

REMERCIEMENT



Je tiens à adresser mes remerciements au centre de formation DORANCO, pour la qualité de la formation et le soutien de l'équipe administrative. Je saisirai cette occasion pour adresser mes profonds remerciements à Mr Yassine AABIDOUCHE pour tous ses conseils. Je tiens également à remercier les formateurs qui ont tous été à la hauteur, et on m'ont tous appris à leurs manière de nouvelles façons de coder. Je voudrais également remercier mes camarades de promotion.



1.0)sommaire

1.1)context

- cadre de l'application

1.2)expression du besoin

- problématique du projet

2.0)Etude du marché

2.1)public cible

- Persona

2.3)Benchmark concurrentiel

- analyse des différents concurrents

- analyse de leurs point fort / faible

3.0)Projection du projet

3.1)maquettage

- Wireframe

3.2)organisation du projet

- diagramme de gantt

3.3)Méthode agile

- KanBan

4.0)Diagramme

4.1)diagramme de classe

- utilisation de diagramme de classe

4.2)diagramme de séquence

- utilisation de diagramme de séquence

4.3)diagramme de cas d'utilisation

- utilisation de diagramme de cas d'utilisation

4.4)MCD/MLD

- MCD

- MLD

5.0)environnement

5.1.1)react

- installation Réact
- librairie des Librairies
- utilisation de composant
- utilisation d'architecture web /public

5.1.2)THREE js

- moyen de visualiser les donnée
- permet de charger ces données
- limiter mais personnalisable

5.1.3)arborescence de l'application web

- utilisateur
- admin

5.2.1)Spring

- installation de spring
- les dépendances

5.2.2)mcv

- model
- vue
- contrôleur

5.2.3)api MongoDb

- utilisation des rôles
- Cors et credentials
- 5.2.4)api SQL
- confirm d'utilisateur email
- utilisation des relations pour publication

6.0)Serveur Minecraft

6.1)Utilisation du Server Minecraft

- installation du server
- récupérations des données
- requêtes https

7.0)CI/CD

7.1)Docker

- docker compose

7.2) Jenkins

-vérification des test avec jenkins tests unitaires

1.1) Context

Tout d'abord je tiens à dire que ce projet répond à une problématique personnelle.

Pour mieux comprendre ce qui m'as amener à ce projet:

J'étais en train d' apprendre l'utilisation d'un moteur de jeux (BluePrint/C++), jusqu'à ce que je voie un cube non-texturé. A partir de là je me suis dit:
-Si je met les textures de Minecraft sur ce cube et que je change les coordonnées des cubes, je peux construire n'importe quel bâtiment de Minecraft.

Donc j'ai créée un moyen de récolter ces coordonnées, via Spigot, mais comme je me suis dit : c'est dommage d'être le seul à en profiter. Donc j'ai créé une version web, qui permet de récupérer ces coordonnées et de les générer sous forme de model 3d. Et, c'est ainsi que je récupère ce que j'ai construit sur Minecraft.

Et plus tard, je voudrais que sur cette application web l'on puisse directement charger un fichier (.schematics /.NTB), et télécharger un model 3d(.OBJ ou autre). Cela permettra à tous de pouvoir générer un modèle , et non uniquement à ceux qui sont passés sur mon serveur, et augmenter l'accessibilité du site.

Bien que le .NTB et le .schematics ne soient pas des formats de model 3d ils sont, très souvent utilisés pour copier, coller ,publier etc, car ils sont utilisable uniquement dans le jeux.

Définition du Projet

-Le principal objectif de ce projet est de créer une plateforme web dédiée à la communauté Minecraft, où les utilisateurs peuvent partager, visualiser et organiser des modèles 3D. Le site vise à offrir un moyen plus interactif et amusant pour les utilisateurs de montrer et de partager leurs créations.

Pour ce projet, j'ai voulue répondre à cette problématique:

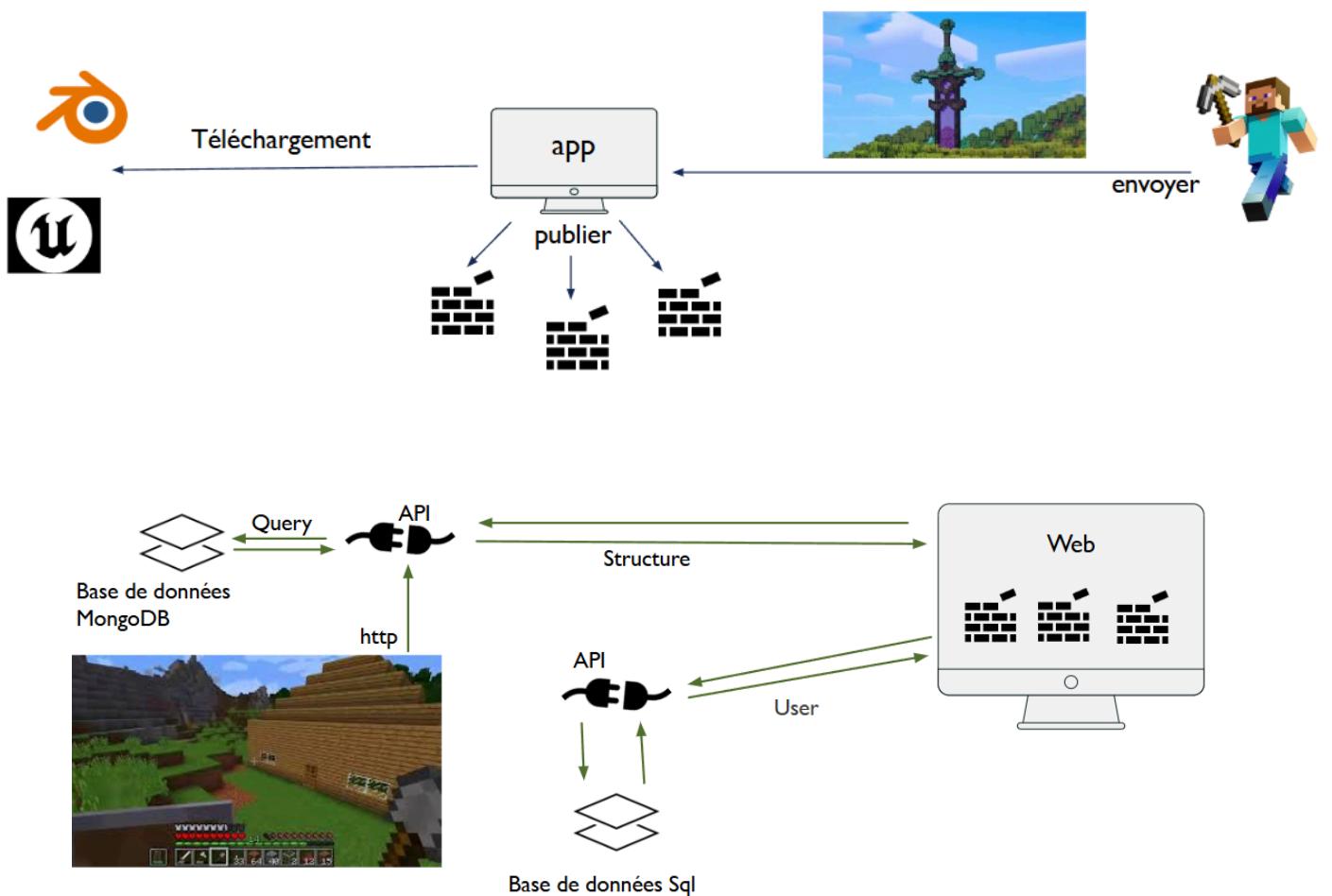
Comment faire pour récupérer facilement et rapidement, ce que j'ai construit sur minecraft?

La partie publication, c'est pour les utilisateurs, car la plupart des joueurs du jeu, aime juste "montrer leurs talents". Je me suis donc dit que cette fonctionnalité peut plaire aux utilisateurs.

De plus, là plupart des modèles 3d en général sont payants, dont également certaines constructions Minecraft.

Je pourrais donc, moi aussi, implémenter par la suite, la possibilité aux utilisateurs de vendre des modèles, et à d'autres de les acheter.

Voici deux schémas qui résume les fonctionnalités de base de mon application.



2.1) Persona

il y à 3 profiles types qui voudront venir sur cette application:

-Dysineurs

Le designer est là pour trouver l'inspiration pour ces œuvres. Ils peuvent venir chercher et télécharger des créations inspirantes, réalisées par des joueurs.

-Joueurs

Le joueur veut construire des bâtiments toujours plus impressionnant, et peut partager ces créations au monde entier.

-Imprimeurs

Il existe des sociétés qui imprime en 3d de models minecraft, ils téléchargent une création afin de l'avoir dans le monde réel.

activité	nom	âge	personnalité	objectif
Designer	Célia	22	visuellement inspiré en ce moment par la nostalgie, doit réaliser une oeuvre sur blender	cherche de l'inspiration depuis l'imagination infinie de la communauté Minecraft
Joueur	Timoté	13	Joue aux jeux vidéo, timide dans le monde réel	Veut montrer ses talents en build
Imprimeur	Mathieu	36	employé rigoureux et à une imprimante 3D	cherche une construction jolie et simple à imprimer

On peut déjà déduire avec ces 3 profils différents, 3 comportements ou façon différentes d'utiliser l'application:

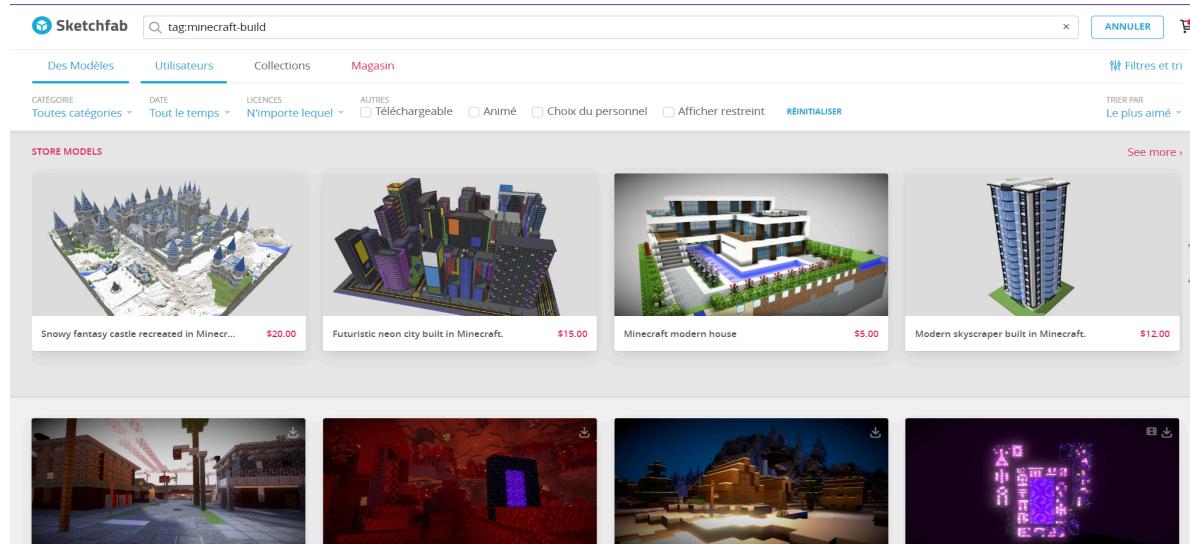
-On à la Designeuse qui va explorer toutes les build publiques, et télécharger si besoin.

-Le joueur qui va collectionner les build privés, en publiant les meilleurs d'entre elles. il regardera les build publiques, pour s'inspirer et voir la concurrence.

