création de votre GitHub Project, vous pouvez sélectionner comme organization templates :* ***web2-project-backlog-template*** *au lieu de* ***Team backlog*** *comme montré dans la vidéo.*

**

*Si vous préférez faire exactement comme dans la vidéo et créer manuellement toutes les tâches, vous utiliserez le template :* ***Team backlog****.*

*Voici la vidéo expliquant la mise en place de votre GithHub Project :* [*Partie1 : Créer et configurer un GitHub Project*](https://youtu.be/y2e_ItjAZ2I)*.*

*Voici le workflow que nous souhaitons vous voir appliquer sur GitHub Project :*

* *Veuillez commencer votre projet en identifiant toutes les tâches principales à réaliser sur votre projet, principalement en soignant l'identification des use cases. Pour ce faire, vous pouvez visualiser la vidéo :* [*Partie 2 : planifier des tâches via GitHub Project*](https://youtu.be/hz0P-PfYqgo)*.  
  NB : si vous avez accéléré la création de votre GitHub Project en sélectionnant le template* ***web2-project-backlog-template****, reste intéressante pour voir comment créer des drafts issues, convertir des draft issues en issues, ajouter des Labels aux issues…*
* *Chaque****tâche****doit être couverte par une****draft Issue****au sein de GitHub que vous devrez convertir plus tard en****Issue****. Veuillez allouer une****Priority****à chaque****Issue****(ou****draft Issue****), ainsi qu'une****Size****.*
* *Lors de l'identification des tâches, les****Issues****associées se trouvent dans la colonne****Backlog****.*
* *Un****cas d'utilisation****doit être couvert par au moins une****Issue****avec un label nommé****enhancement****.*
* *Chaque****Issue****doit être associée à au moins un****Assignee****.*
* *S'il y a plusieurs****Assignees****associés à une****Issue****, celle-ci devra être découpée en suffisamment de tâches pour qu'il y ait un seul****Assignee****par tâche. Pour la découpe d’une Issue en tâches, la création de Label, et la gestion de Milestones, vous pouvez visualiser la vidéo :* [*Partie 3 : gestion approfondie des tâches via GitHub Project*](https://youtu.be/F8lPoL8qiZA)*.*
* *Dans un premier temps, une****Issue****associée à plusieurs****Assignees****peut simplement identifier les tâches associées au sein d'une****task list****. Plus tard, ces tâches devront être converties en nouvelles****Issues****associées à un seul****Assignee****.*
* *Lorsque vous travailler sur une****Issue****:*
  + *elle doit se trouver dans la colonne****In progress****ou****In review****si vous pensez avoir terminé mais que vous attendez le feedback d'un membre de votre projet.*
  + *pour chaque avancée significative sur une****Issue****, vous devez indiquer un commentaire via****Comment****résumant le travail effectué.*
* *Lorsque vous considérez qu'une****Issue****est terminée, faites la passer dans la colonne****Done****, indiquez un message via****Comment****résumant le travail effectué et cliquez sur****Close issue****.*
* *Pour facilement voir le pourcentage de progrès dans la fermeture des****Issues****qui vous sont associées, vous devez créer une****Milestone****par membre de projet et associer cette****Milestone****à toutes les****Issues****où le membre de projet est le seul****Assignee****.*

*Pour visualiser les tâches d’un seul membre de projet, il est possible de faire, via GitHub Project, autant de* ***view*** *qu’il y a de membres de projet, ou une seule vue pour chaque utilisateur connecté. Pour obtenir une vue pour un membre de projet, il suffit de cliquer sur la* ***view****, puis* ***filter****, puis d’écrire :* ***assignee:*** *et d’indiquer le username d’un membre du projet. Il est aussi possible de créer une vue des tâches de l’utilisateur connecté, qui pourra être réutilisée par tous les membres du groupe, en indiquant :* ***assignee:@me*** *pour le* ***filter****.*

