

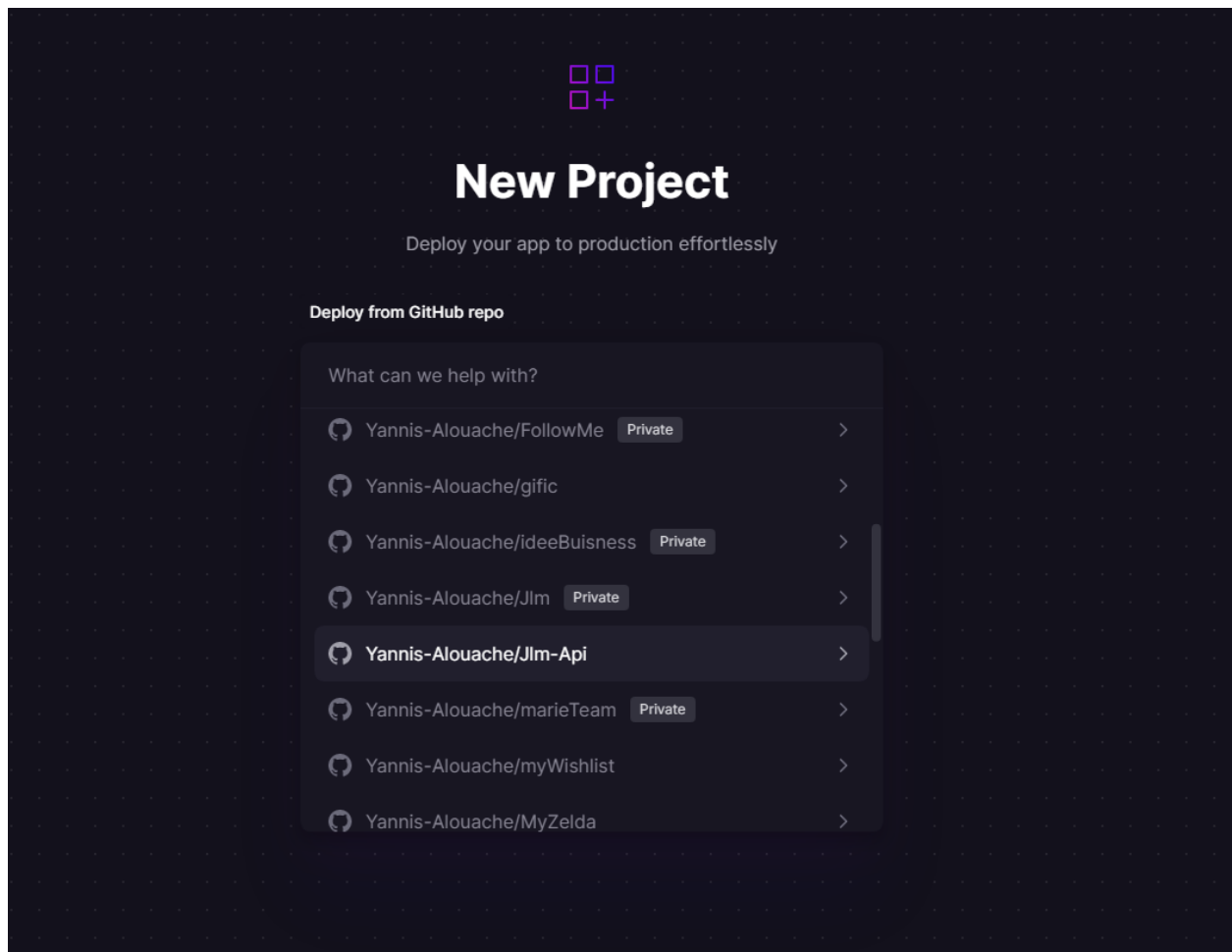


Déploiement de l'api JLM

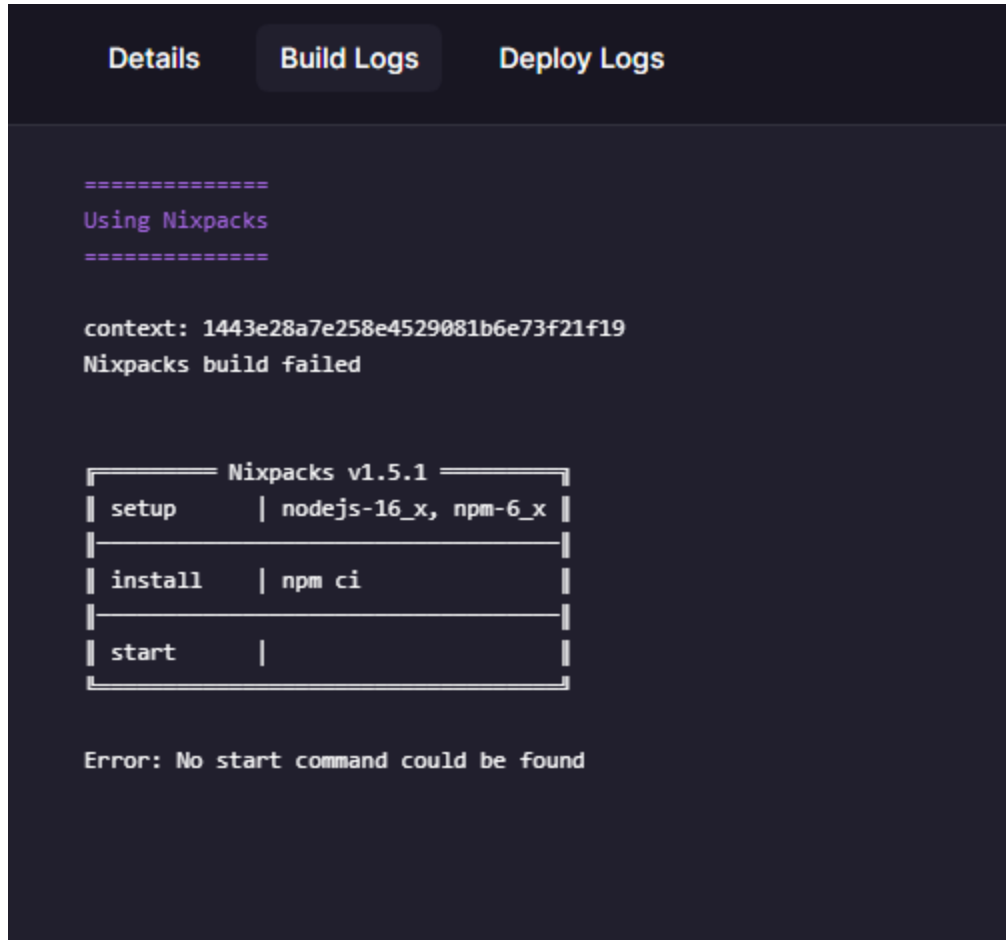
Yannis Alouache

Le Déploiement sur Railways

Pendant mon stage, j'ai utilisé le service de déploiement gratuit Railways pour déployer l'API du site web que j'ai développé. Ce service a permis un déploiement rapide et facile, en évitant les étapes fastidieuses liées à la configuration du serveur.



J'ai dû outrepasser certain problème comme le manque d'une commande de lancement de mon serveur nodeJS



The screenshot shows a deployment interface with three tabs: 'Details', 'Build Logs', and 'Deploy Logs'. The 'Build Logs' tab is active, displaying the following text:

```
=====
Using Nixpacks
=====

context: 1443e28a7e258e4529081b6e73f21f19
Nixpacks build failed
```

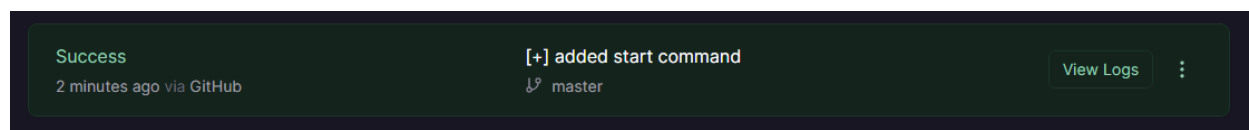
Nixpacks v1.5.1	
setup	nodejs-16_x, npm-6_x
install	npm ci
start	

```
Error: No start command could be found
```

