Note uniksimWeb

Les versions utilisées sur le site :

- Node.js v.22.5.1
- Angular v.18.2.11 (FrontEnd)
- NestJs v.11.0.11 (BackEnd)

Angular / NodeJs compatibilité : Lien

Changer de version avec nvm (Node Version Manager) : `nvm use *version`
Voir la liste sur nvm (Node Version Manager) : `nvm list`
Voir les packages outdated : `npm outdated`
Mettre à jour les packages (classique) : `npm update` *cela update redis!
Mettre à jour les packages (avec npm-check-updates) : `npx npm-check-updates -u` *cela update redis!

Node version : `node -v`
Angular version : `ng version`
NestJs version : `nest --version`
*npm-check-updates (backendEnd) : `npx npm-check-updates`

> Note : Le fichier package.json possède toutes les versions
>
> Note : Ne pas oublier de faire un `npm install` dans backendEnd et frontEnd

Les imports importants NestJs :

- TypeOrmModule : Communication et conversion des données avec la mySql (Query)
- DatabaseProviders : Connection avec mySql
- Guard : Authentification et accès
- MailGun : Système de messagerie
- Generate Password TS : Système de génération de mot de passe (mot de passe oublié)

Pour plus de détails voir : **app.module.ts** ou **package.json**

Les roles :

- user:
  - Permission de jouer (Games uniquement)
- user_uniksim:
  - Permission d'accès au site
  - Permission de jouer
- admin_user:
  - Permission d'accès au site
  - Permission de jouer
  - Permission avec accès limité à la page admin
- admin:
  - Toute permission

*note: Il existe un role superAdmin, mais cela n'est pas vraiment utilisé...

DigitalOcean (Update branch) :

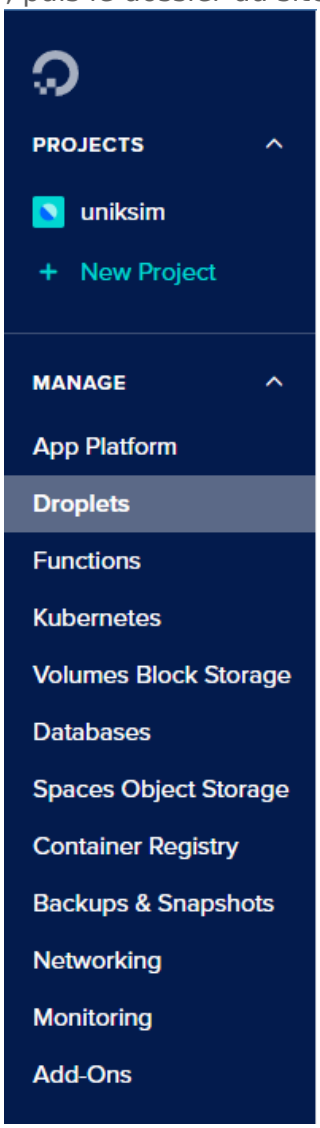
Si vous avez accès à DigitalOcean, vous pouvez faire des changements sur le site en version live* (Principalement pour des updates en Git)

**Ici "live" peut-être un site de développement avec un domain, pas forcément le vrai site*

***n'oublie pas de builder vos channgements avant d'update! (frontend surtout)*

1. Dans la section, **Droplets**

, puis le dossier du site en question



2. Ensuite dans **Console**

   Console: ?

   - Vous devriez voir une console en linux pour faire les modifications

3.

Naviguer à travers les fichiers pour trouver le dossier recherché

- Note 1: Les commandes git se font généralement sur le fichier source (Dans ce cas **sim.uniksim.com**)
- Note 2: Durant un `npm install` sur la console de Droplet, il faudrait ajouter `--legacy-peer-deps` à la fin. (la version de Droplet est plus récente)
  - Note 2.1: Un `npm audit fix` pourrait être intéressant pour fix les soucis de package conflict. [Autre Solution Non recommandé] Pour être aussi `npm audit fix --force`, cependant ceci risque de brisser des packages.
- Note 3: Il se peut que le dossier source **root** n'est pas le dossier recherché, puisque votre **username** de base de donnée (database) n'est pas le même.

```
PROVIDERS: [
    "uniksim"
],
HOSTNAME: [
    "127.0.0.1"
],
USERNAME: [
    "root",
    "forge",
],
PORTS: [
    8095, // Live
    8096, // Dev
    8097, // Demo
    3306, // Local
],
```

  - Faire la commande suivante si c'est le cas (**forge** fait référence au **username**, changer selon le cas)
    - ```
      su - forge
      ```