*Pour votre développement, n’hésitez pas à être ambitieux, tout en restant réaliste. Comment faire ? Nous vous recommandons de spécifier les cas d’utilisations qui* ***doivent*** *être implémentés par une priorité «****Haute****». Ceux qui* ***pourraient*** *être implémentés, mais qui ne sont donc pas indispensables à l’application de base, vous devriez les catégoriser selon une priorité « Moyenne » ou « Basse ».*

*Afin de vous aider dans la création de votre planning au sein de GitHub Project, nous vous proposons une liste de base d’****Issues*** *à prendre en compte**(c’est les draft issues qui sont offertes dans le* ***web2-project-backlog-template****)**:*

* *Create marketing vision (documentation)*
* *Create wireframes (documentation)*
* *Identify list of use cases (documentation)*
* *UC1 : ... (enhancement)*
  + *UC1-frontend : ... (enhancement)*
  + *UC2-api : ... (enhancement)*
    - *Create API HTTP requests (tests)*
* *UC2 : ... (enhancement)*
* *UC3 : ... (enhancement)*
* *UC...*
* *Document API (documentation)*
* *Deploy frontend (deployment)*
* *Deploy API (deployment)*
* *Create Project Report (documentation)*
* *Create Video (documentation)*

*Notons que les rapports individuels d’activités que vous devrez compléter chaque semaine, suite à invitation par e-mail, ne sont pas à reprendre dans votre planning de tâches.*

***Attention que chaque membre de projet doit avoir au moins un use case significatif pour être considéré actif.***

*Veuillez indiquer l'URL vers votre* ***Project*** *public sous GitHub ici :*

* URL vers votre GitHub Project public : <https://github.com/orgs/e-vinci/projects/113>

# Besoins techniques

*La spécification technique des besoins vous est donnée*

*N’hésitez pas à la compléter s’il manque qqch d’important.*

## Système

TRS01 : Vous devez développer une Single Page Application (SPA) à l’aide de JS et Node.js.

TRS02 : Votre RESTful API doit être indépendant de votre frontend ; vous aurez donc deux applications distinctes, une pour le frontend et l’autre pour la RESTful API.

TRS03 : Vous devez utiliser GitHub sur votre projet afin de gérer le développement de chacun des membres d’une équipe.

*Nous vous recommandons d’appliquer un workflow vu dans votre cours de DevOps : pour chaque cas d’utilisation / feature que vous développez, essayez de créer une branche correspondante. De plus, il serait intéressant que vous mettiez en œuvre des revues de code au sein de votre projet via des Pull Request sur Github.*

## Frontend

TRF01 : Votre frontend doit utiliser Webpack en tant que package bundler.

TRF02 : Le frontend, développé en HTML / CSS (bootstrap ou autre) / JavaScript, doit consommer au moins une de vos RESTful API.

*Votre frontend peut consommer des API externes, des APIs que vous n’avez pas développées vous-même (e.g. API de youtube, de google maps…)*

TRF03 : Votre frontend doit mettre en œuvre une librairie JS externe, ou l’API Canvas, afin de réaliser une animation.

*L’animation peut prendre la forme d’une animation 2D, 3D ou d’un jeu vidéo.   
Attention à ne pas juste offrir une minuscule animation à l’aide d’une librairie ne demandant aucun code JS, comme certaines librairies mettant tout en œuvre à l’aide de CSS.*

TRF04 : Votre frontend doit mettre en œuvre au minimum une librairie JS non vue en cours.

*Anime.js est autorisé pour votre animation.*

TRF05 : Votre frontend doit respecter les droits d’auteurs, que ça soit pour les éventuels sons, images, vidéos, librairies et morceaux de codes utilisés. Cela est de votre responsabilité et non pas de celle de vos enseignants.

TRF06 : Vous devez déployer votre frontend sur GitHub Pages ou d’autres providers gratuits supportant votre application.

## API

TRA01 : Vous devez créer une RESTful API afin d’offrir des opérations sur des ressources utiles à votre projet.