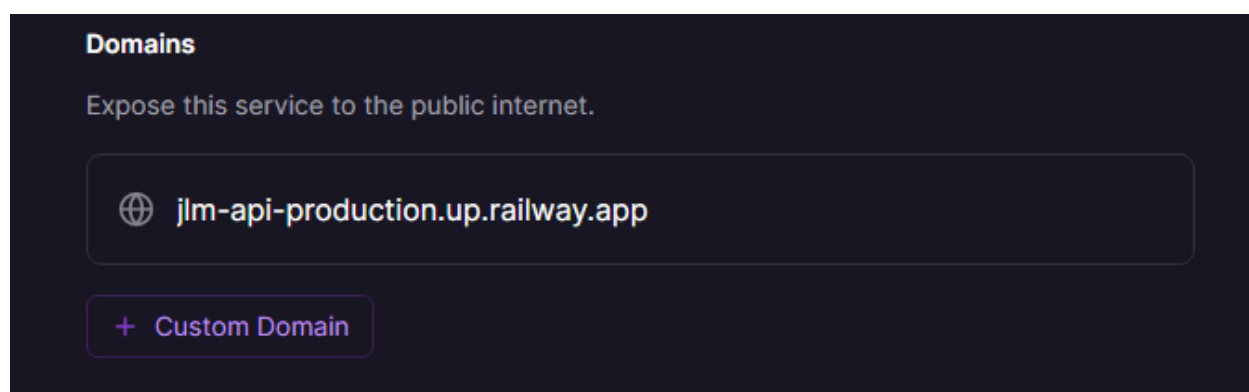
J'ai dû rajouter cette commande dans le fichier package.json qui permet de modifier les configurations et les dépendances d'un projet nodeJS.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node ./src/index.js"
},
```

Le premier déploiement à était un succès



J'ai ensuite dû ajouter un domaine pour exposé l'api sur internet



Après avoir généré le domaine je me suis rendu compte que je ne voyais rien sur l'adresse de l'api, j'ai donc décidé de rajouter une page d'accueil afin d'avoir un visuel et confirmer que le système était fonctionnel.

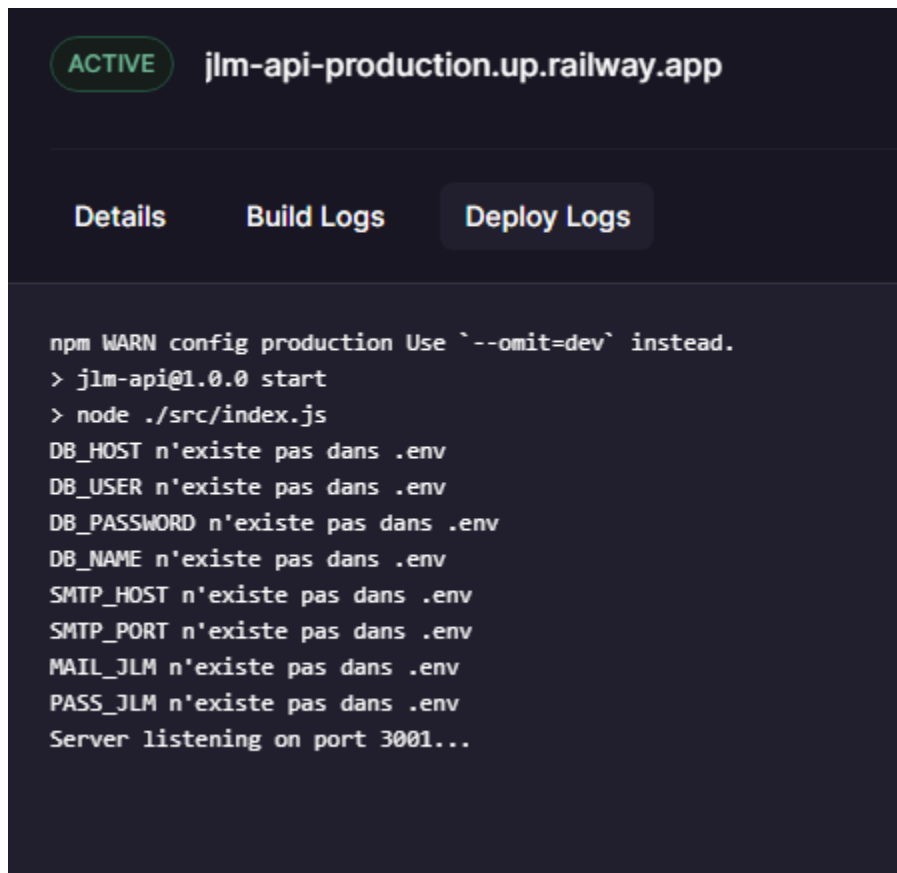
```
this.server.get('/', this.home.home.bind(this.home))
```

