



Université d'Abomey-Calavi
The Abdus Salam International Center for Theoretical Physics
INSTITUT DE MATHÉMATIQUES ET DE SCIENCES PHYSIQUES



Projet Web

Pour l'obtention d'une note de seconde devoir au semestre 2

Conception d'une application de Gestion Immobilière : Immob Porto

Etudiant:

David Duamel DOSSEH

Professeur:

Dr Sanda

Année Universitaire:2020-2021

RÉSUMÉ

On remarque la plupart du temps que les particuliers, pour trouver un logement à louer ou acheter passent par les démarcheurs ou négociateurs, et ne sont pas souvent satisfaits de leurs services. En plus, le délai de retour de ceux-ci est souvent long.

ImmobPorto vient répondre à ces problèmes en facilitant la recherche de logements que ce soit pour l'achat ou la location en répondant aux exigences des utilisateurs d'une part et en permettant à ces derniers d'y publier des annonces de demandes ou d'offres.

ACRONYME

TABLE DES MATIÈRES

1. Analyse et conception
 - a. Exigence fonctionnelles
 - b. Exigences non fonctionnelles
2. Les outils utilisés
 - a. Nest js
 - b. Passport
 - c. ElasticSearch
 - d. docker
 - e. JWT
 - f. Nodemailer
 - g. Swagger
 - h. Postgresql
 - i. Angular
 - j. Bootstrap
3. Modélisation
 - a. Diagramme des cas d'utilisation
 - b. Diagramme de Classes
 - c. Schéma logique relationnel
4. Réalisation
 - a. Postman
5. CONCLUSION

Bibliographie

1. Analyse et conception

a. Exigences fonctionnelles

- Tout internaute doit pouvoir parcourir les annonces publiées
- Rechercher une annonce
- Visiter une annonce
- Souscrire pour une annonce et prendre un rendez-vous pour la visite des lieux
- Gérer son compte
 - Modifier son profil
 - Compléter son profil
- Gérer les annonces d'offre ou de demandes
 - Publier les annonces
 - Modifier les annonces
 - Supprimer les annonces
 - Activer une annonce
 - Désactiver une annonce
- Contacter un utilisateur en cas de besoin
- Administrer les comptes utilisateurs
 - Supprimer un compte
 - Supprimer une annonce

b. Exigences non fonctionnelles

Pour mettre en place une solution adéquate aux attentes des concepteurs des architectures dynamiques, on doit prendre en considération les contraintes qui caractérisent ce système.

L'application doit nécessairement assurer les besoins suivants:

- Les informations de consultation des offres sont accessibles aux internautes mais tout autre opération n'est accessible que par authentification.
- Chaque utilisateur ne doit pas accéder à des services s'il n'a pas les permissions de ceux-ci.
- L'application doit respecter les principes d'IHM.
- L'application doit être facile à prendre en main et sécurisée.
- L'application doit être portable.

2. Les outils utilisés

★ Nest js

NestJS est un framework permettant de créer des applications côté serveur Node.js efficaces et évolutives, conçues avec et prenant en charge TypeScript. Il utilise des frameworks HTTP Server robustes comme Express ou Fastify. Nest fournit un niveau d'abstraction supérieur aux frameworks Node.js communs et expose leurs API au développeur. Cela donne une grande liberté pour utiliser des modules tiers.

Une bonne raison de choisir NestJS plutôt qu'ExpressJS (l'un des frameworks Node.js les plus populaires) est le fait que lorsqu'un nouveau projet dans Node.js est démarré, il s'agit d'une architecture claire basée sur quelques composants simples (contrôleurs, modules et fournisseurs). Cela facilite grandement la division des applications en microservices .

★ Passport

Passport est un middleware d'authentification pour Node.js. Extrêmement flexible et modulaire, Passport peut être déposé discrètement dans n'importe quelle application Web basée sur Express. Un ensemble complet de stratégies prend en charge l'authentification à l'aide d'un nom d'utilisateur et d'un mot de passe, Facebook, Twitter, etc.