*La RESTful API ne peut pas être uniquement un « copier/coller » de ressources offertes dans le cours (notamment les ressources users et auths). Vous pouvez utiliser les ressources offertes dans le cours, mais vous devez y apporter des ajouts significatifs.*

TRA02 : Votre RESTful API doit mettre en œuvre au minimum un package non vu en cours.

TRA03 : Vous devez documenter les opérations de votre API conformément aux conventions REST.   
*Vous pouvez documenter votre API soit sous forme de tableau, comme vu dans le cours, soit à l’aide d’outils tel que Swagger.*

TRA04 : Les tests de votre API, les requêtes HTTP, doivent être données au sein de votre projet. Pour chaque opération de votre API, il doit exister au minimum une requête HTTP associée.

TRA05 : Votre API doit respecter les droits d’auteurs, que ça soit pour les éventuelles librairies utilisées, les morceaux de code, les sons, images, vidéos… Cela est de votre responsabilité et non pas de celle de vos enseignants.

TRA06 : Vous devez déployer votre backend sur Azure ou d’autres providers gratuits supportant votre application.

# Choix technologiques

*Principalement pour la ou les librairies JS non vue en cours que vous avez ou allez appliquer, veuillez décrire la technologie choisie pour répondre à un ou plusieurs cas d’utilisation identifié(s) au §6. Ce paragraphe est à rédiger sur environ 20 lignes. Veuillez décrire les liens vers le ou les sites à utiliser (ou utilisés), les ressources principales utiles au développement.*

## Frontend

*Pour le frontend, voici des exemples de librairies open source qui pourraient être choisies, en fonction de votre objectif :*

* *pour la 2D :* [*https://animejs.com*](https://animejs.com)
* *pour la 3D :* [*https://threejs.org*](https://threejs.org)
* *pour les jeux :* [*https://phaser.io*](https://phaser.io)

## RESTful API

*Pour votre Restful API, vous devez découvrir un package non vu en cours pour réaliser l’une ou l’autre des fonctionnalités. Veuillez indiquer ici le ou les packages choisi(s) avec les liens vers les sites utilisés.*

## Wireframe

*Veuillez créer sur PowerPoint, des wireframes de votre application. Il en faut au minimum 3.*

*Exemple d’un jeu : il y aura au minimum, l’accueil inscription, le jeu, une page de résultat, page contact développeur, etc.*

*Veuillez soumettre vos wireframes au sein du répertoire* ***/ergonomics/wireframes*** *de votre repo.*

# Conception & Implémentation

## Code repositories

*Veuillez indiquer l’URL de votre web repository public, générée par GitHub Classroom, sur* *https://e-vinci.github.io/web2/project-page. Pour ce faire, veuillez au moins modifier le champs « Repo frontend » (et éventuellement « Repo backend »). L’URL du web repository de votre groupe devrait correspondre à qqch du style « https://github.com/e-vinci/web2-2022-project-group-04 » si vous êtes membres du Projet N°4.*

*Veuillez aussi indiquer cette URL ci-dessous.*

* URL pour le web repository public associé à votre projet : <https://github.com/e-vinci/web2-2023-project-web2-2023-project-group-27>

*Il est important que pour explorer vos projets, on puisse facilement installer votre frontend et backend. Veuillez veiller à ce que votre application s’installe et s’exécute simplement localement via ces actions :*

* *git clone de votre backend*
* *npm install et npm au niveau du backend*
* *git clone du frontend*
* *npm install et npm start au niveau du frontend*

## Secrets éventuels pour vos API ou base de données

*Si vous utilisez une base de données ou des API nécessitant des secrets, il est important de ne pas rendre public vos secrets. Dans ce cas :*

*- Votre application doit être sur le cloud pour que les autres étudiants puissent la revoir ; les étudiants ne pourront donc pas exécuter l'API localement. Veuillez clairement indiquer dans le README de votre projet si l’application ne peut pas être exécutée localement sans les secrets et veuillez indiquer l'URL tant de votre frontend que de votre API au sein de ce README.*