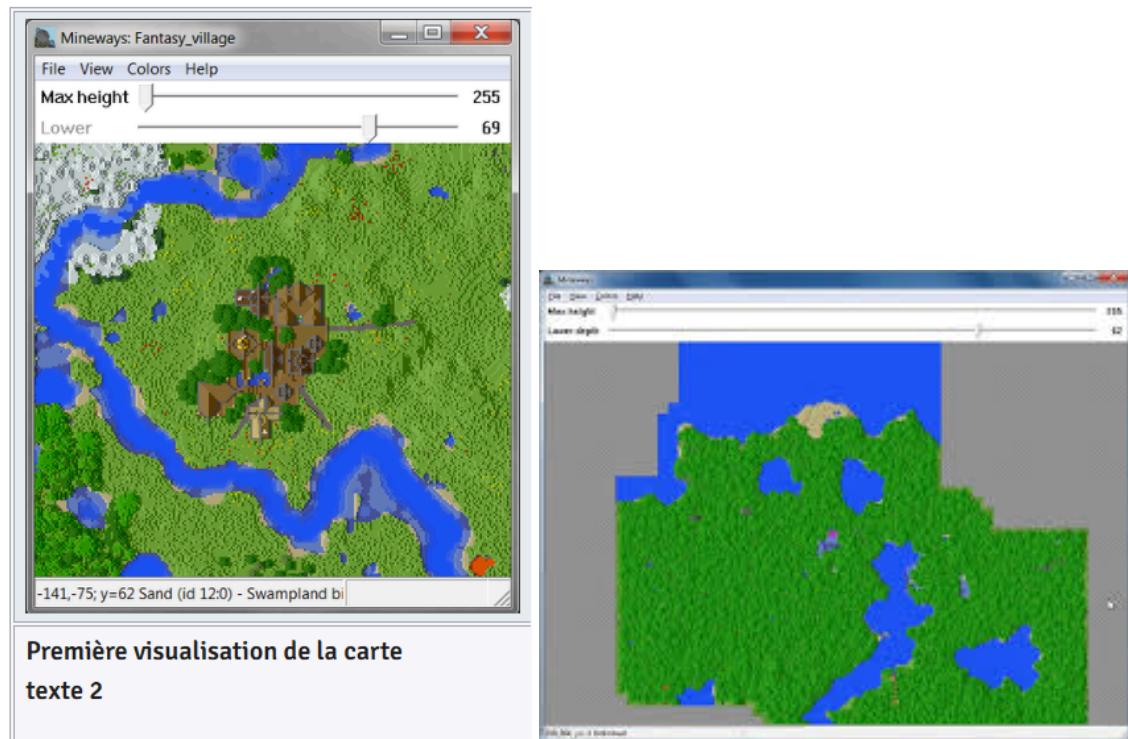
-L'imprimeur qui alterne entre les 2 type de build, par curiosité ou à besoin d'un build avec des conditions spécifiques.

2.2)Benchmark

Lors du Benchmark concurrentielle, j'ai remarqué 2 acteurs majeurs, on à:-Sketchfab: un site web qui permet de publier et vendre des modèles 3D



-Mineways: logiciel qui scanne une partie du monde Minecraft et en fait un modèle 3d.



Mais il existe aussi plusieurs acteurs indépendants proposant des modèles 3D en rapport avec Minecraft ou des solutions alternative comme:
Turbosquid, BlockBensh, Amulet Editor, CurseForge, MCEdit, MCbuild.

Sketchfab:

avantage	inconvénient
Visualisation interactive	Limites de la version gratuite
Facilité d'utilisation	n'est pas spécifique à minecraft
Communauté active	nécessite un autre logiciel pour exporter, un monde minecraft

Mineways:

avantage	inconvénient
Documentation complete	gourmand en ressources pour de grande structure
Gratuit	logiciel donc est à installer et est indisponible sur mobile
Monopole	utilisable qu'en local

Pour conclure, J'ai créé mon application en m'inspirant de ces 2 concurrents. D'une part comme Mineways, mon application peut scannée un monde Minecraft et le récupérer en modèles 3d. D'une autre part, comme Sketchfab, on peut publier la scanne afin que tous puissent avoir accès au modèle.

A noter que quasiment tous les modèles de construction Minecraft qui sont sur Sketchfab ont été scannés avec Mineways.

On peut voir que j'utilise les fonctionnalités principales de chaque concurrent en une application.

Il y a également plusieurs petit concurrent qui propose des logiciels similaire à mineways, mais il faut, le fichier source d'un monde Minecraft (.shematique), et donc indisponible sur un server en route (certain modPacks peuvent contourner cela, mais il y aura plus de travail pour l'administrateur du serveur, donc moi).

L'écosystème de mon application me permet de récupérer les informations du serveur dynamiquement et sans avoir à stopper le serveur Minecraft. De plus, on peut publier ses constructions et donc avoir une bibliothèque de construction minecraft.

3.1)Maquettage et Wireframe

Pour les couleurs, je me suis permis de choisir des couleurs similaires aux site officiel de Minecraft (noir et vert). Malgré le fait que l'application soit dédiée aux ordinateurs j'ai comme même fait en sorte qu'elle soit un minimum responsive, pour les visiteurs mobile.



fn-mc
420

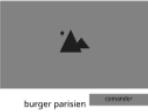
Home Structure Tuto Connection

Quelque exemple :

Nos best burger


 raclette burger
conservé


 burger Veggie
conservé


 burger parisien
conservé

Notre server :

Loreum ipsum dolor sit amet consectetur. Sed dictum risus at odio aliquam vestibulum et amet sem. A ac amet cras laboreet pharetra sit facilisi molestie scelerisque. Nullam nec auctor nunc non id. Ac potenti vulputate ullamcorper morti augue a eu sit.



HYLEXMC

357/2500 en cours de lecture

HYLEXMC est le meilleur MINECRAFT --

REGARDER LA BANDE ANNONCE

[sm.hylexmc.net](#)
[COPIER L'IP](#)

Fait ton Build

Loreum ipsum dolor sit amet consectetur.



En 5 étapes

1. Rejoint le server
2. Construit
3. Enregistre
4. Connecte toi
5. Publie ta création

C'est partis

Structure Public









[Voir plus](#)

Récemment ajouté





Le Summer est Arrivé

Loreum ipsum dolor sit amet consectetur. Sitiamet veluptate donec fasse diam et sed. Amet grande sapien morbi sem. Ut enim tellus tellis laoreet etiam semper sem. Ut elementum erat enim venenatis nulli sit curvus nesciunt.

[Lire la suite ...](#)

Le Summer est Encore Arrivé

Loreum ipsum dolor sit amet consectetur. Sitiamet veluptate donec fasse diam et sed. Amet grande sapien morbi sem. Ut enim tellus tellis laoreet etiam semper sem. Ut elementum erat enim venenatis nulli sit curvus nesciunt.

[Lire la suite ...](#)

c'est un site à but non commercial et donc il est interdit d'utiliser les ressources officielles de Mojang

Réaliser dans le cadre d'une formation de Doranco

12 Rue de plancha
750020 Paris

101001:phone
Super.resto@gmail.com

Voici la page où l'on peut visualiser les données privées une fois connecté.

Fn-mc

Home Structure Tuto MyStruct

Build priver de : tds_ipa

1. titre_choisie
description_test

2. titre_choisie2
description_autre2

3. titre_choisie
description_test

4. titre_choisie
description_test

5. titre_choisie
description_test

suivant

télécharger **texture pack**

up **douwn**

left **right**

zoom + **zoom -**

Pour tous les utilisateurs, voici comment sont affichées les données avec les structures publiques.

Fn-mc

Home Structure Tuto Connection

structure_public

crée par: **Nothe**

Lorem ipsum dolor sit amet
consectetur.

Lorem ipsum dolor sit amet consectetur. Velit vel diam sagittis faucibus
proin risus. Odio sagittis scelerisque orci libero purus pellentesque
integer ipsum consectetur. Dignissim porta eget sollicitudin eget ut nibh
egestas. Amet pretium mattis in sed mi nisl ut donec rhoncus.

télécharger **texture pack**

up **douwn**

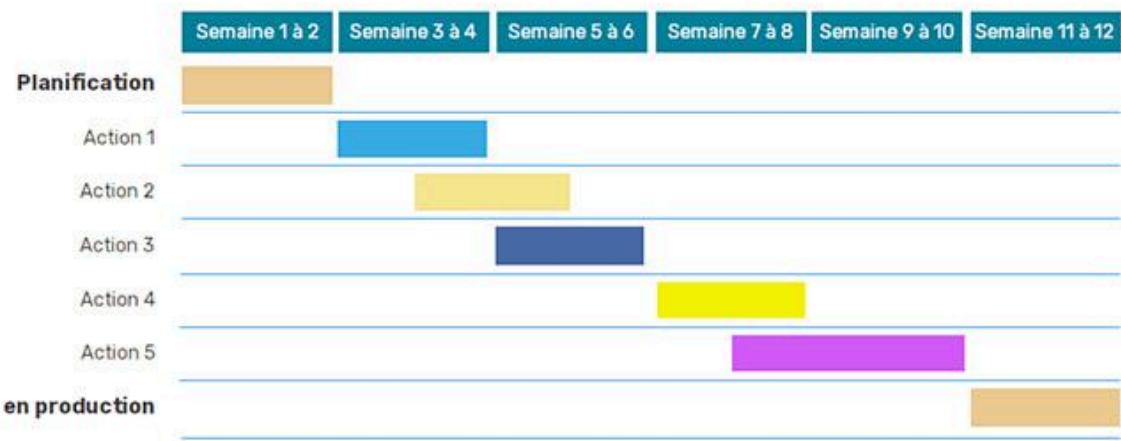
left **right**

zoom + **zoom -**

Partager

publier depuis : 19/03/2025

3.2)Diagramme de gantt

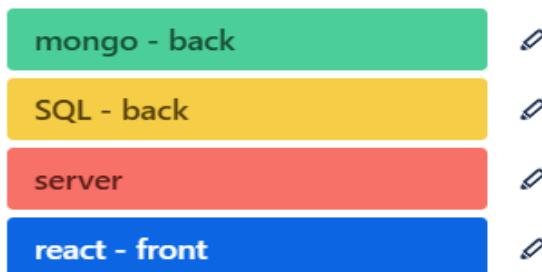
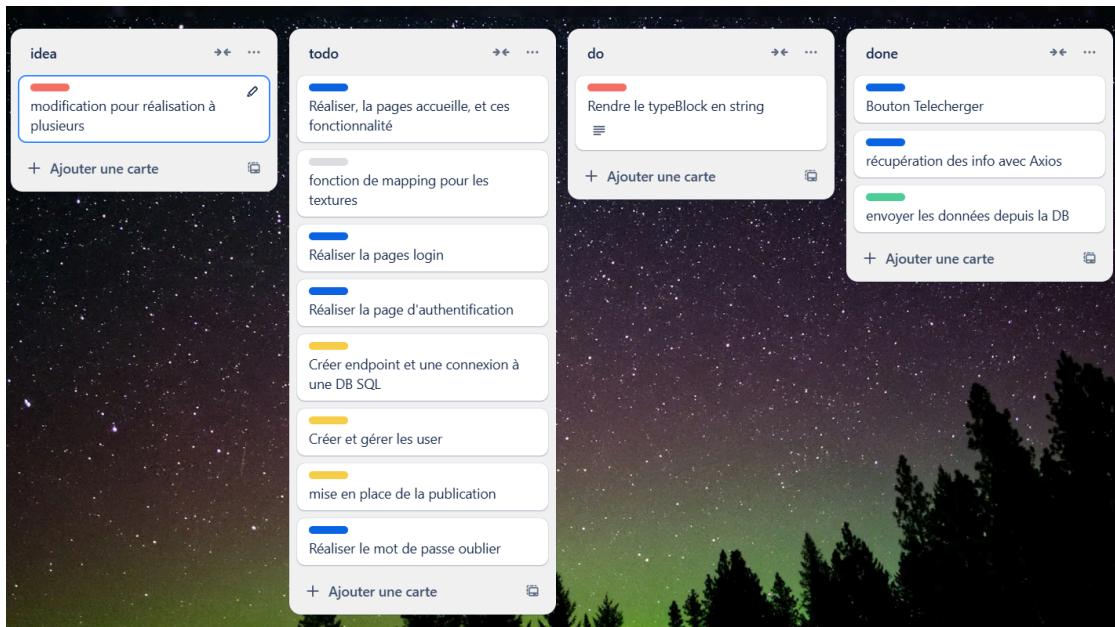


Légende

- action 1 : Structure globale du site web
- action 2 : Server minecraft et structure des données
- action 3 : Api MongoDB et sécurité
- action 4 : Api Sql et finition du front-end
- action 5 : Docker et Jenkins