★ Elasticsearch

Elasticsearch est un [logiciel](#) utilisant [Lucene](#) pour l'indexation et la recherche de données. Il fournit un [moteur de recherche](#) distribué

et [multi-entité](#) à travers une interface [REST](#). C'est un logiciel écrit en [Java](#) distribué sous licence Elastic2 ([Open core](#)). L'éditeur propose aussi une version sous Server Side Public License ainsi que la possibilité de souscrire à une offre [Saas](#).

Elasticsearch est le serveur de recherche le plus populaire chez les professionnels, suivi par [Apache Solr](#) qui utilise aussi Lucene3. Il est associé à deux autres produits libres, [Kibana](#) et [Logstash](#), qui sont respectivement un visualiseur de données et un [ETL](#) (initialement destiné aux logs).

L'indexation et la recherche des données s'effectue à partir d'une [API REST](#). Les données échangées sont au format [JSON](#).

★ Docker

Docker est un [logiciel libre](#) permettant de lancer des [applications](#) dans des conteneurs logiciels3.

Selon la firme de recherche sur l'industrie 451 Research, « Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur ». Il ne s'agit pas de [virtualisation](#), mais de [conteneurisation](#), une forme plus légère qui s'appuie sur certaines parties de la machine hôte pour son fonctionnement. Cette approche permet d'accroître la flexibilité et la [portabilité d'exécution](#) d'une application, laquelle va pouvoir tourner de façon fiable et prévisible sur une grande variété de machines hôtes, que ce soit sur la machine locale, un cloud privé ou public, une machine nue, etc.4

★ JWT

JSON Web Token (JWT) est un standard ouvert défini dans la RFC 7519¹. Il permet l'échange sécurisé de jetons (tokens) entre plusieurs parties. Cette sécurité de l'échange se traduit par la vérification de l'intégrité des données à l'aide d'une signature numérique. Elle s'effectue par l'algorithme [HMAC](#) ou [RSA](#).

★ Nodemailer

Nodemailer est un module pour les applications Node.js pour permettre l'envoi d'e-mails facile comme du gâteau. Le projet a démarré en 2010 alors qu'il n'y avait pas d'option sensée pour envoyer des e-mails, c'est aujourd'hui la solution vers laquelle la plupart des utilisateurs de Node.js se tournent par défaut.

★ Swagger

Swagger est un [langage de description d'interface](#) pour décrire les API [RESTful](#) exprimées à l'aide de [JSON](#) . Swagger est utilisé avec un ensemble d' outils [logiciels open source](#) pour concevoir, créer, documenter et utiliser [des services Web](#) RESTful . Swagger inclut une documentation automatisée, la génération de code (dans de nombreux langages de programmation) et la génération de cas de test.

★ Postgresql

PostgreSQL est un [système de gestion de base de données relationnelle](#) et [objet](#) (SGBDRO). C'est un outil [libre](#) disponible selon les termes d'une licence de type [BSD](#).

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme [MariaDB](#) et [Firebird](#)), ou propriétaires (comme [Oracle](#), [MySQL](#), [Sybase](#), [DB2](#), [Informix](#) et [Microsoft SQL Server](#)). Comme les projets libres [Apache](#) et [Linux](#), PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

★ Angular

Angular (communément appelé "Angular 2+" ou "Angular v2 et plus")^{2,3} est un [cadriciel \(framework\)](#) côté client, [open source](#), basé sur [TypeScript](#), et co-dirigé par l'équipe du projet « Angular » à [Google](#) et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de [AngularJS](#), cadriciel construit par la même équipe. Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « *Single Page Applications* » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action. Le Framework est basé sur une architecture du type [MVC](#) et permet

donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités. Un type d'architecture qui a largement fait ses preuves et qui permet une forte maintenabilité et une amélioration du travail collaboratif.