```
export class Home {  
  
  async home(req, res) {  
    return res.status(200).send({  
      status: "success",  
      msg: "Bienvenue sur l'api JLM"  
    })  
  }  
}
```

Après ces manipulations et une mise à jour du dépôt github on peut remarquer que l'api est accessible

```
1 // 20230319143430  
2 // https://jlm-api-production.up.railway.app/  
3  
4 {  
5   "status": "success",  
6   "msg": "Bienvenue sur l'api JLM"  
7 }
```

j'ai remarqué des erreurs dans la console de déploiement de mon API, bien que celle-ci semblait fonctionner correctement à première vue. Après avoir enquêté plus en profondeur, j'ai identifié que ces erreurs étaient dues à une classe nommée "Env" que j'avais créée.

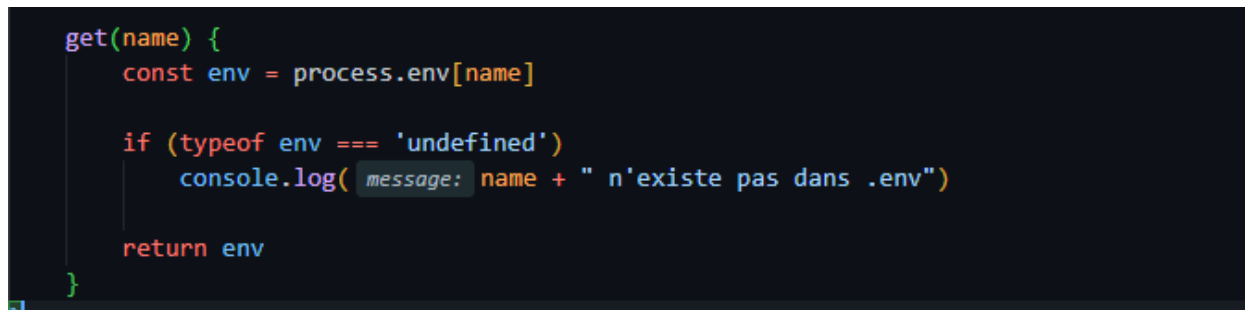


```
ACTIVE jlm-api-production.up.railway.app

Details Build Logs Deploy Logs

npm WARN config production Use `--omit=dev` instead.
> jlm-api@1.0.0 start
> node ./src/index.js
DB_HOST n'existe pas dans .env
DB_USER n'existe pas dans .env
DB_PASSWORD n'existe pas dans .env
DB_NAME n'existe pas dans .env
SMTP_HOST n'existe pas dans .env
SMTP_PORT n'existe pas dans .env
MAIL_JLM n'existe pas dans .env
PASS_JLM n'existe pas dans .env
Server listening on port 3001...
```

La source de l'erreur :



```
get(name) {
  const env = process.env[name]

  if (typeof env === 'undefined')
    console.log( message: name + " n'existe pas dans .env")