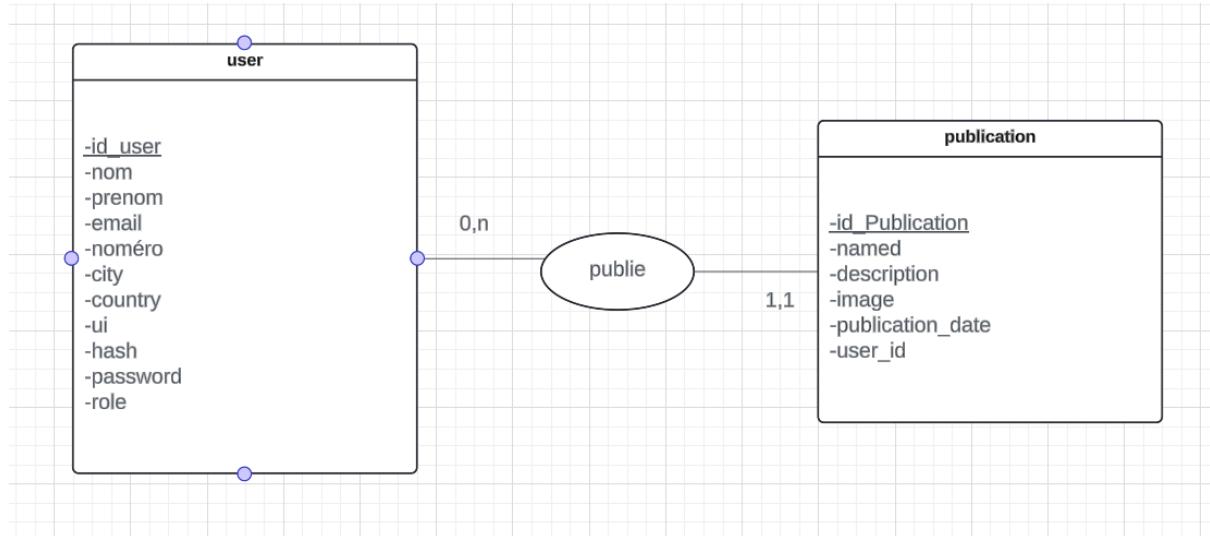
3.3) Méthode agile

Pour ma part, étant donné que je suis seul sur ce projet, j'ai utilisé la méthode kanban, avec l'outil Trello.

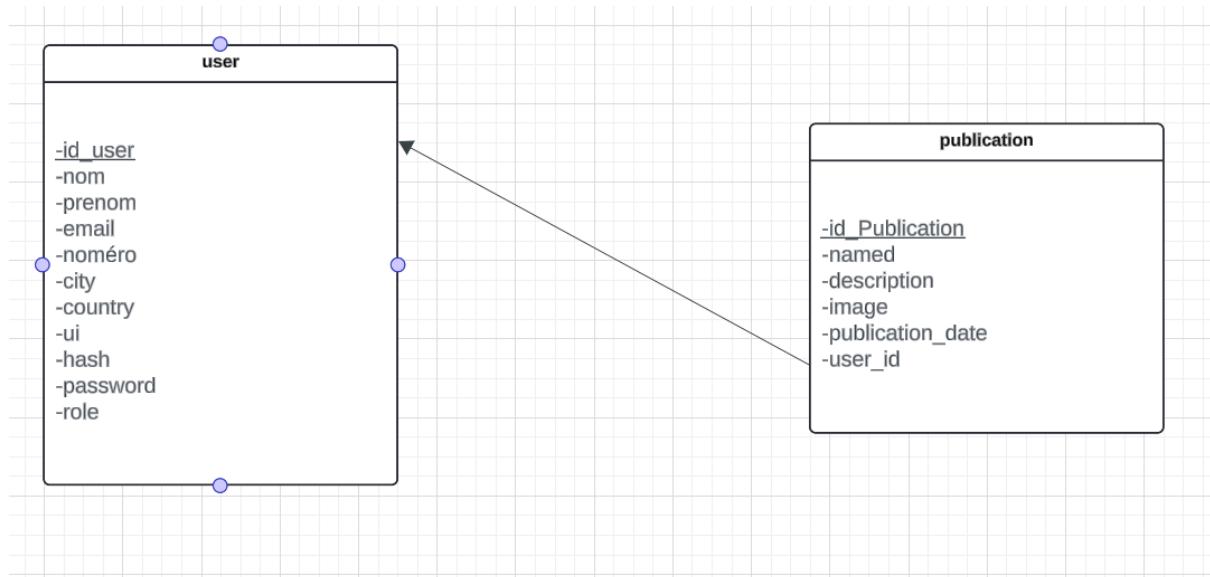


Cette méthode me permet de mieux visualiser les tâches que j'ai à faire et de m'organiser plus facilement dans ce projet, mais également avec d'autres projets.