*- Vous devez mettre à disposition tous ces secrets (fichiers de configuration) à disposition de vos enseignants lors de la soumission de ce rapport. Le devoir Moodle vous permettra d’inclure les fichiers nécessaires.*

*- Pour la création de votre éventuelle DB, si elle ne se fait pas automatiquement lors du démarrage de votre API, vous devez offrir un script et le mettre au sein de votre projet. Dans ce cas, la procédure pour créer la DB doit être documenté au sein du README de votre projet.*

## Documentation de votre API

*Veuillez documenter les opérations de votre API, soit à l'aide de tableaux à donner ci-dessous, comme ceux vu dans le cours, soit en référençant la documentation qui aurait été générée à l'aide d'outils (Swagger par exemple). Votre fichier README de votre projet doit indiquer l'endroit où se situe la documentation de votre API.*

* Tableaux représentant les opérations de votre API ou lien vers la documentation de votre API :

*Veuillez aussi documenter les tests de votre API : les requêtes http doivent être données au sein de votre projet. REST Client devrait être utilisé, mais si vous préférez un autre client léger, vous devez vous mettre d’accord au sein de votre équipe de projet. Veuillez indiquer où se trouvent les requêtes HTTP si ça n’est pas dans le répertoire* ***/api/REST Client****.*

*Rappel : il ne peut pas y avoir d'opération de votre API sans au minimum une requête HTTP associée.*

* Requêtes HTTP se trouvent dans : **/api/REST Client**

## Déploiement de vos applications

*Veuillez indiquer l’URL de votre frontend déployés sur* *https://e-vinci.github.io/web2/project-page. Pour ce faire, veuillez modifier le champs « URL du site ».*

*Veuillez aussi indiquer ci-dessous deux URLs, comme par exemple https://e-vinci.github.io/wowapp :*

* URL de votre frontend déployé : <https://steedlan.github.io>
* URL de votre RESTful API déployée : <https://unovinci.webpubsubs.azure.com>

(En socket.io)

## Code réutilisé

*Il est important que vous citiez les parties de code que vous avez réutilisées, du code issu d’un tiers, au sein de votre code source. Pour ce faire, dans votre code source, utilisez des commentaires, en complétant au minimum les informations associées à l’auteur et à l’endroit où le code est disponible (URL). Voici le format que pourrait prendre votre commentaire :*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*    Title: <title of program/source code>  
\*    Author: <author(s) names>  
\*    Date: <date>  
\*    Code version: <code version>  
\*    Availability: <where it's located, URL>  
\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*Veuillez résumer tous les codes sources utilisés dans votre code, au sein de ce tableau :*

| **Chemin du fichier où se trouve le code réutilisé** | **Auteur du code source réutilisé** | **URL où le code réutilisé est disponible** | **Raison de la réutilisation du code** |
| --- | --- | --- | --- |
| *e.g. /webApp/src/index.js* | *Dogan Erisen* | *https://github.com/Azure-Samples/active-directory-b2c-javascript-msal-singlepageapp* | *Code pour recevoir un access token via Azure AD* |
| Logo | Dall-E |  | Besoin d’un logo |
| ./frontend/src/scripts/chat.js | Chat-GPT |  | Affichage des 5 derniers messages et disparition après 10 secondes |

# Analyse des résultats par le groupe

## Évaluation du résultat par rapport au planning des tâches et des cas d’utilisation

*Veuillez décrire si vous avez pu atteindre vos objectifs fonctionnels et autres tâches. Si vous avez dévié des objectifs fonctionnels identifiés initialement, veuillez décrire les raisons de ce changement.*