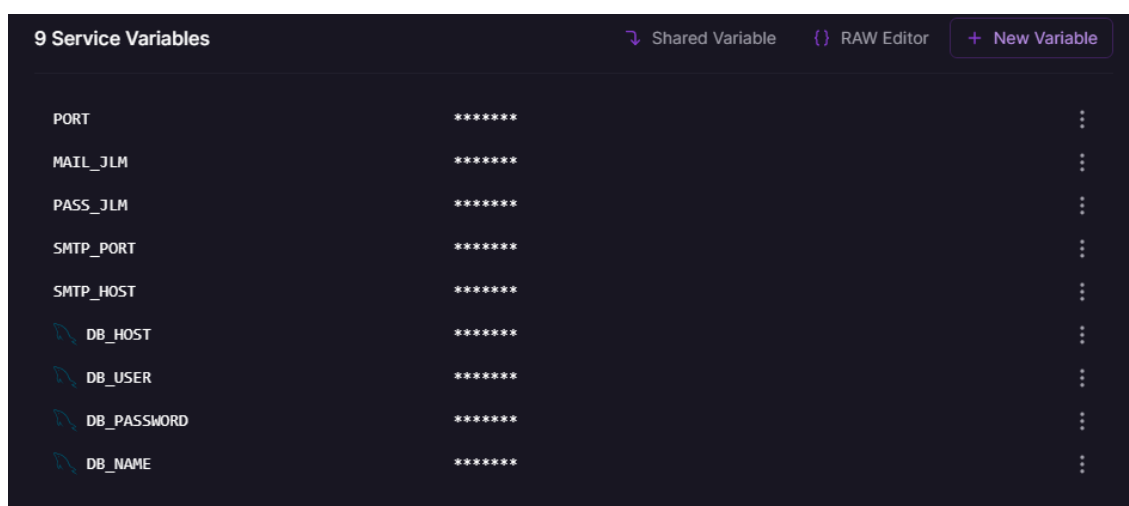
  return env
}
```

Comme nous pouvons le voir l'api n'arrive pas à trouver les informations du fichier de configuration .env, ce qui est tout à fait normal car sur un serveur nous utilisons en général les variables système et non un fichier .env

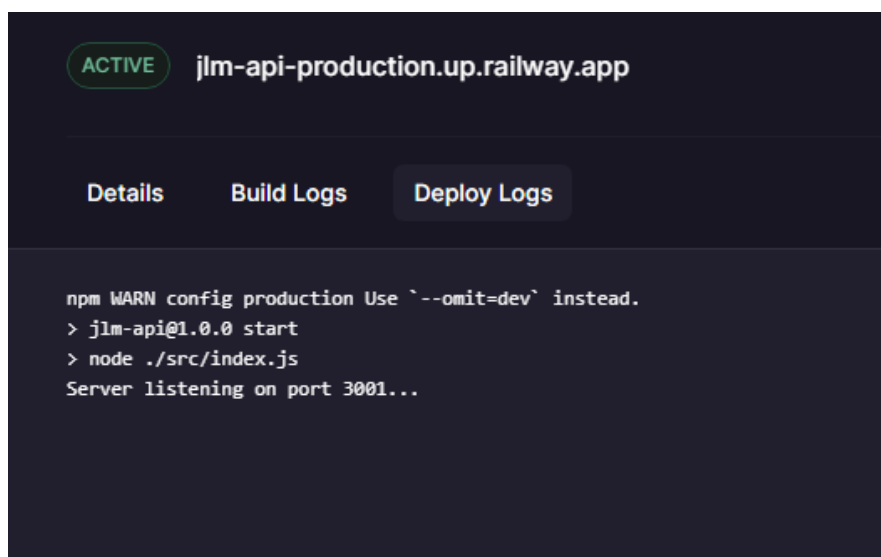
Comme je n'avais pas accès à un VPS (serveur virtuel privé) je ne pouvais utiliser les variables système.

J'ai donc investigué le site de railways jusqu'à trouver un moyen de mettre en place des variables d'environnement.

J'ai donc rempli les informations nécessaires au bon fonctionnement de l'application.

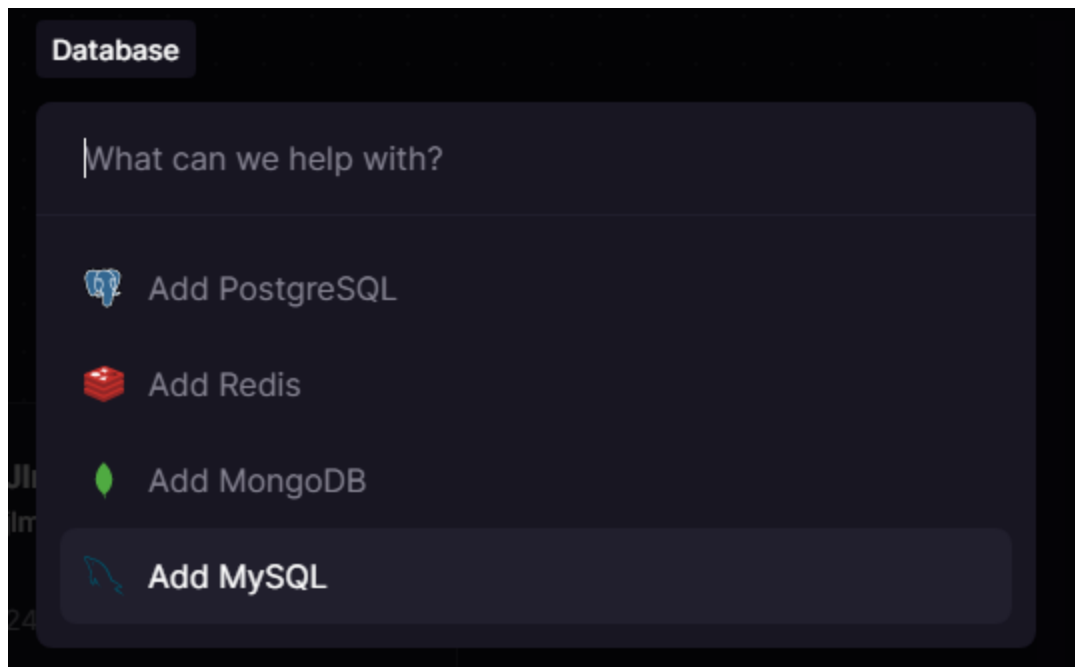


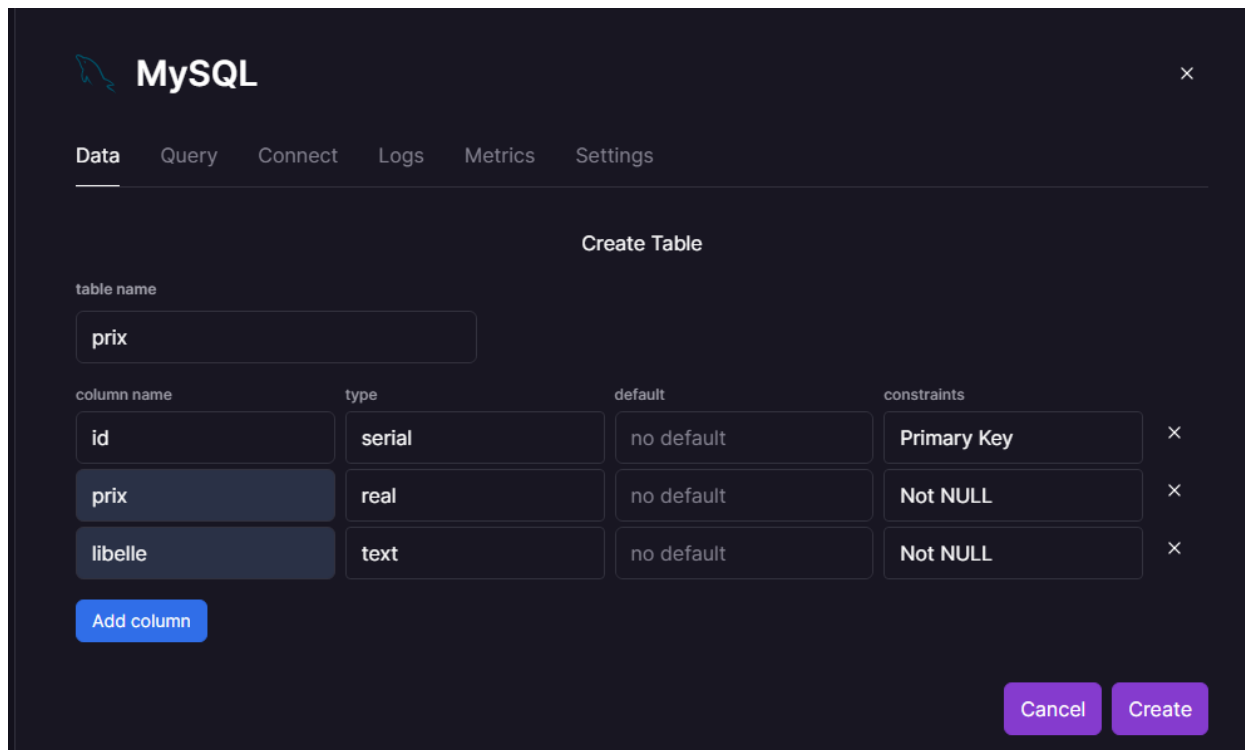
Voici la console de l'api après la mise en place des variables d'environnement :



La base de donnée

L'API que j'avais développée était fonctionnelle, mais il manquait encore la base de données MySQL qui était indispensable au bon fonctionnement du module de devis. En effet, les prix des différentes prestations étaient renseignés dans cette base de données. J'ai donc travaillé à la mise en place de cette base de données afin de permettre le bon fonctionnement de l'ensemble du site web. (encore un fois à l'aide de Railways)