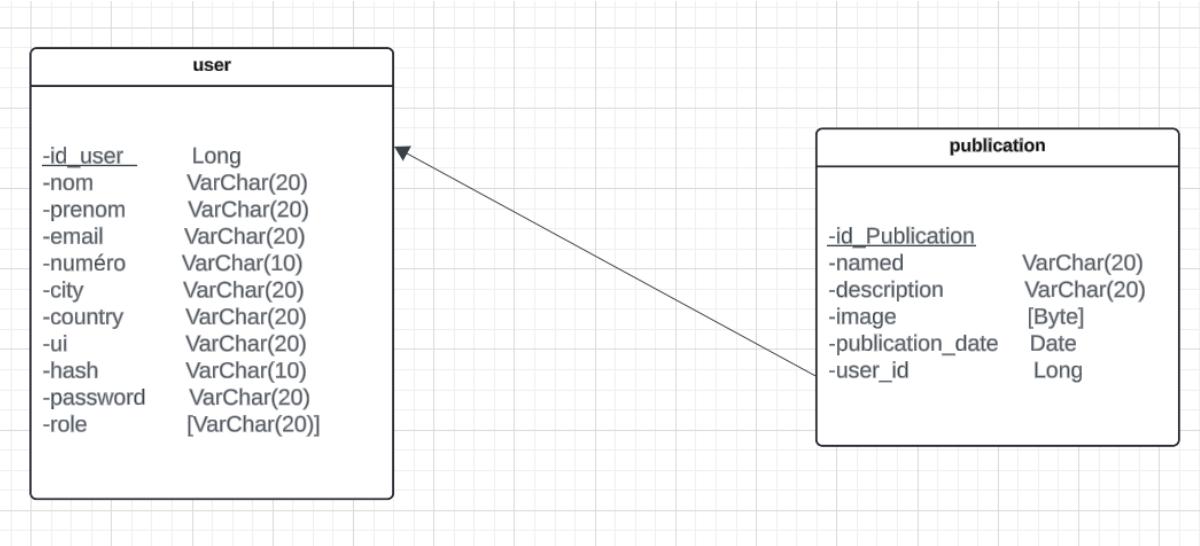
3.1) MCD



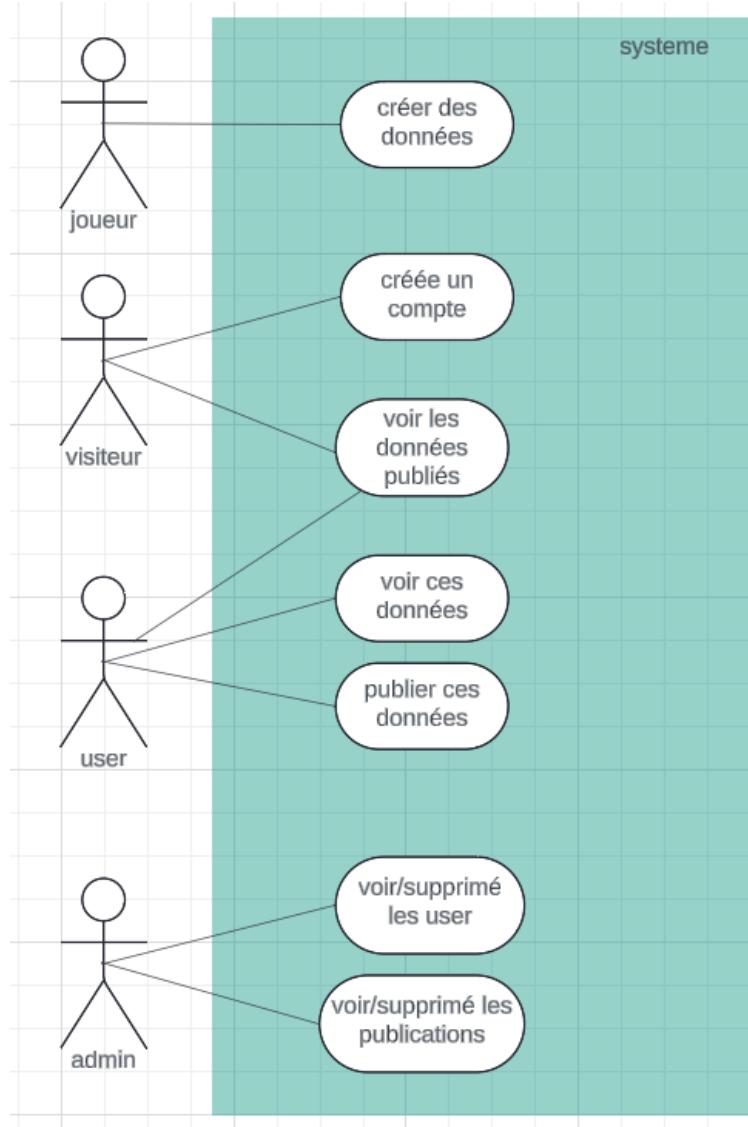
3.2)MDL



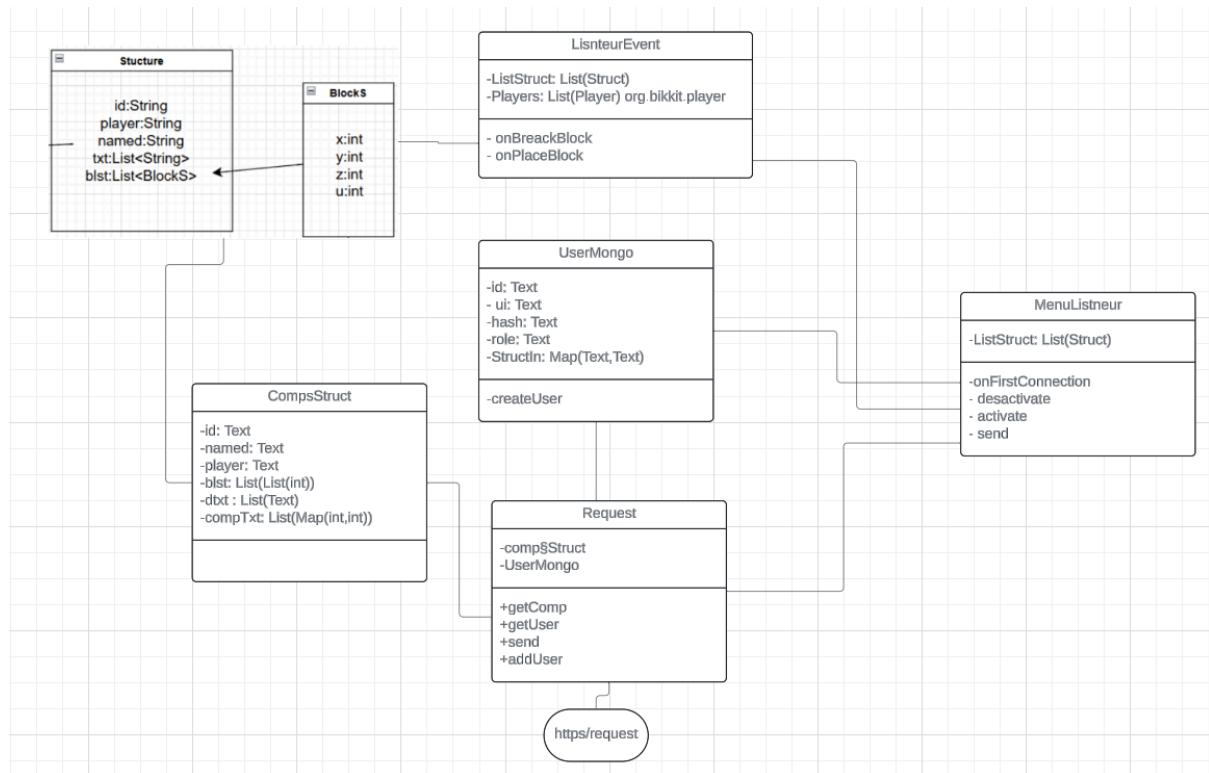
3.3)MPD



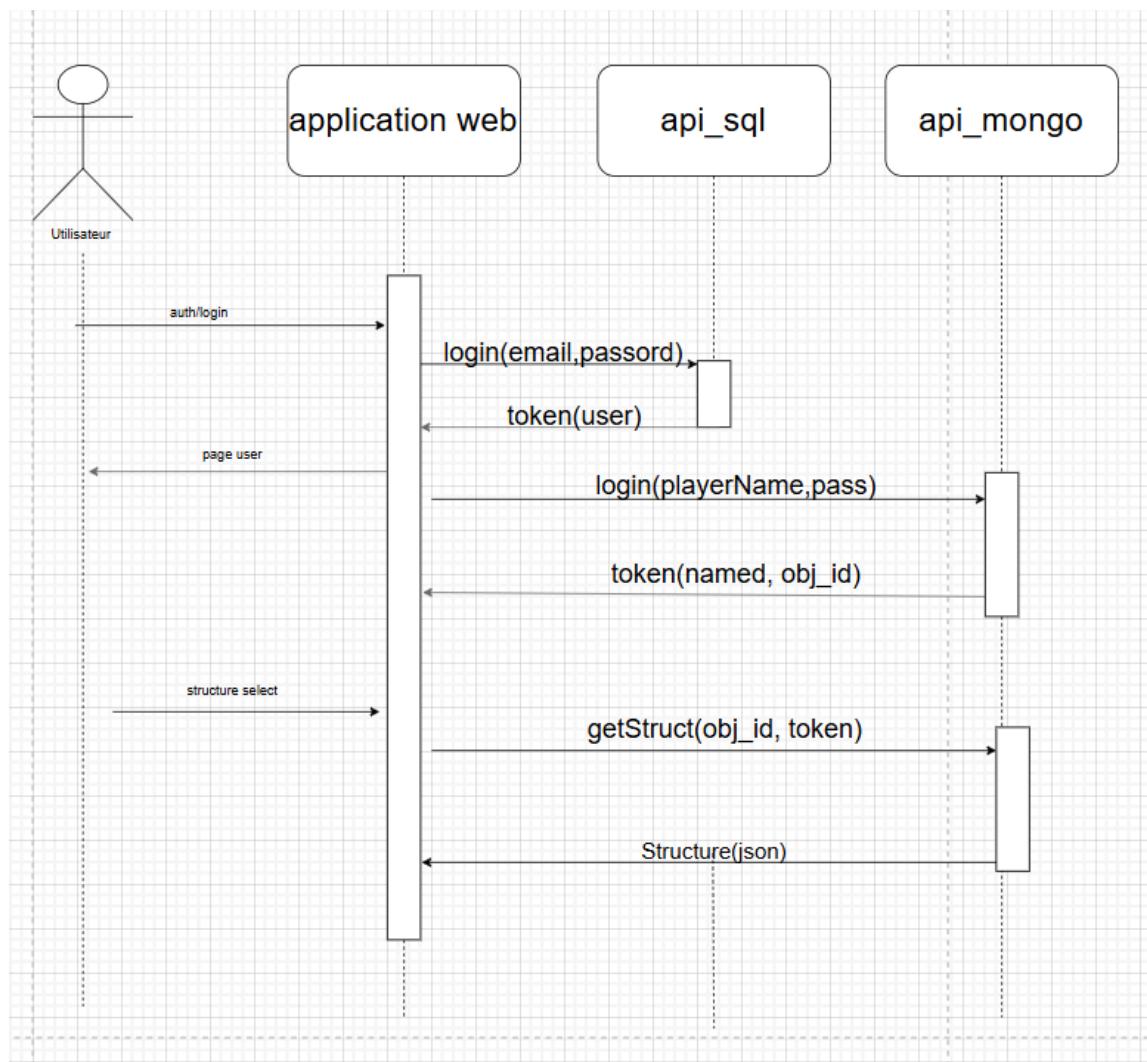
3.4) diagramme de cas d'utilisation

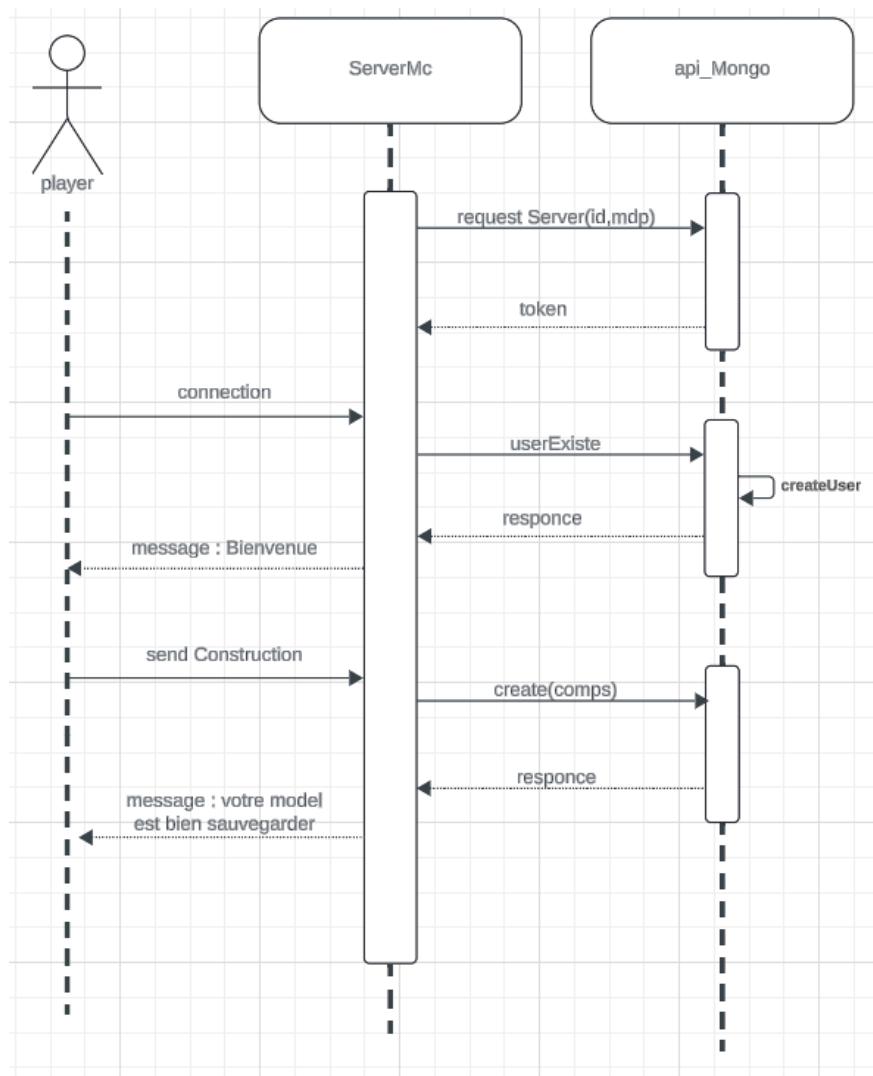


3.6)diagramme de classe

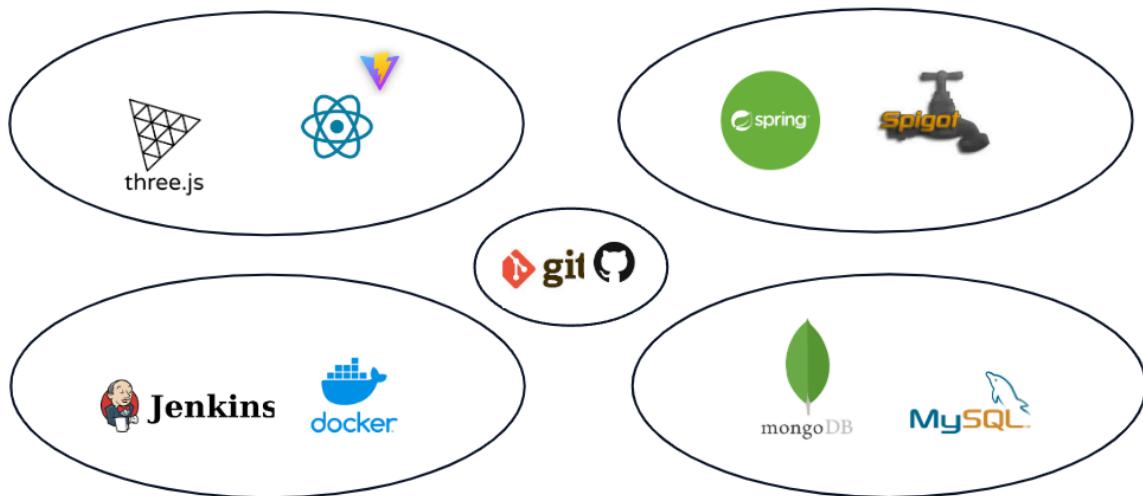


3.6)diagramme de séquence





4.0)Environnement technique



Voici un schéma qui regroupe tous les stacks utilisés.

5.1.1) React

A) Installation de l'environnement React avec node:

installer la version stable (TLS) de nodeJs sur le site <https://nodejs.org/>.
Puis après avoir téléchargé et lancé .msi sur windows, on ouvre un terminal.
Exécute la commande suivante:
`#node -v`

Si on à bien la version télécharger afficher sur l'écran,c'est que node est désormais installé sur la machine. on peut installer réact avec:

`#npx create-react-app nom-project`

Mais pour ma part, j'ai utilisé Vite Js car plus rapide et fait toujours partie de l'écosystème de react.

Donc j'ai exécuter cette commande:

`#npm create vite@latest nom-project`

Une fois arrivé à cette fenêtre on sélectionne Réact, et cela va créer le dossier du projet.

```
? Select a framework: » – Use arrow-keys. Return to submit.
  Vanilla
  Vue
> React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Angular
  Others
```

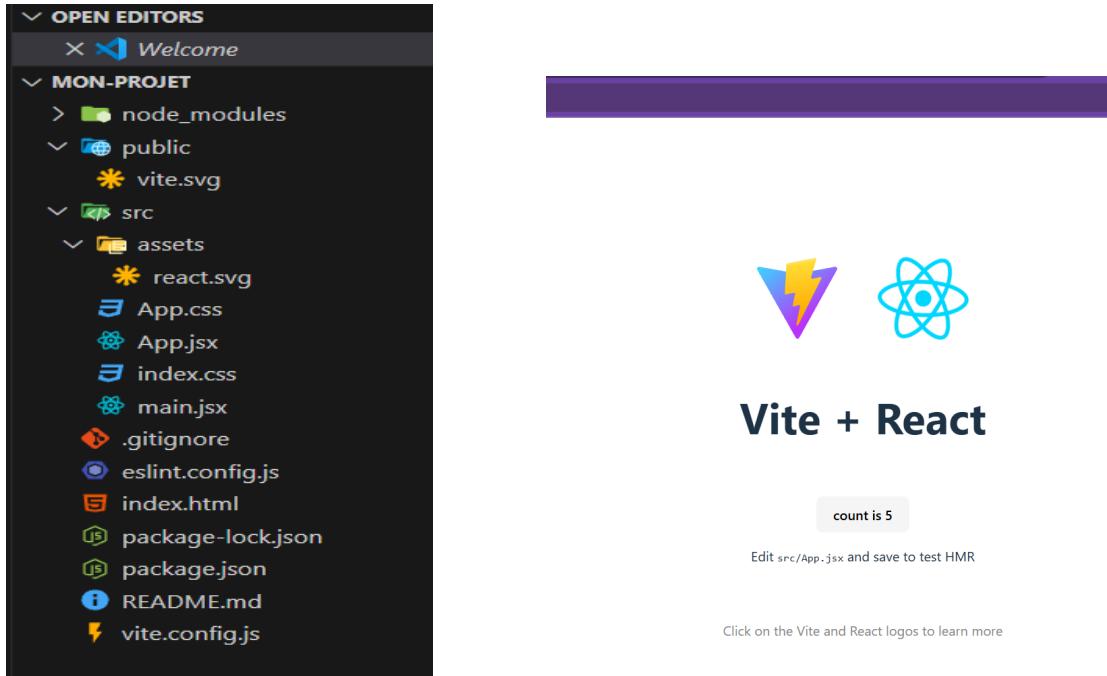
Une fois créer j'exécute ces 3 commandes, pour aller dans le dossier puis installer les dépendances, avant de lancer le projet.

```
#cd nom-project
#npm install
#npm run dev
```

```
VITE v6.1.0 ready in 495 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Voici les informations données une fois le lancé.

Par défaut, Vite nous donne une architecture de base dans le code ainsi qu'un exemple de site type.



J'ai utilisé Vite Js puisque en plus d'être compatible avec réact, il peut démarrer très rapidement.

B) 4.1.2) Librairie Utiliser

L'avantage de réact c'est que l'on peut utiliser une grande panoplie de librairie différente. Celle que j'ai utilisés sont:

- axios: pour les requêtes https
- React-routeur-dom: pour le rooting de l'application (notamment via le composant Link)
- react-calendar: afin d'avoir des ui de calendrier préfabriqué
- threeJS: visualiser et modéliser des models 3D