*Ce paragraphe est à rédiger sur environ 15 lignes.*

D’un point de vue fonctionnel la plupart de ce qui était prévu a été ajouté, à l’exception de petits changements mineur comme le contre-uno dû à des bugs qui n’ont pas pu être corrigé à temps, les statistiques sur les comptes par manque de temps. D’autres fonctionnalités comme pouvoir jouer deux cartes en même temps si elles sont identiques n’ont pas été fait non plus car au fur et à mesure du développement du jeu, nous ne trouvions pas nécessaire de rajouter cette fonctionnalité en sachant qu’une partie avait déjà le rythme parfait. Pareil pour les accumulations où on avait décidé de se rapprocher des règles officielles (qui ne les incluent pas). Enfin, la personnalisation des profils n’a pas été fait par manque de temps. Le manque de temps lié à la feature des comptes était dû au fait qu’on s’est rendu compte au moment du déploiement de l’API qu’un seul port était disponible, réservé au websocket, et au dernier moment on a décidé de quand même rajouter cette fonctionnalité en faisant une courte requête websocket pour les comptes. Nous savons qu’en basant tout notre projet sur cette technologie nous nous éloignons un peu de la matière vue au cours, mais pour un jeu en ligne en temps réel nous avions estimé nécessaire d’avoir des websockets.

## Audit ergonomique de votre projet

*Veuillez utiliser les outils vus en cours pour analyser l’ergonomie de votre projet et si les règles de GDPR sont respectées. Créer ensuite ci-dessous un rapport à rédiger d’environ 10 lignes.*

Une bannière reprenant les conditions d’utilisation du site doit être accepté au moment de l’inscription du site, car avant cela aucune donnée n’est enregistrée. D’un point de vue ergonomique l’interface a été simplifiée à son maximum avec un unique bouton « jouer », en un clic il est donc possible de pouvoir rejoindre une partie, dans le cas où aucun pseudo a été spécifié un pseudo aléatoire est préalablement choisi. La logique de jeu a été simplifié de la manière la plus simple possible, quand le joueur n’a aucune carte jouable il n’est pas nécessaire pour le joueur de se rendre compte qu’il n’a pas de cartes, on fait donc clignoter la pioche pour indiquer au joueur qu’il doit piocher. Nous n’avions pas estimé nécessaire de faire clignoter les cartes jouables au joueur afin de lui donner un minimum de sentiment de liberté en visitant notre site.

## Difficultés techniques rencontrées

*Indiquez ici les difficultés techniques que vous auriez rencontrées au cours de votre projet.*

*Ce paragraphe est à rédiger sur environ 15 lignes.*

Le plus gros défi aura été d’héberger notre api utilisant un serveur websocket, étant donné que l’api a besoin de deux ports (un pour l’api et l’autre pour socket.io) la totalité des hébergeurs gratuit ne proposent pas plus d’un port. La première solution « maison » était de le faire héberger sur le VPS d’un ami et ça avait fonctionné, jusqu’à ce qu’on déploie le frontend et que l’on se rendit compte que puisque github.io est en HTTPS, l’api devait aussi être en HTTPS, il fallait donc des certificats et c’était trop compliqué. Donc nous avions désespéré jusqu’à ce qu’on se rendît compte qu’Azure venait tout récemment de rendre son service de proxy socket.io gratuit, nous avons donc tenté le tout pour le tout et cela a fonctionné, le service de base tourne donc toujours sur le VPS d’un ami, mais nous avons un proxy pour les sockets en HTTPS permettant donc de faire profiter le plus de monde.

## Conseils pour appliquer cette technologie

*Veuillez résumer tout ce qui serait intéressant pour quelqu’un qui souhaiterait appliquer le même genre de technologie, sur base de ces questions :*   
*Qu’est-ce que vous auriez aimé savoir avant de démarrer ?*

*Des conseils pour éviter certains problèmes rencontrés ?*   
*Des liens vers les sites les plus intéressants ?*

*Etc.*

*Ce paragraphe est à rédiger sur environ 15 lignes.*

Le plus compliqué est le déploiement de websockets qui a dû demander pas mal de semaines d’essais infructueux, nous avons déjà expliqué ci-dessus l’aventure qu’il a fallu pour déployer l’API, mais créer un projet avec websockets demande pas mal de détermination et de motivation vis-à-vis du projet. J’aurais probablement aimé plus d’explications liés à l’usage de l’hébergement Azure car l’interface prête à confusion et m’a amené à découvrir le milieu du déploiement qui semble faire partie d’un métier à part entière.