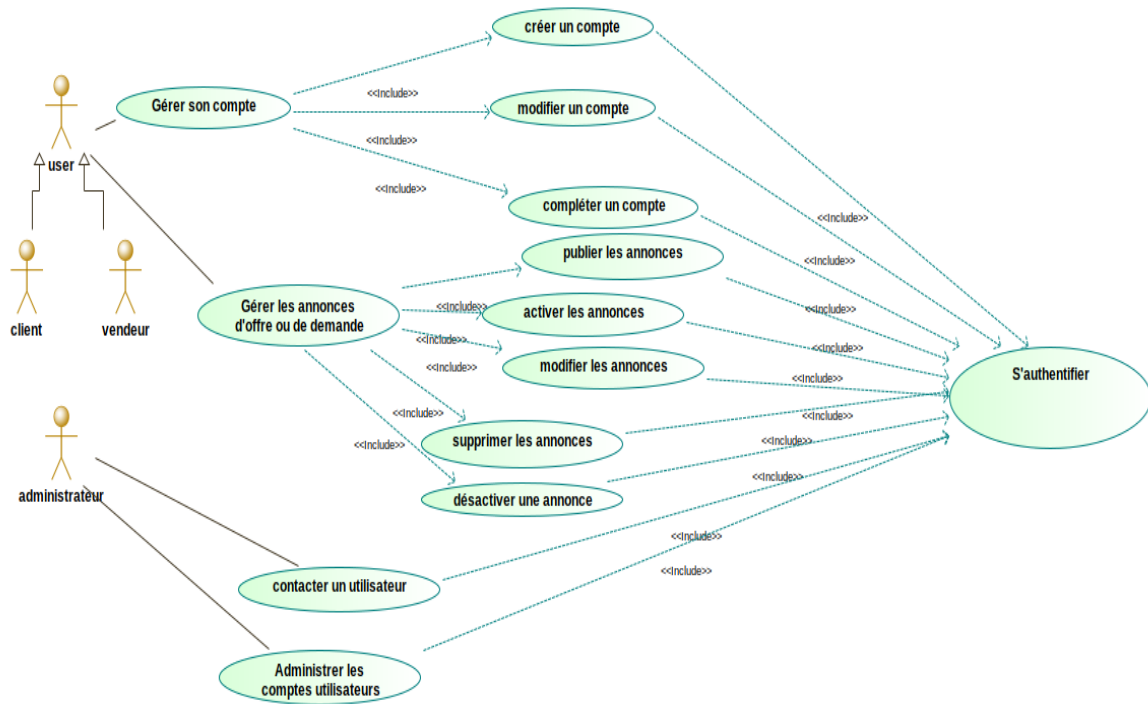
★ **Bootstrap**

Bootstrap est une [collection d'outils](#) utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de [sites](#) et d'[applications web](#). C'est un ensemble qui contient des codes [HTML](#) et [CSS](#), des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions [JavaScript](#) en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement [GitHub](#).