4. Utiliser les commandes nécessaire à vos changements
      ```

- Remettre à l'état d'origine le dossier
  - ```
    git reset .
    ```
  - ```
    git reset --hard
    ```
    - `git reset --hard` risque de nécessiter `origin/master` comme paramètre
- Récupérer la branch commit
  - ```
    git pull
    ```
- Aller vers la branche
  - ```
    git checkout
    ```
- Voir l'état de la branche actuel
  - ```
    git status
    ```
- Similaire à reset, elle permet de retirer les dossiers non nécessaires
  - ```
    git clean -fd
    ```

5. Faire `cd uniksim-back-end`, puis faire `npm run build` pour appliquer les changements
6. (optionnel) Faire `npm run start` pour lancer le site. *n'oublie de faire de même pour admin_server si nécessaire

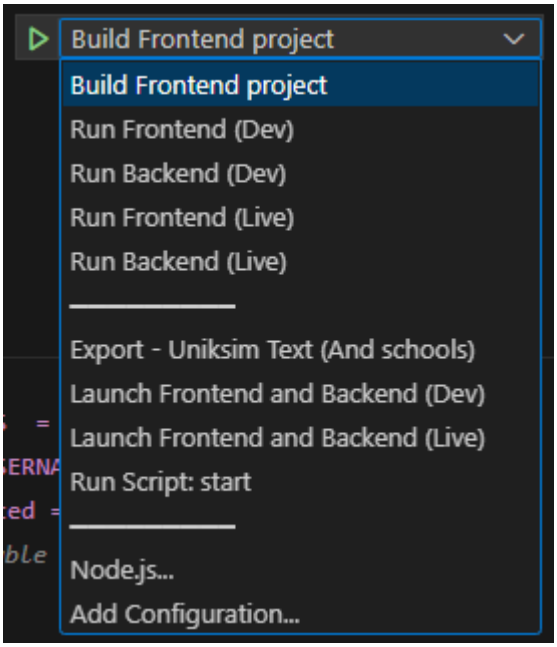
Cookieyes :

Url : https://app.cookieyes.com/

Script url : https://cdn-cookieyes.com/client_data/b08568c123e8efb0b38c9ce6/script.js

Build :

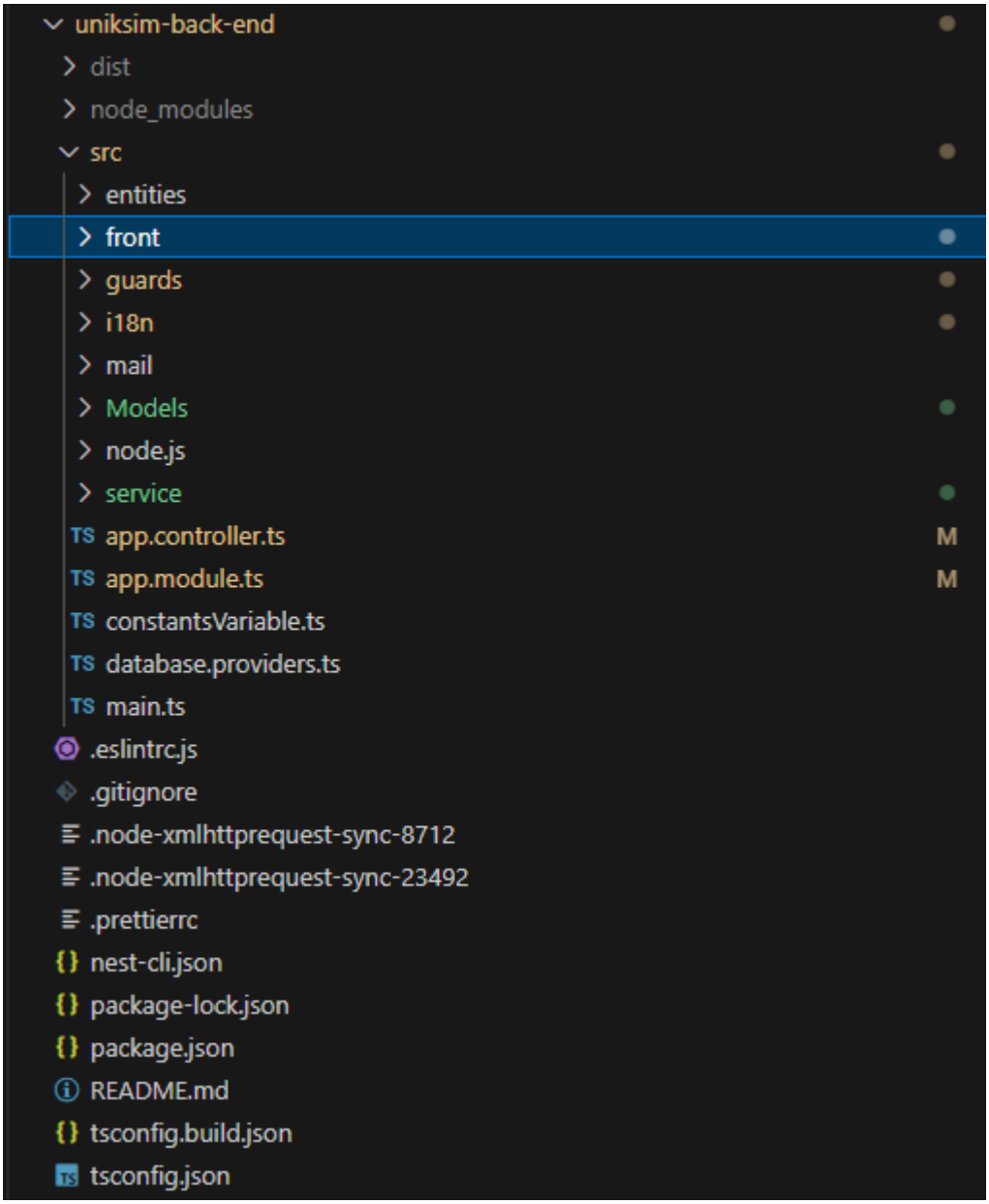
- Pour les build, utiliser la commande **Build Frontend project**