```
},
"dependencies": {
  "axios": "^1.7.7",
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router-dom": "^7.1.3",
  "three": "^0.167.1"
},
```

Toutes les dépendances sont notées dans le Package.json

C) 5.1.1) Utilisation de composant

Réact est particulièrement bien pour créer des éléments, des composants réutilisables, modulaires, afin d'économiser des lignes des codes et simplifie l'évolutivité de l'application.

Inscription

nom
Prénom
email
465
retaper le mot de passe
téléphone
uui Minecraft
hash

Valider

email:
mot de passe:

Valider s'inscrire

Dans cet exemple, on peut voir que l'input de l'email est un composant. Que j'ai utilisé pour le "mot de passe" et le formulaire d'inscription

En plus de pouvoir nous éviter des tout réécrire, les composant peuvent être personnalisé via des paramètres, ce qui nous permet d'optimiser le code au maximum.

```
<form className="form" onSubmit={sub}>
  <Input params="nom" />
  <Input params="Prénom" />
  <Input params="email" />
  <Input params="mot de passe" />
  <Input params="retaper le mot de passe" />
  <Input params="téléphone" />
  <Input params="uui Minecraft" />
  <Input params="hash" />
  <button className="formBtn">Valider</button>
</form>
```

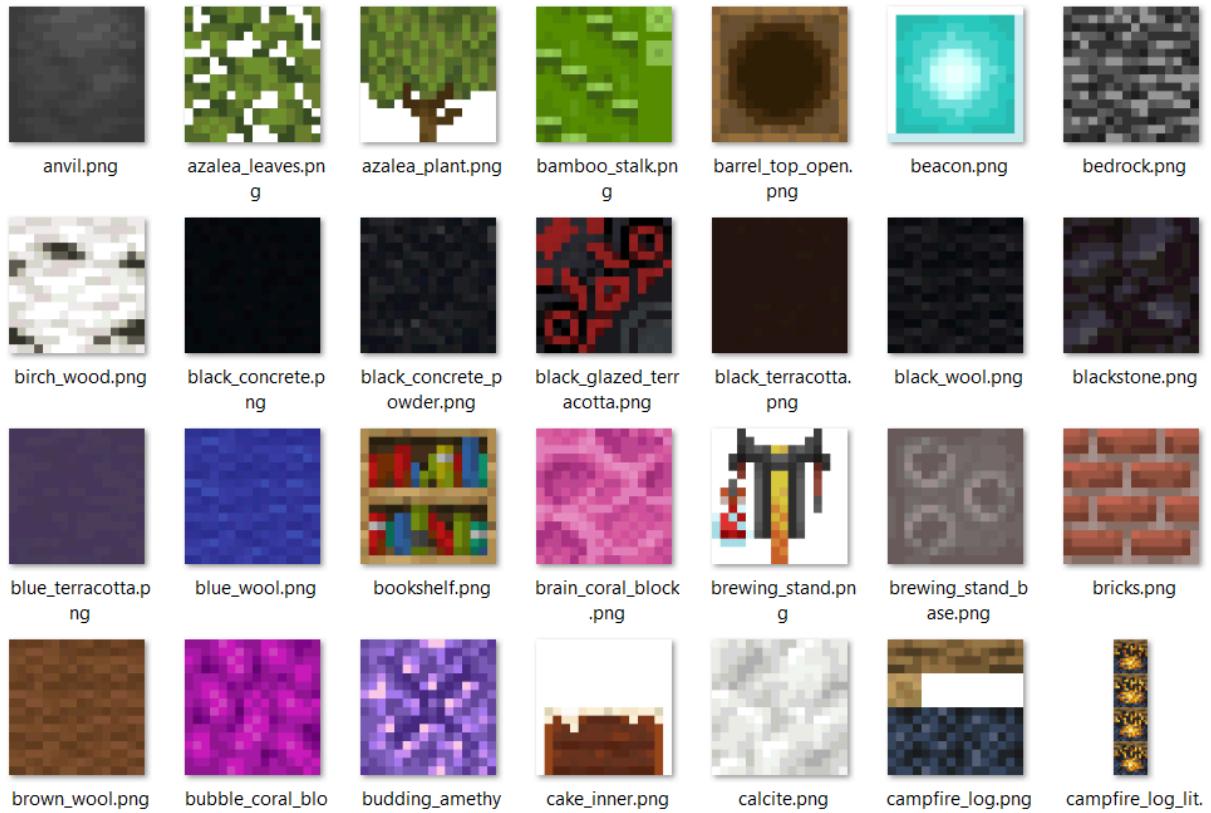
```
{tab.map((element) => {
  return <Input params={element} />;
})}
```

D) 4.1.3)architecture web

```
▽ src
  > components
  > data
  > layout
  > pages
  > threeJS
  > utils
  ⚡ App.css
  ⚡ App.jsx
  ⚡ index.css
  ⚡ main.jsx
```

J'ai modifier les dossier src pour ajouter un dossier layout et pages, qui me serviront à gérer les comptes des utilisateurs.

Le dossier Public contient toutes les textures officielles du jeux, les images sont toutes au format .PNG et ont une résolution de 128*128 et un poids moyen de 1000 octets par images.



Ce sont ces textures qui seront utilisées pour générer un modèle.

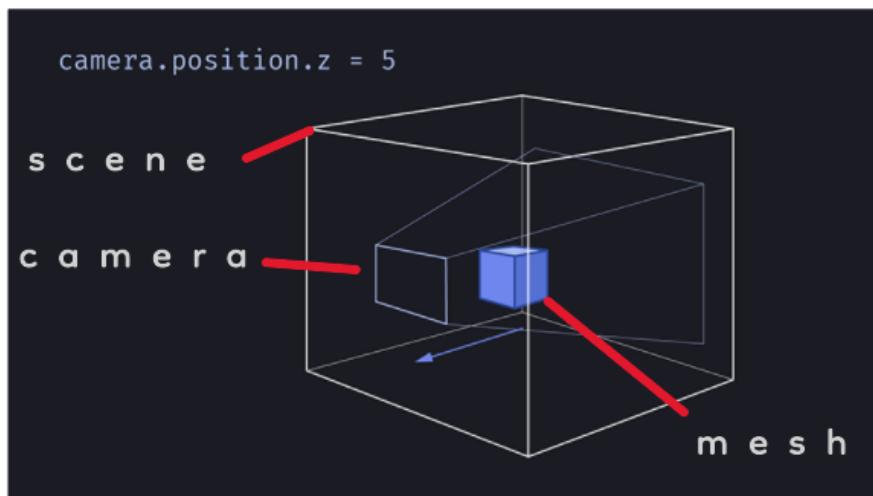
Elles sont également disponibles sur : <https://minecraft.fr/categorie/resource-pack/>, où l'on retrouve les texture-pack officiel ou réalisé par la communauté.

5.1.2)Three Js

A) Un moyen de visualiser les données

Il existe plusieurs façons de modéliser des objets sur le web surtout avec javascript, pour ma part j'ai utilisé Threejs, car en plus être compatible avec Vite.Js, cela était pour moi simple à apprendre.

Pour un peu de context, le fonctionnement de Three JS suit le schéma suivant :



On à une scène qui peut contenir plusieurs assets ou mesh, une caméra qui va nous renvoyer une projection de la scène, et facultativement une Lumière afin de bien voir les reflets pour un rendu de meilleurs qualité.

La scène est en quelque sorte le lieu où l'on va placer des acteurs tels que les mesh, caméra, lumière. Tous ces acteurs ont des coordonnées de position et de rotation avec pour 3 de valeurs (X,Y,Z). On peut avoir une ou plusieurs scènes dans une application.

THREE.Scene() permet de créer une scène ,scene.add() d'ajouter un élément à cette scène.

```
const scene = new THREE.Scene();
scene.add();
```

La caméra renvoie une projection de la scène et donc les objets qui sont dans son champ de vision. Comme les mesh elles ont des coordonnées qui part défaut sont à 0 et si on à une mesh au même endroit, la caméra sera à l'intérieur et on ne verra rien. Donc avec on l'utilise la caméra à des coordonnées différentes.

Puis pour ajouter la caméra dans la scène avec Three Js, on utilise:

```
const camera = new THREE.PerspectiveCamera(75, size.width /  
size.height);  
scene.add(camera);
```

Enfin pour déplacer cette caméra j'utilise des boutons qui onClick, me permet de modifier ces coordonnées avec un fonction, et dynamiquement grâce au render qui met à jour la canvas.

```
camera.translateY(-1.5);
```

```
renderer.render(scene, camera);
```

Pour récupérer le rendu de la caméra, je l'associe ThreeJs à une canvas qui a été ajoutée dans index.html, canvas qui va servir d'écran sur laquelle on va afficher la projection.

```
const size = {  
    width: 800,  
    height: 600,};  
const canvas = document.querySelector("canvas.webgl");  
const renderer = new THREE.WebGLRenderer({  
    canvas: canvas,  
});  
renderer.setSize(size.width, size.height);  
  
<body>  
| <div id="root"></div>  
| | <canvas class="webgl"></canvas>  
| | | <script type="module" src="/src/main.jsx"></script>  
| </body>  
</html>
```

Il est important de savoir que l'on peut avoir au maximum une caméra par scène, ou une caméra pour plusieurs scènes. Mais qu'un seul render par application.

On peut voir que Three Js utilise webGL(web graphique librairie) ce qui fait que n'importe quel navigateur puisse avoir des images en 3D, et rend l'application portable.

Les Mesh dans Three Js, sont composé de 2 chose:

-La géométrie est la forme que va prendre une mesh (triangle, rectangle, cylindre) mais pour ma part c'est un cube simple avec 1 pour unité de longueur sur chaque côtés.

-Le matériel, défini la texture et type de matière que va avoir l'objet.

Elle est composée de plusieurs choses dont les textures qui sont dans le dossier /public.

Pour le reste des caractéristiques d'un matériel, je n'ai pas l'occasion de l'utiliser car Minecraft est un jeu qui ne cherche pas à être réaliste. En effet les autres caractéristiques vont jouer sur la réflexion de la lumière, la rugosité etc, ils servent surtout à avoir un rendu avec une qualité réaliste.

B) 5.1.2) Comment charger les données

Nous allons voir, comment avec une réponse Json, on peut construire un modèle identique à ce que le joueur a construit.

Tout d'abord il faut savoir que pour récupérer une structure, la requête demande soit l'identifiant, soit (le nom et le joueur), puis l'api Mongo répond en Json une structure sous le format suivant.