The screenshot shows the MySQL 'Create Table' interface. At the top, there's a navigation bar with 'Data', 'Query', 'Connect', 'Logs', 'Metrics', and 'Settings'. The 'Data' tab is selected. Below the navigation bar, the title 'Create Table' is centered. Underneath, there's a 'table name' field containing 'prix'. Below that, there's a table with four columns: 'column name', 'type', 'default', and 'constraints'. The table contains three rows: 'id' with type 'serial' and 'Primary Key' constraint; 'prix' with type 'real' and 'Not NULL' constraint; and 'libelle' with type 'text' and 'Not NULL' constraint. Each row has a delete icon (X) to its right. Below the table is an 'Add column' button. At the bottom right, there are 'Cancel' and 'Create' buttons.

MySQL

Data Query Connect Logs Metrics Settings

Create Table

table name

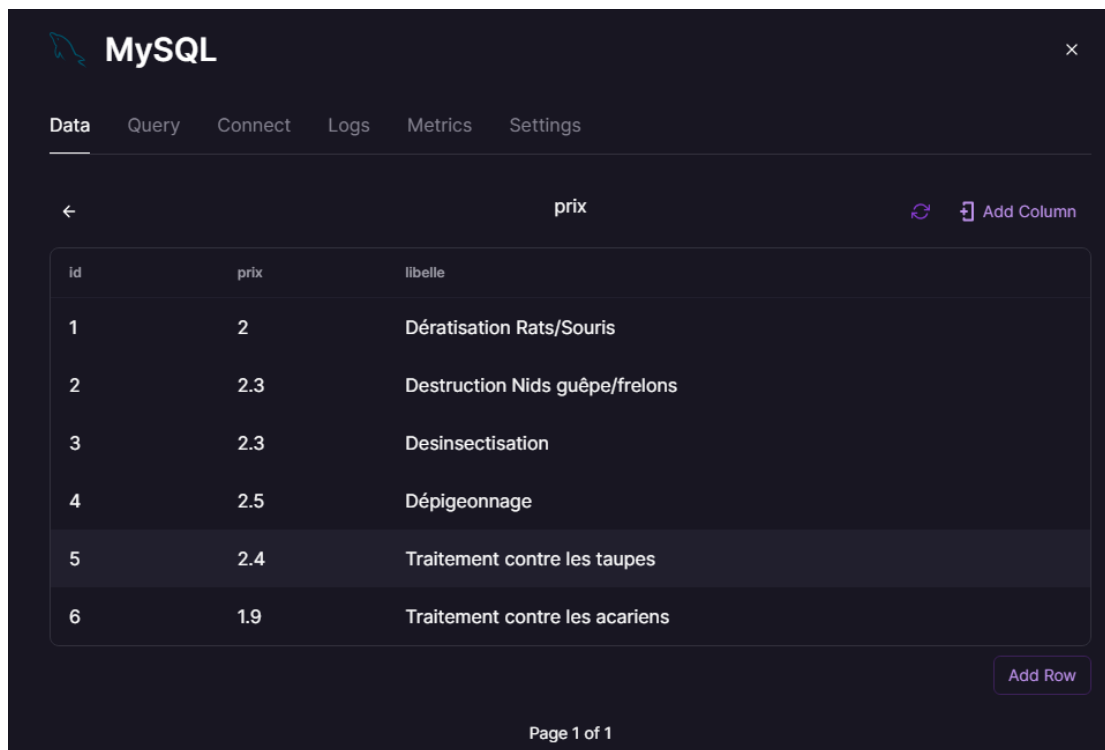
prix

column name	type	default	constraints
id	serial	no default	Primary Key
prix	real	no default	Not NULL
libelle	text	no default	Not NULL

Add column

Cancel Create

J'ai donc dû recréer la table et la remplir avec les éléments de la BDD



The screenshot shows the MySQL 'Data' view for the 'prix' table. The table has three columns: 'id', 'prix', and 'libelle'. It contains six rows of data. At the top right, there's a refresh icon and an 'Add Column' button. At the bottom right, there's an 'Add Row' button. The page number 'Page 1 of 1' is displayed at the bottom center.