## Quels sont les points positifs à la manière dont s’est déroulée la collaboration au sein du groupe ?

*Veuillez donner des arguments justifiant votre réponse. Cette réponse est à rédiger sur environ 10 lignes.*

Chacun de nous savait quoi rajouter tout le temps, il n’y a pas eu un moment où on sentait avoir fait le tour du projet et à chaque fois on s’imaginait toujours grimper plus haut en fonctionnalités jusqu’à doucement atteindre la fin du projet et ralentir jusqu’à atteindre une version fonctionnelle.

## Quels sont les points qui seraient à améliorer pour de futures collaborations ?

*Veuillez donner des arguments justifiant votre réponse. Cette réponse est à rédiger sur environ 10 lignes.*

Si nous avions eu plus de temps nous aurions pu mieux schématiser le projet avant de le développer, mais l’ambition du site et la deadline nous a poussé à le commencer plus tôt. si nous aurions aussi pu prendre le temps de documenter l’api mais le développement du projet prenait déjà beaucoup en dehors des horaires du cours.

# Analyses individuelles des résultats

*Les analyses individuelles sont à réaliser via les sessions de feedback qui seront à soumettre individuellement à la fin de chaque semaine de cours, pendant le WE, via l’outil TEAMMATES (*[*https://teammatesv4.appspot.com*](https://teammatesv4.appspot.com)*).*

*Vous recevrez des e-mails vous invitant à compléter votre feedback hebdomadaire sur le projet. Attention aux pénalités si vous ne complétez pas votre feedback.*

*NB : Ce §11 peut être entièrement effacé du rapport que vous soumettrez sur Moodle.*

# Présentation vidéo

*Voici les exigences associées à votre présentation vidéo :*

* *Elle doit viser une durée de 5 minutes, et ne peut pas dépasser 10 minutes.*
* *Elle doit être visible sous youtube par n’importe qui possédant son URL. Sa visibilité doit donc être en ‘’Unlisted" ou "Public", mais pas « Private » !*
* *Elle se basera principalement :*
  + *sur la présentation de votre application web : exécution, en live, de votre API et du frontend ;*
  + *la présentation de l’expérience utilisateur ;*
* *Elle pourra aussi se baser sur d’autre(s) point(s) éventuel(s) vous permettant de vendre au mieux votre travail.*
* *Votre présentation devra être bien visible et audible.*
* *Il serait bien que celle-ci soit bien structurée, notamment via l’affichage éventuel de titres.*
* *Vous pouvez ajouter une bande son, et des images, mais seulement si celles-ci respectent les droits d’auteurs.*
* *Vous veillerez à ce que, dans la description de votre vidéo, vous fournissiez les liens vers le web repository associé à votre projet ainsi que l’URL vers le frontend déployé.  
  L’idée est que si un projet intéresse des visiteurs de votre repo, ils aient accès à tout ce qui est nécessaire pour bien le comprendre, voire pour le réutiliser, sous réserve de bien citer vos ressources.*

*En plus de ces exigences, la présentation vidéo a pour but de vendre un projet qui vous tient à cœur. Il est possible que si le résultat soit accrocheur, les enseignants demandent votre autorisation afin de rendre votre projet public. Avec votre autorisation, nous pourrions notamment présenter votre projet lors de salons d’étudiants, soit via votre vidéo, ou directement en exécutant votre application déployée sur le cloud.*

*Le site* [*https://e-vinci.github.io/web2/*](https://e-vinci.github.io/web2/) *présentera les projets qui auront été sélectionnés pour être publics. De plus, vous pourriez utiliser vos projets comme portfolio pour vos futurs employeurs.*