- Si cela n'est pas possible, utiliser la commande ci-dessous dans un terminal (dans le dossier uniksimFrontEnd)

```
ng build
```

- Vous devriez avoir créer un build nommer **Front** dans le dossier **uniksim-back-end**



- Par la suite **ServeStaticModule** (Permet de rendre le build static et disponible pour le serveur) va pourvoir recréer ce dossier à l'intérieur de **dist**, lorsque le serveur sera en marche.
- Note: le fichier **nest-cli.json** est important pour importer tous les fichiers du dossier **front** généré. Spécifiquement la ligne :

`{ "include": "front/**/*"}`

uniksim-back-end > {} nest-cli.json > ...

```json
{
  "$schema": "https://json.schemastore.org/nest-cli",
  "collection": "@nestjs/schematics",
  "sourceRoot": "src",
  "compilerOptions": {
    "deleteOutDir": true,
    "assets": [
      { "include": "i18n/**/*", "watchAssets": true },
      { "include": "front/**/*"}
    ]
  }
}
```

- Note: si vous changez unity build (ex: Build.data.unityweb), s'assurer de mettre la même version dans **constantsVariable**

```
// Constant set in the server (Some values is set at main.ts, to check if the server is running in local)
export const ConstantsInUse = {
    PROVIDERS: ConstantsVariable.PROVIDERS[0],
    HOSTNAME: "",
    PORTS: ConstantsVariable.PORTS[3],
    USERNAME: "",
    PASSWORD_DATABASE: "",
    SITE_URL: "",
    DEFAULT_SERVER_INSTANCE: "",
    LOCAL_WEBGL_VERSION: 505, // Please change this value after building a new version in local
}
```

Notes :

- Dans **database.providers.ts**, "synchronize" permet de faire en sorte que les entitées de node.js (typeOrm) sont créer ou modifier sur Sql. Il est donc préférable de le laisser **false**
- Si vous voulez augmenter la taille maximum pour le phpmyadmin suivre ce post:

  > https://stackoverflow.com/questions/3958615/import-file-size-limit-in-phpmyadmin

- La commande pour envoyer le serveur backend en local (il faut bien sûr "cd" dans le bon dossier) :

  > npm run start

  - Version en live/demo :

    > npm run start:live

    - Cette commande nécessite le build `npm run build`

    -

Celui du frontend est :

```
ng serve
```

- Pour debug le code du frontend (Angular), il faudra utiliser l'inspecteur du navigateur (Dans source)
- Accès au SQL :