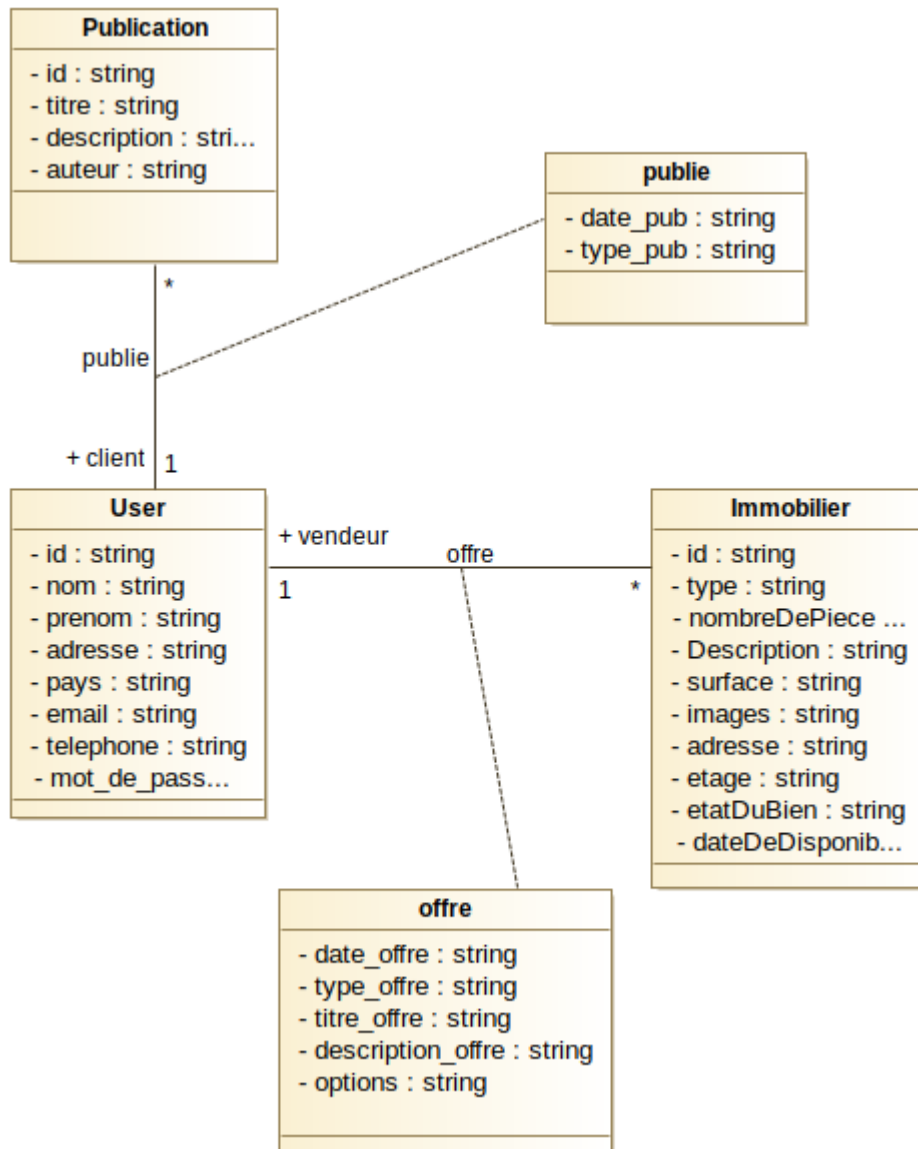
3. Modélisation

a. Diagramme de cas d'utilisation

Systeme de gestion immobilier



b. Diagramme de classe



c. Schéma logique relationnel

USER (id, nom, prenom, adresse, pays, email, telephone, mot_de_passe)





















IMMOBILIER(id, type, nombreDePiece, description, surface, images, adresse, etage, etatDuBien, dateDeDisponibilite, *date_offre*, *type_offre*, *titre_offre*, *description_offre*, *option*, *id_user*)

PUBLICATION(id, titre, description, auteur, *date_pub*, *type_pub*, *id_user*)