MySQL

Data Query Connect Logs Metrics Settings

← prix ↻ Add Column

id	prix	libelle
1	2	Dératisation Rats/Souris
2	2.3	Destruction Nids guêpe/frelons
3	2.3	Desinsectisation
4	2.5	Dépigeonnage
5	2.4	Traitement contre les taupes
6	1.9	Traitement contre les acariens

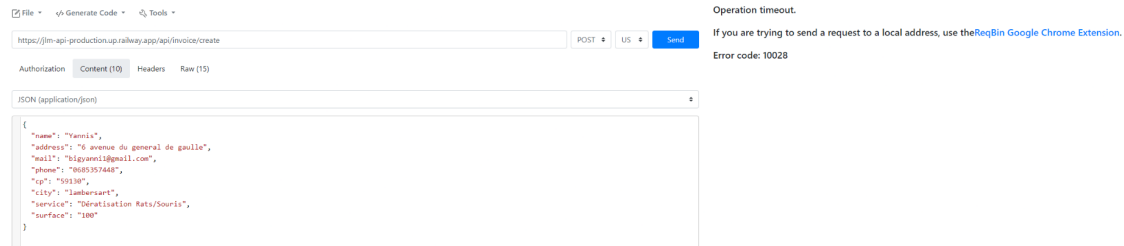
Add Row

Page 1 of 1

Après avoir remis à jour les variables d'environnement je me suis rendu compte que la création de devis n'était pas possible car j'avais un timeout lors de ma requête ce qui signifie que la requête a mis trop de temps avant de recevoir une réponse.

Online REST & SOAP API Testing Tool

ReqBin is an online API testing tool for REST and SOAP APIs. Test API endpoints by making API requests directly from your browser. Test API responses with built-in JSON and XML validators. Load test your API with hundreds of simulated concurrent connections. Generate code snippets for API automation testing frameworks. Share and discuss your API requests online.



Après des investigations pour résoudre ce problème de déploiement j'ai pu me rendre compte que la base de donnée mysql n'était pas présente sur le port par défaut de mysql qui est le 3306.



J'ai donc dû effectuer un changement dans le code de la classe DB, pour récupérer le contenu de la variable d'environnement DB_PORT.