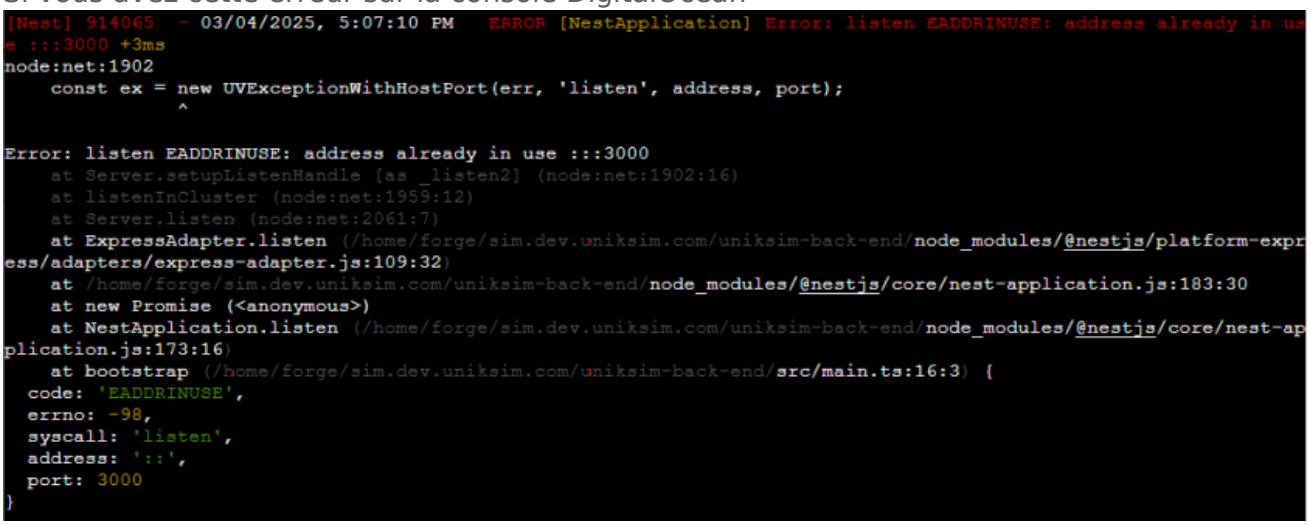
  https://db.uniksim.com/phpmyadmin/sql.php?server=1&db=uniksim&table=constants&pos=0
- La commande pour lancer admin_server (après su- forge) :

  ```
  node sim.dev.uniksim.com/uniksim-back-end/src/node.js/server/admin_server.js
  ```

  - Il est possible d'ajouter un paramètre : `node admin_server.js dev`

Problèmes :

- Si les modifications appliqués sur le dossier **server** ne sont pas appliqués, vous avez probablement compilé le script sur "dist/../"
- Si vous avez un problème de **ENOENT: no such file or directory, scandir** sur le serveur dev (sim.dev.uniksim), il y a probablement un problème de version dans les packages. Une solution serait de refaire les fichiers non tracés par git (supprimer et installer)
- Si vous avez cette erreur sur la console DigitalOcean



  - Faite `sudo lsof -i :3000` pour trouver le node en cours
  - `~hFHs>!SLx3)d4Yc_HbQ` si nécessaire
  - Puis `kill -9 <PID>` pour le terminer
  - Refaire votre commande

Les références de création :

- Site de NestJs : https://docs.nestjs.com/first-steps
- Site de compatibilité de version Angular/Node : https://v17.angular.io/guide/versions
- Léger tread de discussion sur les version NestJs/Node :

  https://stackoverflow.com/questions/77160198/how-to-find-supported-node-js-version-for-

nest-js

- Configuration de NestJs/MySQL (phpMyAdmin) : https://blog.stackademic.com/connecting-nestjs-to-mysql-a-step-by-step-guide-ab4086baf918
  - Other alternative : https://medium.com/@yatin.kumar07/integrate-nestjs-with-typeorm-and-mysql-68ad4c10f6ff
- Configuration de NestJs : https://medium.com/@anteprocess/%EF%B8%8Fnestjs-for-beginners-part-1-overview-and-project-setup-launch-33dc7d8071e9
- Imports of node.js (Javascript files) : https://medium.com/@eladk/moving-from-javascript-to-typescript-lifehacks-e29d7e1aa3e0
- Decoration pour les expressions Sql : https://typeorm.io/decorator-reference#column-decorators
- Typeorm wiki (NestJs/Sql data communication) : https://orkhan.gitbook.io/typeorm/docs/find-options
- Debugger for NestJs : https://dev.to/gentax/nestjs-right-settings-for-debugging-kl0
- bCrypt détailes post : https://stackoverflow.com/questions/15733196/where-2x-prefix-are-used-in-bcrypt/36225192#36225192
- TypeOrm relations (Many-to-one / one-to-many): https://github.com/typeorm/typeorm/blob/master/docs/many-to-one-one-to-many-relations.md
- Simple Unity/Angular intégration : https://medium.com/@6386391ritesh/unity-webgl-in-angular-686397677500
- Build le projet Angular à l'intérieur de Nestjs : https://medium.com/codeshake/deploy-an-angular-nestjs-app-to-heroku-807a2833d88c
- SSE notification : https://medium.com/@andrewkoliaka/implementing-server-sent-events-in-angular-a5e40617cb78
- Erreur commun pour nvm lorsqu'il refuse de changer de version : https://stackoverflow.com/questions/47017812/nvm-use-does-not-switch-node-versions

---

Revision #78
Created Fri, Jun 7, 2024 1:33 PM by Larry
Updated Thu, Jul 2, 2026 5:24 PM by Larry