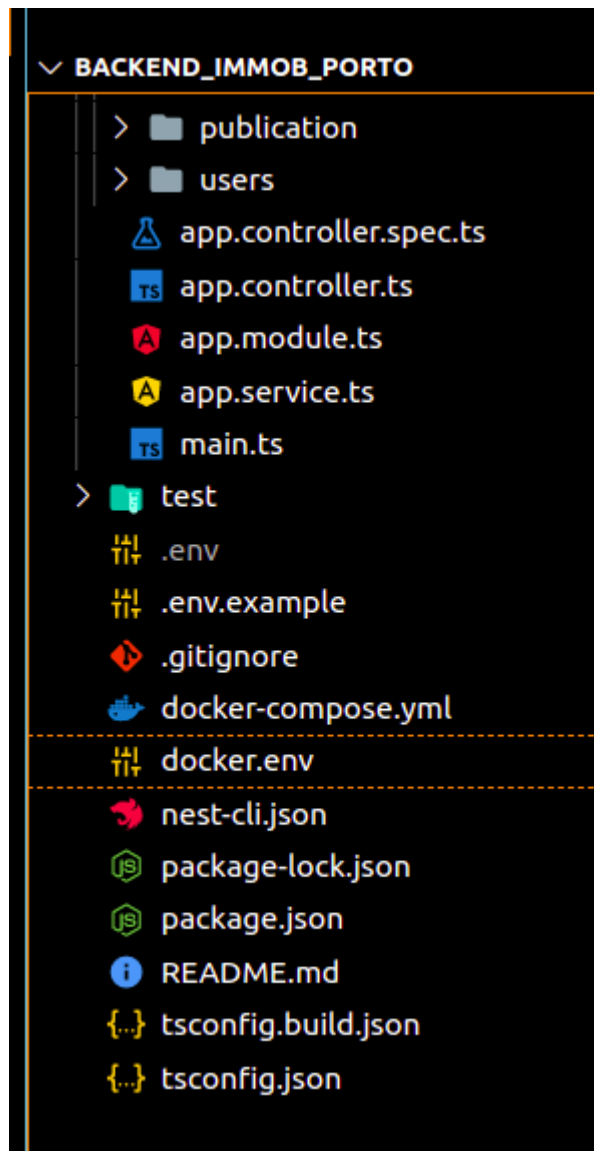
4. Réalisation

Concernant la réalisation de l'API, nous avons opté pour une architecture modulaire couplée avec le concept MVC de nest js.

▼ **BACKEND_IMMOB_PORTO**

- >  dist
- >  node_modules
- ▼  src
 - >  core
 - ▼  modules
 - >  auth
 - >  common
 - >  immobilier
 - >  publication
 - >  users
 -  app.controller.spec.ts
 -  app.controller.ts
 -  app.module.ts
 -  app.service.ts
 -  main.ts
- >  test
-  .env
-  .env.example
-  .gitignore
-  docker-compose.yml