```
_id: ObjectId('679d3415222dab0a5f3bbe7')
player : "tds_lpa"
named : "bluem"
▶ blst : Array (78)
  ▶ compData : Array (empty)
  ▶ dtxt : Array (3)
    0: "warped_nylium"
    1: "netherrack"
    2: "magma_block"
  ▶ compTxt : Array (9)
    ▶ 0: Array (2)
      0: 1
      1: 0
    ▶ 1: Array (2)
      0: 15
      1: 1
  ▶ 2: Array (2)
  ▶ 3: Array (2)
```

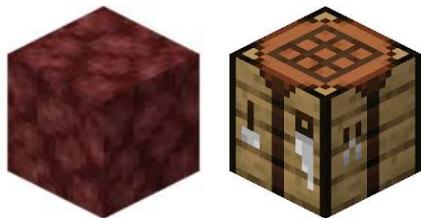
-ID: on a l'id de la structure
-player: le nom du joueur de la structure
-named: le nom de la structure
-blst: une liste de position qui est un object qui contient 3 integer (X,Y,Z)
-compsData: c'est pour les rotations, mais plus tard ce servira pour les mesh qui ne sont pas des cubes
-dtxt: puis on a une liste qui contient les noms des blocs utilisés pour faire la structure
-compTxt: enfin on a une listes qui indique que tel texture appliquée sera appliquée à tel bloc

Une fois que l'on à récupéré le Json, on peut commencer à créer des meshs puis modifier leurs coordonnées en fonction de blst.

Ensuite, on doit appliquer la bonne texture à la Mesh que l'on à créé, pour cela on va utiliser comTxt et dtxt. Par exemple, pour une boucle qui va de index=0 à la taille max du tableau de blst (78), on peut voir que lorsque l'index ==15 on applique là texture numéro 1 donc "netherrack".

Les textures sont préchargées dans un tableau de matériel, dans un 1er temps nommer arlod, puis dans une boucle on associe un matériel et une géométrie dans une mesh.

```
const useMesh = new THREE.Mesh(gEO, arlod[itc]);
```



Enfin, il faut un système pour appliquer les bonnes textures sur toutes les faces, si le block à plusieurs faces.

```
case "_side":  
    all.side = new THREE.MeshBasicMaterial({  
        map: texture.load("/block/_side/" + element +  
"_side.png"),  
    });
```

Pour ce faire, j'ai malheureusement utilisé un fichier Json qui contient une liste de tous les blocs, et indique sur quelle faces ce bloc à tel texture.

```
["smooth_stone_slab", ["_side"]],  
["grindstone", ["_side", "main"]],  
["warped_nylium", ["_top", "_side"]]
```

Plus tard, je serais amené à utiliser des voxels, qui sont en résumé comme des pixels mais en 3d, mais qui servent à beaucoup de choses.

Je ne les ai pas encore implémenter, car complexes à comprendre et j'ai eu un temps limité.

Ces voxels vont me servir à gérer les textures via des coordonnées UV complexes.

Cela va me servir à optimiser le stockage des matériaux, surtout que ce dernier est assez lourd à stocker.

Mais pour importer un model généré par l'application dans un autre logiciel il faut le télécharger, pour ce faire on va créer un fichier .GLTF, qui est grossièrement c'est un format pour les model 3d destiné au web, qui est composé d'objets json.

Le .GLTF comme la plupart des fichiers de rendu 3d, sont comme une recette de cuisine, Le fichier contient toutes les instructions et informations pour construire le modèle. Ainsi peut assez facilement, avec un logiciel comme blender le reconvertis dans une autre extensions comme: .blend / .obj / .fbx / .stl / .gltf .

Pour ma part j'ai choisi d'exporter en .gltf car plus compatible au web et disponible par Three via GLTFExporter.

```
const exporter = new GLTFExporter();
  exporter.parse(
    scene,
    function (gltf) {
      download(JSON.stringify(gltf));
    },
  );
```

Je modifie au préalable, le nom du fichier par le nom de la structure, puis je rajoute l'extension.

Puis j'export la scène en Json (la caméra n'est pas pris en compt);

Le reste du téléchargement est géré par le navigateur.

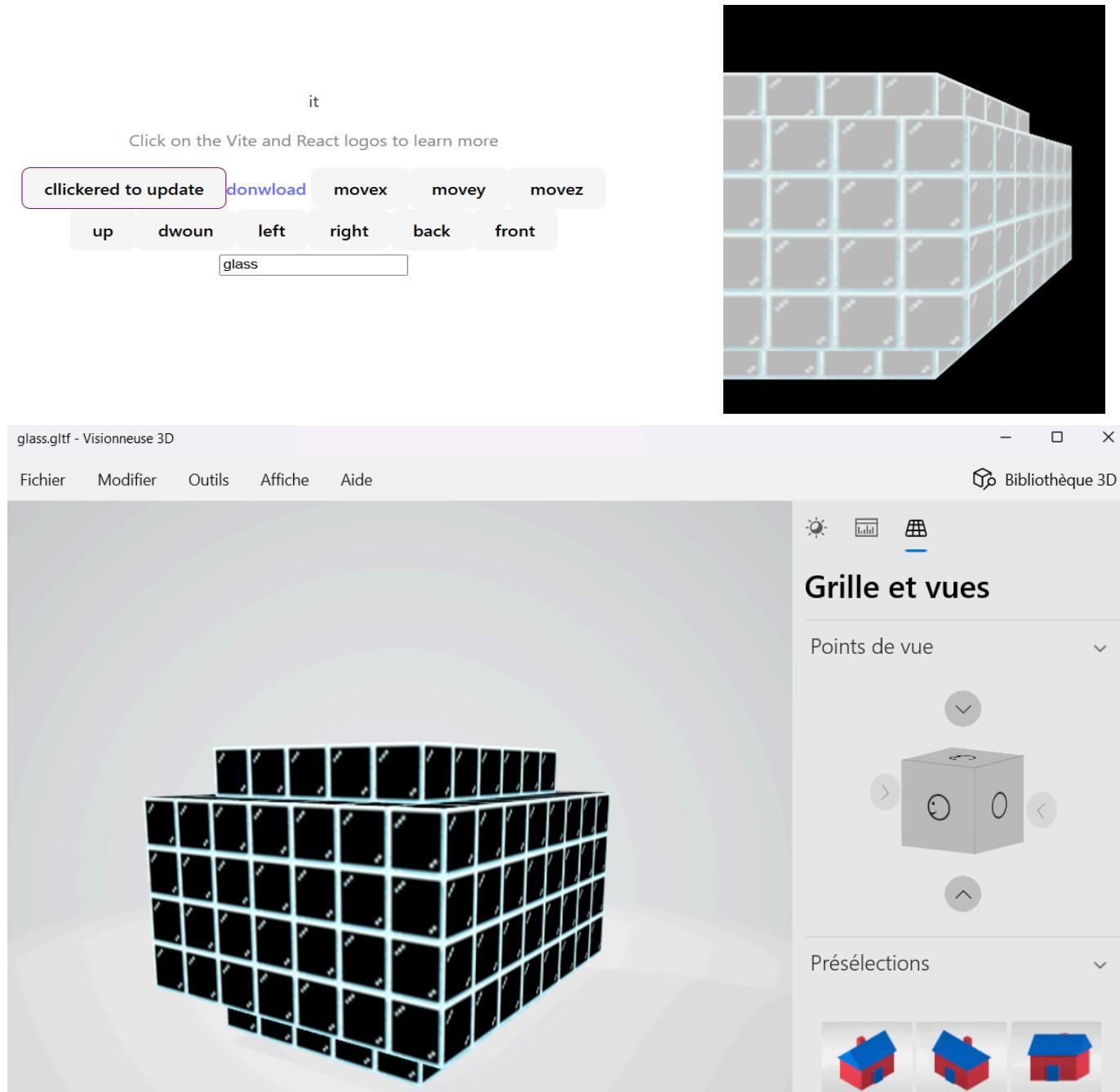
```
const name = named + ".gltf";
```

 glass.gltf	31/01/2025 22:00	3D Object	13 Ko
 home2.gltf	31/01/2025 21:58	3D Object	15 Ko

A noter que cette démarche fonctionne aussi avec le .obj

```
const expo= new OBJExporter()
```

Voici un exemple de comparaison, pour un résultat de qualité moyenne.



La version télécharger sur la Visionneuse Windows.

C) 5.1.2) Les limites et avantages

Voici sous forme de tableau un tri des avantages et inconvénient des technos utilisées pour la partie Front-end.

THREE JS

avantage	inconvénients
Facile à utiliser	Performances
Multiplateforme	Gestion de la physique
Compatible avec Réact	Documentation détaillée

Format .glTF

avantage	inconvénient
Léger	La conversion depuis d'autres formats peut parfois créer des erreurs.
Prend en charge les textures et animation	/

Vite JS

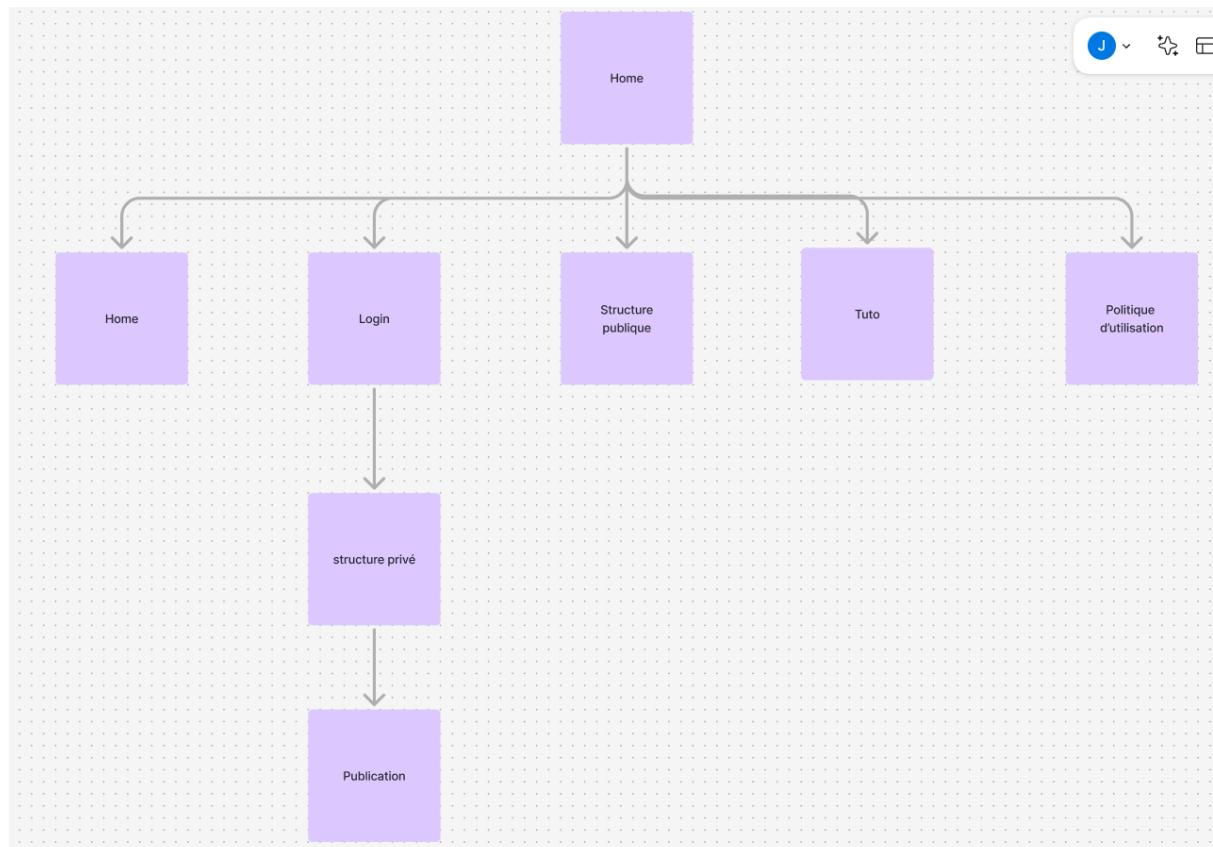
avantages	inconvénient
démarrage et build rapides	plus d'actualité
Simplicité	/

Réact

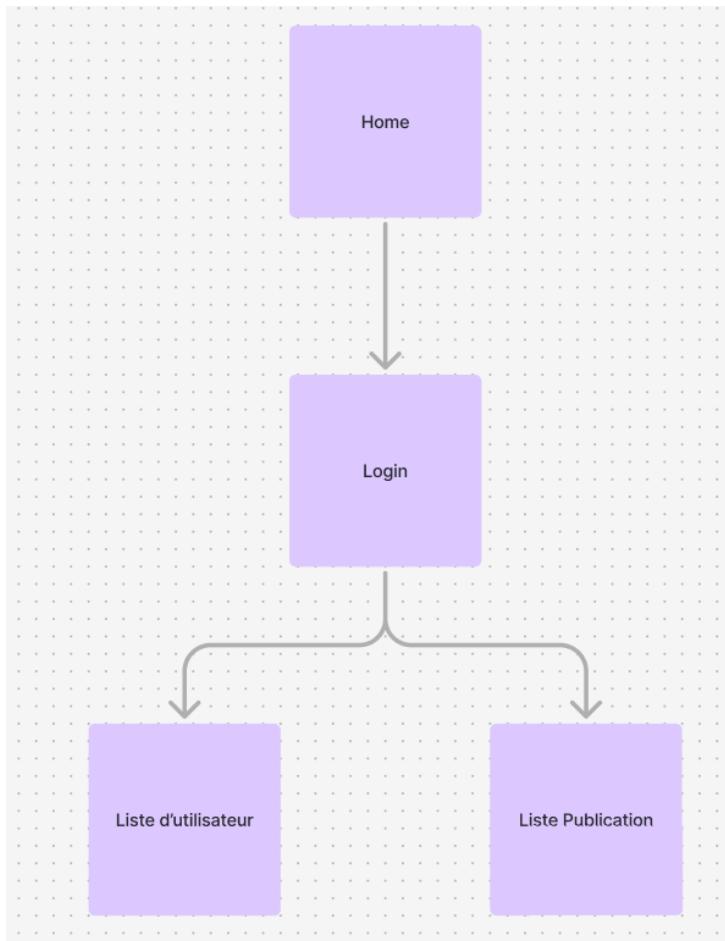
avantage	inconvénients
Composant Réutilisable	Difficile pour débuter
Écosystème riche	L'organisation du code peuvent devenir complexes
Virtuel DOM donc plus rapide	/

5.1.3) Arborescence de l'application web

A) Utilisateur



B) Admin



5.2.1)Spring Boot

A) installation de Spring

Ils existe plusieurs moyen de créer un projet spring, pour ma part j'ai utiliser:
spring initializr -> <https://start.spring.io/>

B) Les dépendances (**dependency**)

j'ai choisie les langages Java et Maven, et utiliser les librairie suivantes:

lombok

Afin d'avoir des annotations permettant une écriture de code plus rapides.

spring-boot-starter-security

Pour la sécurité de l'application.

jjwt-jackson

Pour générer et utiliser des token, pour un trafic plus sécurisé.

spring-boot-starter-web

Afin d'avoir des endpoint sur son application tout en respectant l'architecture MVC.

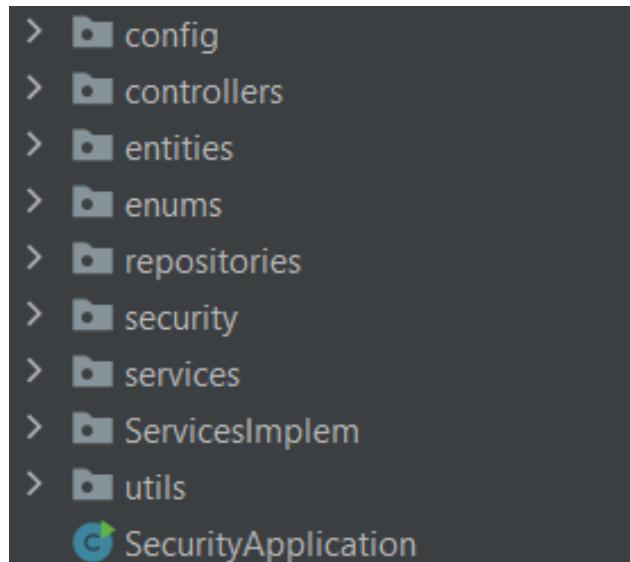
```
pour mongoDb  
spring-boot-starter-data-mongodb  
  