```
import mysql from "mysql2"

export class Db {
  constructor(env) {
    this.env = env

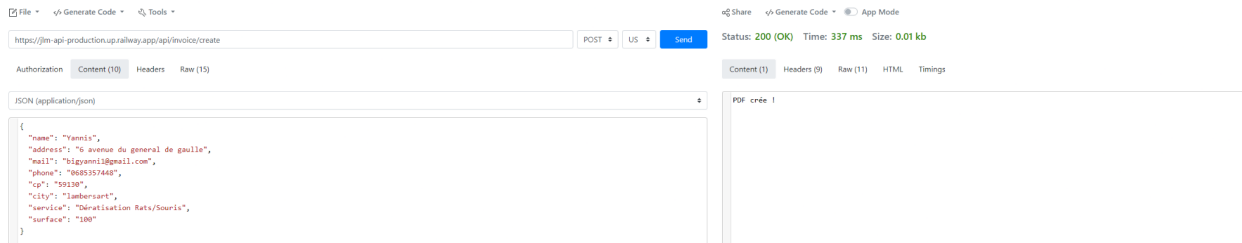
    this.pool = mysql.createPool( config: {
      host:      this.env.get("DB_HOST"),
      user:      this.env.get("DB_USER"),
      password:  this.env.get("DB_PASSWORD"),
      port:      this.env.get("DB_PORT"),
      database:  this.env.get("DB_NAME")
    }).promise()
  }
}
```



J'ai ensuite créé la variable sur mon interface Railways.

Online REST & SOAP API Testing Tool

ReqBin is an online API testing tool for REST and SOAP APIs. Test API endpoints by making API requests directly from your browser. Test API responses with built-in JSON and XML validators. Load test your API with hundreds of simulated concurrent connections. Generate code snippets for API automation testing frameworks. Share and discuss your API requests



Après un nouveau test de la création de devis je reçois un message positif de l'api.
Le déploiement de l'api est donc réussi.