▼ **OUTLINE**



Capture d'écran des services fonctionnels



ImmobPorto API description ^{1.0} OAS3

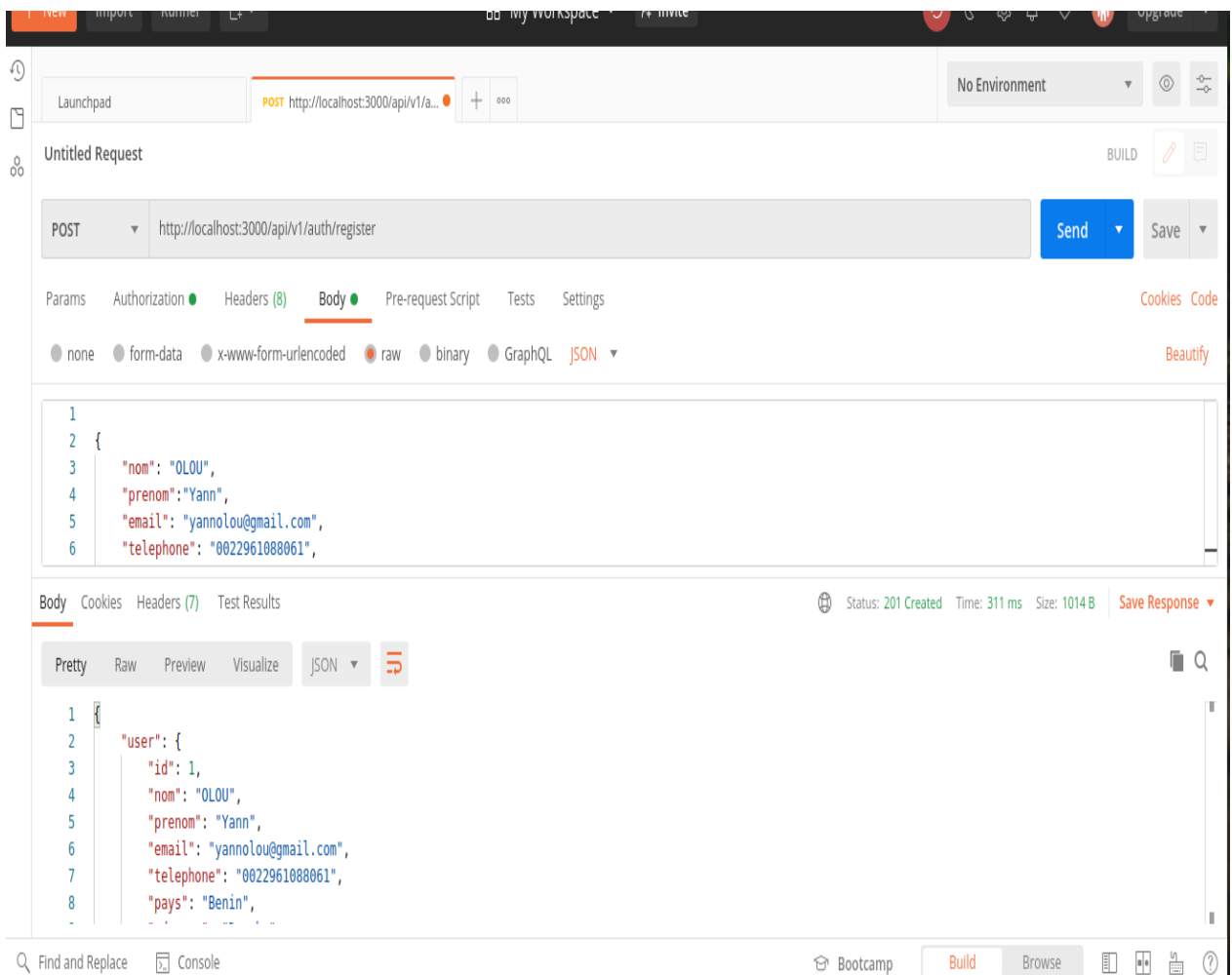
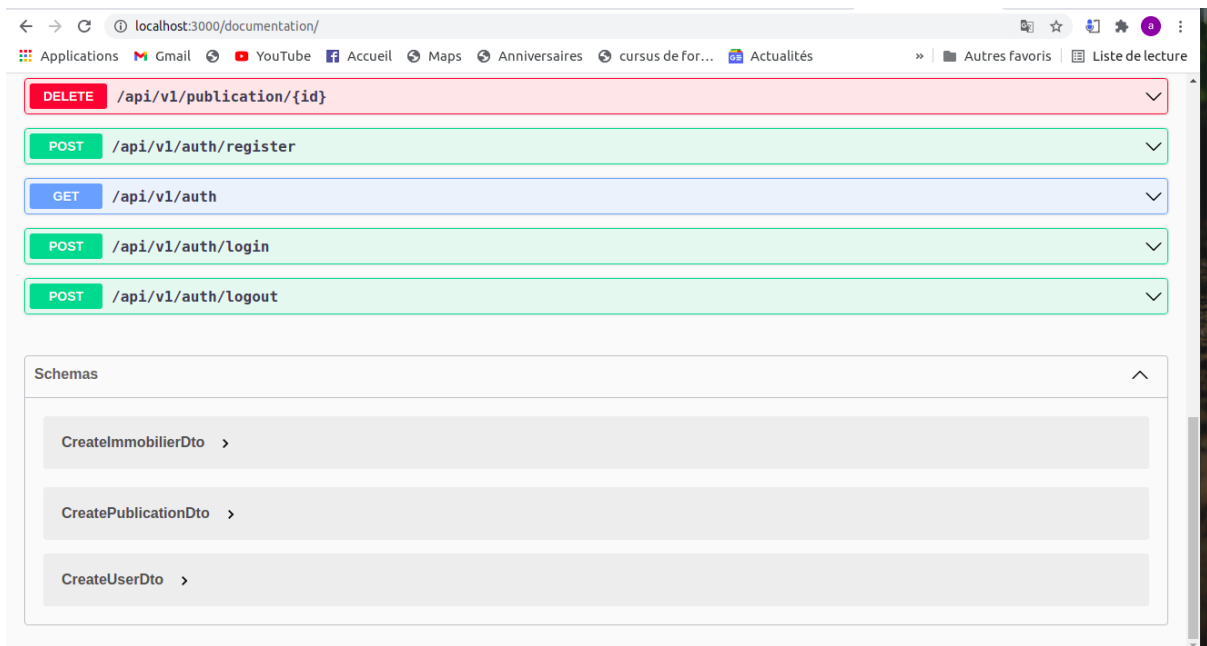
This API is the first step to build a large system