pour SQL  
spring-boot-starter-data-jpa  
mysql-connector-j
```

Dans mon cas j'ai 2 bases de données(une SQL , et une MongoDB), donc j'ai mis en place 2 api (indépendantes) qui communique à leurs base de données respective.

J'utilise Jpa afin de communiquer et de persister les données.

5.2.2) Architecture MVC

J'ai opté pour une architecture mvc, car elle permet une organisation structurée et une bonne lisibilité. Mais également, permet de collaborer facilement entre développeurs tout en offrant une bonne modularité.



A) Model

Les modèles sont responsables de la gestion des données et de la logique métier de l'application.

Ils récupèrent et manipulent les données depuis différentes sources (comme des bases de données) et les préparent pour les contrôleurs. En centralisant la gestion des données, les modèles facilitent les mises à jour et la maintenance de l'application.

Ils garantissent également l'intégrité des données grâce à des validations et des règles de gestion. En utilisant un modèle structuré, on assure une organisation claire et efficace du code.

Les modèles peuvent être réutilisés dans différents contrôleurs et vues, ce qui réduit la redondance et améliore la cohérence de l'application. Ainsi, les modèles permettent une gestion efficace et structurée des données, en intégrant les règles métier essentielles.

```
import java.util.List;
18 usages
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Comps {
    no usages
    @Id
    private String id;

    no usages
    private String player;
    no usages
    private String named;
    no usages
    private List<List<Long>> blst;
    no usages
    private List compData;
    no usages
    private String compData2;

    no usages
    private List<String> dtxt;
    no usages
    private List<String> compTxt;
    no usages
    private boolean isPrivate=false;
```

Un exemple de modèle compatible avec MongoDB.

B) Vue

La couche vue dans une architecture MVC (Modèle-Vue-Contrôleur) est responsable de la présentation des données à l'utilisateur. Elle reçoit les données du contrôleur et les affiche sous une forme compréhensible et utilisable. Les vues sont souvent constituées de modèles ou de composants d'interface utilisateur (UI) qui permettent de présenter l'information de manière dynamique et interactive.

Les vues jouent un rôle crucial dans l'amélioration de l'expérience utilisateur, car elles permettent une interaction fluide et réactive avec l'application.

En résumé, la couche vue permet de rendre les données accessibles et attrayantes pour l'utilisateur, tout en garantissant une interaction efficace avec le reste de l'application.

En utilisant des frameworks comme React, la couche vue peut être construite de manière modulaire et réutilisable.

Réact s'intègre bien avec les contrôleurs, affichant les données de manière fluide et interactive. Son riche écosystème de bibliothèques et outils, et son architecture modulaire rendent le développement et la maintenance plus faciles.

C) Contrôleur

Les contrôleurs ont pour mission de récupérer les données des utilisateurs, de les filtrer et de les vérifier, puis de déclencher le traitement approprié (via le modèle) et enfin de déléguer la création du document de sortie à la vue.

Comme mentionné précédemment, l'utilisation des contrôleurs permet également de structurer l'application de manière hiérarchique, ce qui facilite la compréhension du code et l'accès rapide aux parties à modifier. En organisant logiquement une application MVC en contrôleurs et actions, on induit ainsi une organisation normalisée.

Les contrôleurs, comme leur nom l'indique, ont pour rôle essentiel de coordonner les séquences d'actions et de réactions d'une application web.

Ils reçoivent les requêtes, déclenchent une logique basée sur des objets métiers et choisissent la vue qui constitue la réponse à la requête.

```
@PostMapping("login")
@CrossOrigin(origins = "http://localhost:5173")
public ResponseEntity loginE(@RequestBody Map<String, String> request) {
    String nom = request.get("name");
    String password = request.get("password");
    String id = request.get("id");
    Map<String, Object > res = userMcService.login(nom, password, id);

    if (res==null) {
        return new ResponseEntity( body: "ID, ou Mot de passe incorrect", HttpStatus.UNAUTHORIZED);
    }

    return ResponseEntity.ok(res);
}
```

5.2.3)Api Mongo

L'api qui est lié à MongoDB est faite en spring et utilise l'architecture MVC.

Cette Api aura pour rôle de récupérer les informations envoyées depuis le serveur Minecraft, et de les enregistrer dans une base de données. Bien sûr, entre-temps ,on vérifie les données.

Cette api est également liée au front-end afin de pouvoir récupérer ces informations.

J'ai donc mis en place 2 rôles:

- serveur : permet d'ajouter des utilisateurs et d'envoyer des structures.
- Player (user): permet de lire (et de supprimer) des structures liées à lui-même.

Chaque rôle doit entrer un nom d'utilisateur et un mot de passe pour se connecter, puis communique avec un bearer Token, jusqu'à la déconnexion ou l'expiration. l'utilisateur User peut voir uniquement les structures privées qu'il à créé. Alors que les constructions publiques sont visibles par tous.

```
public interface CompRepo extends MongoRepository< Comps, String> {  
  
Optional<Comps> comps= compRepo.findById(id);
```

La demande de connexion du rôle serveur se fait par requête https, et se déclenche si je me connecte sur le serveur en question. On peut remarquer que je n'ai pas encore de rôle admin, car je souhaite plus tard pouvoir ajouter plusieurs serveurs à mon application et les gérer via le compte administrateurs.

Voici un exemple d'user, dans la base de données.

```
_id: ObjectId('672b8c6ec8177e53d789a1da')  
name : ".GreyCactus67220"  
hash : "$2a$10$i2ARnbA19Ac2upSB8BZIve1CyJtjacKQradDOTgBWzEeM00zD2ga6"  
- roles : Array (1)  
  0: "PLAYER"  
- structIn : Object  
  default : "okokok0704"  
  tin7rin3 : "6710de13c1a4140da6af25157"  
  opkr : "672dd2442ab19f48e9bd4091"  
  poro : "6733708440285b5ce2d2d1ba"  
  zz : "6735fee2a7d7bb23f53440be"  
  origin : "67371f335b14ae395c529386"
```

Pour récupérer les structures de chaque utilisateur, j'ai utilisé un "tableau" qui enregistre, les identifiants et leurs noms.

Aussi l'application limite tous les ports sauf le (22555 et 5275)

5.1.4)Api Sql

L'application utilise l'api SQL pour enregistrer ces utilisateurs, et protéger leurs données sensibles. Cette api communique avec l'application Réact et la base de données, car ces derniers sont isolés dans un conteneur docker afin de limiter la propagation de code malveillant.

C'est pour cela que lorsqu'un utilisateur s'inscrit, il doit rentrer son nom de joueur et un code, qui n'est visible que sur le jeu, en plus des champs nom,prénom...

firstname	hash	lastname	num	password	roles	ui
j	15	f	0154657	\$2a\$10\$.BQgulomamsAHvkDh2zPy.JavojGKDE...	BLOB	tdsJ
dedej	Java	cecf	0154657	\$2a\$10\$Odgrjj/cSudvP6e.wXYY.ez3N4xXmBLf6...	BLOB	123
Louison	Java2	Kolia	0781909349	\$2a\$10\$CB3usbwRONicVWYpcgXD6.mSSBDgy...	BLOB	tesl123456
Jessy	p&2D&uoQn...	Laupa	0771881903	\$2a\$10\$fZnnQfl5gv1LWw4yJ17Z2uiWIfkdBPLQ...	BLOB	tds_lpa
HULL	HULL	HULL	HULL	HULL	HULL	HULL

(les valeurs du champs ‘hash’ sont censés être hasher comme le password)

De plus l’utilisateur doit confirmer son compte par mail,c'est fait par SMTP:
J'ai ajouté cette dépendance avec maven.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

Puis j'ai modifié le fichier application.properties

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=test02@gmail.com
spring.mail.password=XXXX XXXX XXXX
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Enfin, je peut créer une classe où, l'on peut personnaliser et envoyer le mail avec [JavaMailSender](#)

```
SimpleMailMessage message = new SimpleMailMessage();
message.setTo(email);
message.setSubject("Your Verification Code, please");

mailSender.send(message);
```

Les api utilisent un token jwt afin de sécuriser les communications, pour ce faire, j'ai dans un premier temps bloquer les requêtes qui ne sont pas authentifier, sauf celle dont tous peut y avoir accès.