*Pour créer votre vidéo, avant de la mettre sous youtube, veillez à ce que celle-ci soit bien visible et bien audible. Nous vous recommandons :*

* *de la réaliser au format 1920 X 1080*
* *d’utiliser un logiciel gratuit pour la réaliser. Voici ceux que nous pouvons vous conseiller :*
  + [*https://obsproject.com/*](https://obsproject.com/)*: logiciel open source demandant un temps d’adaptation, mais permettant de faire énormément*
  + [*https://www.loom.com/*](https://www.loom.com/)*: logiciel pouvant être utilisé gratuitement sous réserve d’accepter un logo. Très facile d’utilisation.*
  + [*https://screencast-o-matic.com/*](https://screencast-o-matic.com/)*: logiciel pouvant être utilisé gratuitement sous réserve d’accepter un logo. Très facile d’utilisation.*

*Veuillez indiquer le lien vers la vidéo youtube que vous avez créée sur le site* [*https://e-vinci.github.io/web2/project-page*](https://e-vinci.github.io/web2/project-page)*. Pour ce faire, veuillez modifier le champ « Vidéo de présentation ».*

*De plus, veuillez indiquer ci-dessous ce lien :*

Lien vers la vidéo youtube : <https://www.youtube.com/watch?v=y6ryb_5VL3Y>

# Revues de projets par les pairs

*Une fois la deadline de soumission des projets atteinte, la saison de revues des projets sera ouverte !*

*Le but ?*

* *Participer à la revue bienveillante, aux critiques constructives, de sites web faits par vos pairs ;*
* *Identifier les projets qui plaisent, notamment afin d’améliorer leur visibilité !*

*Pour vos revues de projets, voici les règles :*

* *5 revues sont attribuées automatiquement à chaque membre d’un projet, ainsi qu’un coup de cœur lorsque vous accédez à* [*https://e-vinci.github.io/web2/my-reviews-page*](https://e-vinci.github.io/web2/my-reviews-page)*.*
* *Toute revue doit comprendre au moins 1 point fort identifié et 1 point d’amélioration ; vous pouvez baser ces points suite au visionnage de la vidéo uniquement, mais nous vous recommandons de le faire après avoir exécuté l'application associée au projet ; vous pouvez bien sûr aussi accéder au code de l’application pour votre revue.*
* *Libre à vous de vous attacher au design, au gameplay, à l'ergonomie, au code, à la vidéo ou tout autre aspect dans votre revue. Chaque revue sera affichée – ainsi que votre nom – dans le détails d’une revue. Soyez donc bienveillants et constructifs dans votre analyse critique ; )*
* *Les résultats des revues ne sont accessibles qu’à vos pairs ! Votre analyse critique n’est donc pas publique, seuls les membres d’un projet de Web2 y ont accès, uniquement pour les membres ayant réalisé au moins 5 revues. Les projets sont listés par nombre de coups de cœur reçus, par nombre de revues faites, puis simplement par ordre alphabétique.*
* *Une revue soumise peut être modifiée. Nous vous conseillons, avant de donner un coup de cœur, d’avoir jeté un œil à tous les projets qui vous intéressent ; un coup de cœur donné ne peut pas être retiré ; )  
  Par contre, vous pouvez faire vos revues tranquillement, puis mettre à jour celles-ci plus tard pour attribuer vos coups de cœur.*
* *Après avoir effectué vos 5 revues attribuées, vous obtenez :*
  + *un deuxième coup de cœur à offrir,*
  + *l’accès aux résultats généraux des revues et aux détails de chacune des revues.*
  + *Le pouvoir de revoir n’importe quel projet non encore revu.*

*Comme l’hébergement gratuit d’API devient de plus en plus compliqué, nous souhaitons garantir que les 3 projets les plus aimés puissent bénéficier d’un hosting offert et géré par la HE Vinci.*

*NB : Ce §13 peut être entièrement effacé du rapport que vous soumettrez sur Moodle.*