Immobilier [^]

default [^]

GET	/api/v1	▼
GET	/api/v1/users/user/{email}	▼
GET	/api/v1/immobilier	▼
POST	/api/v1/immobilier	▼
GET	/api/v1/immobilier/{id}	▼

GET	/api/v1	▼
GET	/api/v1/users/user/{email}	▼
GET	/api/v1/immobilier	▼
POST	/api/v1/immobilier	▼
GET	/api/v1/immobilier/{id}	▼
PUT	/api/v1/immobilier/{id}	▼
DELETE	/api/v1/immobilier/{id}	▼
GET	/api/v1/publication	▼
POST	/api/v1/publication	▼
GET	/api/v1/publication/{id}	▼
PUT	/api/v1/publication/{id}	▼



Launchpad

POST http://localhost:3000/api/v1/a...

No Environment

Untitled Request

POST http://localhost:3000/api/v1/auth/login

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	yannolou@gmail.com	
<input checked="" type="checkbox"/> password	yann	

Body Cookies (1) Headers (8) Test Results

Status: 200 OK Time: 147 ms Size: 708 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "isAdmin": null,
4   "isClient": null,
5   "isVendor": null,
6   "nom": "OLOU",
7   "prenom": "Yann",
8   "email": "yannolou@gmail.com",
9   "telephone": "0022961088061",
10  "nouv": "Ranin"
```

Find and Replace Console

Bootcamp Build Browse

Launchpad

POST http://localhost:3000/api/v1/p...

No Environment

Untitled Request

POST http://localhost:3000/api/v1/publication

Send Save

Params Authorization Headers Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "titre": "demande de chambre",
3   "description": "je suis à la recherche d'une chambre, un salon, cuisine et douche",
4   "date_pub": "29-10-06",
5   "type_pub": "location"
6 }
```

Body Cookies (1) Headers (7) Test Results

Status: 201 Created Time: 77 ms Size: 496 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "titre": "demande de chambre",
4   "description": "je suis à la recherche d'une chambre, un salon, cuisine et douche",
5   "date_pub": "29-10-06",
6   "type_pub": "location",
7   "clientId": 1,
8   "updatedAt": "2021-06-01T18:25:03.391Z",
9   "createdAt": "2021-06-01T10:15:03.301Z"
```

Find and Replace Console

Bootcamp Build Browse

Launchpad

GET http://localhost:3000/api/v1/publication/2

No Environment

Untitled Request

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "titre": "demande de chambre",
3   "description": "je suis à la recherche d'une chambre, un salon, cuisine et douche",
4   "date_pub": "29-10-06",
5   "type_pub": "location"
6 }
```

Body Cookies (1) Headers (7) Test Results

Status: 200 OK Time: 24 ms Size: 768 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "titre": "demande de chambre",
4   "description": "je suis à la recherche d'une chambre, un salon, cuisine et douche",
5   "date_pub": "29-10-06",
6   "type_pub": "location",
7   "clientId": 1,
8   "createdAt": "2021-06-01T18:25:15.460Z",
9   "updatedAt": "2021-06-01T18:25:35.076Z"
10 }
```

Find and Replace Console

Bootcamp Build Browse