```
.requestMatchers("/api/user/connection").permitAll()
.requestMatchers("/api/user/verify").permitAll()
    .anyRequest().authenticated()
```

Spring security nous donne la possibilité d'implanter UserDetails, qui nous donne le fait d'avoir des rôles à notre table user.

```
public class User implements UserDetails {  
  
    @Override  
    public Collection<? extends GrantedAuthority> getAuthorities() {  
        List<GrantedAuthority> grantedAuthorities = new ArrayList<>();  
        for (RoleEnum roleEnum : roles) {  
            GrantedAuthority authority = new SimpleGrantedAuthority(roleEnum.toString());  
            grantedAuthorities.add(authority);  
        }  
        return grantedAuthorities;  
    }  
}
```

```
@PostMapping("publier")  
@PreAuthorize("hasAuthority('USER')")  
public ResponseEntity Publication(@RequestBody Course course,  
Authentication authentication) {
```

Ainsi côté contrôleur, on peut filtrer les requêtes via le rôle, pour cela il faut qu'il soit authentifié. De plus, on peut également récupérer le nom de l'utilisateur authentifié, simplifiant le code pour le retrouver.

```
authentication.getName();
```

Pour conclure l'api sql, communique avec réact, qui sert d'intermédiaire entre l'api sql et l'api mongo.

Pour ce faire, lorsque un utilisateur se connecte à l'api sql, elle renvoie dans un token les identifiants et mot de passe (hashé) de l'api mongo, pour que réact puisse faire une demande de connexion.

Lors de la publication, j'enregistre juste l'id de la structure mongo et les informations ajoutées par l'utilisateur, titre, image ...

Puis en parallèle, côté mongo, je passe à true le champ isPublic.

L'administrateur à certe peut de pouvoir (n'intervient pas sur la partie mongo). Mais étant donné que les utilisateurs sont associés à leurs publications, il est plus facile de savoir qui à construit quoi, et donc de bannir ou interdire l'accès à ceux qui ne respectent pas les règles politiques d'utilisation du site.

5.2.5) HTTPS

Afin de rendre mon application plus sécurisée, il est mieux d'utiliser des requêtes https, car en plus de chiffrer les informations, l'https permet également de certifier l'application.

Pour ce faire j'ai utilisé Keytool pour générer un certificat ssl. Puis je l'ai ajouté sur spring.

```
server.port=8443
server.ssl.key-store=classpath:mykeystore.jks
server.ssl.key-store-password=xxx
server.ssl.key-password=xxx
server.ssl.key-alias=mycert
```

6.1.1) Serveur Minecraft

Etant donné que Minecraft est un jeu open source, il est assez simple de trouver de quoi créer un serveur(.jar) sur internet ou le site officiel du jeu.

Ce .jar va créer tous les dossiers et ressources nécessaires au serveur, mais avant il faut accepter les conditions d'utilisation du serveur en passant la variable EULA à true, du fichier eula.txt.

```
#By changing the setting below to TRUE you are indicating your agreement to our
EULA (https://aka.ms/MinecraftEULA).
#Tue Dec 10 16:48:02 CET 2024
eula=true
```

on à juste à exécuter cette commande sur terminal ou un .bat :

```
java -Xms4G -Xmx4G -XX:+UseG1GC -jar paper.jar nogui
pause
```

Avec les arguments, on peut choisir les ressources qui seront allouées au serveur.

Pour ma part j'ai choisie un serveur paper, il ajoute un dossier plugins ce qui me permet d'utiliser du code java.

plugins	19/02/2025 12:16	Dossier de fichiers
versions	10/12/2024 16:47	Dossier de fichiers
world	19/02/2025 12:20	Dossier de fichiers
world_nether	19/02/2025 12:20	Dossier de fichiers
world_the_end	19/02/2025 12:20	Dossier de fichiers
.console_history	19/02/2025 09:59	Fichier CONSOLE_... 1 Ko
banned-ip.json	19/02/2025 12:17	Fichier JSON 1 Ko
banned-players.json	19/02/2025 12:17	Fichier JSON 1 Ko
bukkit.yml	19/02/2025 12:17	Fichier source Yaml 1 Ko
commands.yml	19/02/2025 12:17	Fichier source Yaml 1 Ko
eula.txt	10/12/2024 16:48	Document texte 1 Ko
help.yml	10/12/2024 16:49	Fichier source Yaml 0 Ko
ops.json	19/02/2025 12:17	Fichier JSON 1 Ko
paper.jar	10/12/2024 16:43	Executable Jar File 49721 Ko
permissions.yml	10/12/2024 16:49	Fichier source Yaml 0 Ko
runing.bat	10/12/2024 16:47	Fichier de comma... 1 Ko

On peut modifier les .json et .yml pour personnaliser, sécuriser son serveur.

J'utilise un framework nommé Spigot, ce framework permet d'interagir avec quasi tout les éléments et objets du jeu, pour personnaliser un maximum l'expérience de jeu.
Une fois compilé le projet Spigot génère un .Jar, que l'on mettra dans le dossier plugins du serveur.

Spigot permet de:

- avoir des listeners
- persister des données
- utilisé des dépendances maven en jeux

Les listeners permettent d'exécuter du code à après un événement particulier (PlayerJoinEvent, PlaceBlockEvent), par exemple j'utilise le PlaceBlockEvent pour récupérer les coordonnées et le type du bloc posé, afin de le mettre dans un tableau.

```
@EventHandler
public void joinPlayer(PlayerJoinEvent event) {
    event.getPlayer().sendMessage(ChatColor.RED + "Salut");
    System.out.println("Player : "+event.getPlayer().getName());
    players.put(event.getPlayer().getName(), new
    PlayerS(event.getPlayer()));
    try {
        persiste(event.getPlayer());
    } catch (Exception e) {
        System.err.println("bad request on join: "+e);
    }
}
```

La persistance des données se fait avec une librairie qu'utilise spigot : org.bukkit. Tous les blocs, éléments, entités du jeu peuvent persister. Mais dans ce projet je l'utilise uniquement sur les joueurs, de la façon suivante.

Si un joueur n'est pas persisté, je persiste avec un code généré aléatoirement. Ce code je le mets dans l'inventaire du joueur pour que uniquement lui y ait accès. Enfin, j'envoie à l'api mongo son nom de joueur et le code généré.

Le plugin reste avant tout un projet java qui utilise des librairies, donc il est totalement possible d'ajouter d'autres librairies. Personnellement, j'ai utilisé maven pour gérer les dépendances. Et j'ai ajouté Lombok, pour la rédaction du code, et jackson, pour utiliser du JSON. Le Json me permet d'envoyer à l'api mongo les données dont elle a besoin via une simple requête https.

```
System.out.println(requestBody);
HttpRequest request = HttpRequest.newBuilder()
    .POST(HttpRequest.BodyPublishers.ofString(requestBody))
    .uri(URI.create("https://127.0.0.1:8080/" + uri))
    .setHeader("Content-Type", "application/json")
    .setHeader("Authorization", token)
    .build();

HttpResponse<String> response = null;
try {
```

La sécurité était, durant un moment, faible sur les serveurs minecraft, puisque tout le monde a accès au code sources et peuvent trouver des failles. Mais aujourd'hui, grâce aux dev et à la communauté, les serveurs sont désormais beaucoup plus sécurisés. On peut ajouter plusieurs mod ou plugins comme (LuckPerms) qui permet de limiter les accès au joueurs (interdire les commandes administrateurs) et de détecter les joueurs malicieux.

```
[10:40:05 INFO]: [SOFS] Enabling SOFS v1.0-s
[10:40:05 INFO]: [SOFS] [STDOUT] Plugin is active !
[10:40:05 WARN]: Nag author(s): '[]' of 'SOFS v1.0-s' about their usage of System.out/err.print. Please use your plugin's logger instead (JavaPlugin#getLogger).
[10:40:05 INFO]: [SOFS] [STDOUT] k
[10:40:05 INFO]: [SOFS] [STDOUT] {"password":"Java", "name":"123"}
[10:40:06 INFO]: 200
[10:40:06 INFO]: [SOFS] [STDOUT] eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiIxMjMiLCJpYXQiOjE3NDAzMzM2MDYsImV4cCI6MTc0MDMwNzIwNn0.ejvg-wSgjvTnNkJA_wzxgXd8a-jRNogsFCqo0TZ3rs
[10:40:06 INFO]: 200
[10:40:06 INFO]: [SOFS] [STDOUT] ["123", ".GreyCactus67220BackTest", "tds_lpaTest", "tesl123456", "oemgl", "tds_lpa"]
[10:40:06 INFO]: ["123", ".GreyCactus67220BackTest", "tds_lpaTest", "tesl123456", "oemgl", "tds_lpa"]
[10:40:06 INFO]: [SOFS] [STDOUT] ".GreyCactus67220BackTest"
[10:40:06 INFO]: [spark] Starting background profiler...
[10:40:06 INFO]: [spark] The async-profiler engine is not supported for your os/arch (windows11/amd64), so the built-in Java engine will be used instead.
[10:40:06 INFO]: Done preparing level "world" (2.955s)
[10:40:06 INFO]: Running delayed init tasks
[10:40:06 INFO]: Done (16.861s)! For help, type "help"
>
```

Par exemple ici sur ce serveur test, au démarrage il fait une requête d'authentification à l'api mongo, pour récupérer le token de l'utilisateur type "Serveur", afin de faire les autres requêtes sans souci.

De plus, la plupart des services d'hébergement proposent directement des serveurs Minecraft sécurisés et encadrés. Il est également possible d'utiliser des plugins pour auto-héberger son serveur sur sa machine, ou utiliser des images docker de serveur minecraft. Or avec ces moyens il est nécessaire de configurer la sécurité et d'implémenter manuellement, des moyens d'intégration et de déploiement continue. De nos jours, ce genre de système avec une approche ci/cd, sont déjà disponibles dans certaines offres d'hébergement de serveur minecraft.

7.1)Docker

Afin de sécuriser, et déployé plus facilement mon application via Docker et son système de conteneurisation, et jenkins pour son lien étroit avec git. Cela permet également de rendre l'application portable et d'avoir un environnement de travail fixé.

Bien que ces outils soient bien pour la CI/CD

Pour ma part j'ai conteneurisé d'un coté l'api mongo et sa DB, d'un autre, j'ai utilisé docker-compose pour créer 3 conteneur dans le même et ouvert uniquement le port de l'application réact, sql, et l'api sql les faisant communiquer via le réseau docker.

<input type="checkbox"/>	<input checked="" type="radio"/>	dev	-	-
<input type="checkbox"/>	<input checked="" type="radio"/>	backend_mc-1	fa67bc79ad80	dev-backend_mc
<input type="checkbox"/>	<input checked="" type="radio"/>	web_affiche-1	db680429c347	dev-web_affiche

```
services:
  web_affiche:
    build:
      context: ./webAffiche
      dockerfile: ./DockerFile.yml
    restart: always
    ports:
      - "5173:5173"
    volumes:
      - ./data_pack:/src
    environment:
      - VITE_BACKEND_ADDRESS=http://localhost:8080
  networks:
    - app-network

  backend_sql:
    build:
      context: ./backend
      dockerfile: ./DockerFile.yml
    depends_on:
      - database
    ports:
      - "8080:8080"
    networks:
      - app-network
    restart: always

networks:
```

7.2) Jenkins

pour automatiser les prochaines mises à jour et compiler les tests unitaires (avec JUnit). J'ai créé un conteneur avec jenkins qui va procéder à la récupération du code sur git et tester les tests en place.

Enfin, afin d'avoir un code de qualité et documenté, j'ai en plus utilisé l'extension de sonarqube sur mon IDE.

The screenshot shows two sections of a Jenkins configuration page:

- Invoquer les cibles Maven de haut niveau**:
 - Version de Maven: mvn
 - Cibles Maven: mvn --version
 - Avancé
- Exécuter un script shell**:
 - Commande:
Voir [la liste des variables d'environnement disponibles](#)
 - cd backend
ls
.mvnw clean install