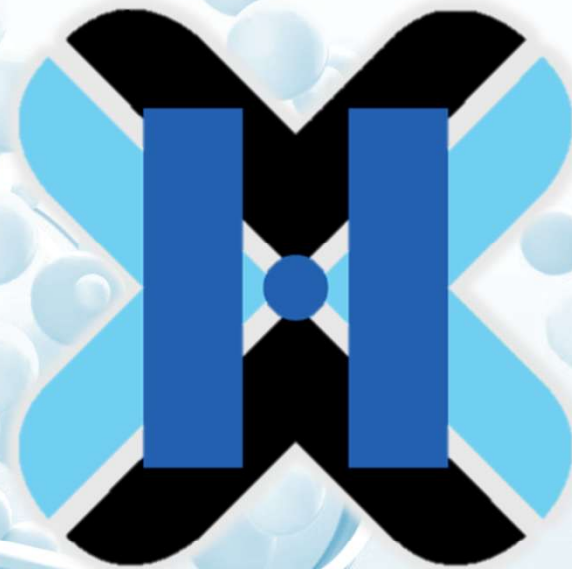


Formation

Par ValT et Orchidium



HOSTING



Que propose hosting ?

1. Authentification
2. Création de VMs
3. Manager la Vm
4. Créer une entrée DNS



1

Entrez votre identifiant et votre mot de passe.

Identifiant :

Mot de passe :

[SE CONNECTER](#)

2

Créez votre Machine Virtuelle

Présentation des types de machine virtuelle possibles

VM Vierge
Une simple VM vierge avec Debian 11, utilisez la comme stockage, machine de calcul ou ce que vous souhaitez.

Serveur-Web
Wordpress prêt à utilisation avec un serveur FTP.

Type

Nom de la VM

Clé SSH

Compte utilisateur

Mot de passe

Confirmer le mot de passe

Votre mot de passe doit contenir au minimum :
• 12 caractères.

Information sur la VM

ID : 153 Propriétaire : nstchepinsky
Type : bare_vm Créée le : 2023-06-20
Statut : running Dernière backup : 2024-05-25 04:12
Nom : hosting-dev-seaweedbrain Uptime : 64d 17h 46min 27s
Adresse IPv4 : 157.159.195.9

Démarrage automatique :
 Décochez cette option si vous ne souhaitez pas que votre VM soit automatiquement démarrée dans le cas d'un redémarrage de notre serveur

[Changer mes identifiants](#)

Informations

CPU : 2
RAM (GB) : 4
Espace disque (GB) : 10

[Supprimer](#) [Eteindre](#)

3

Admin DNS

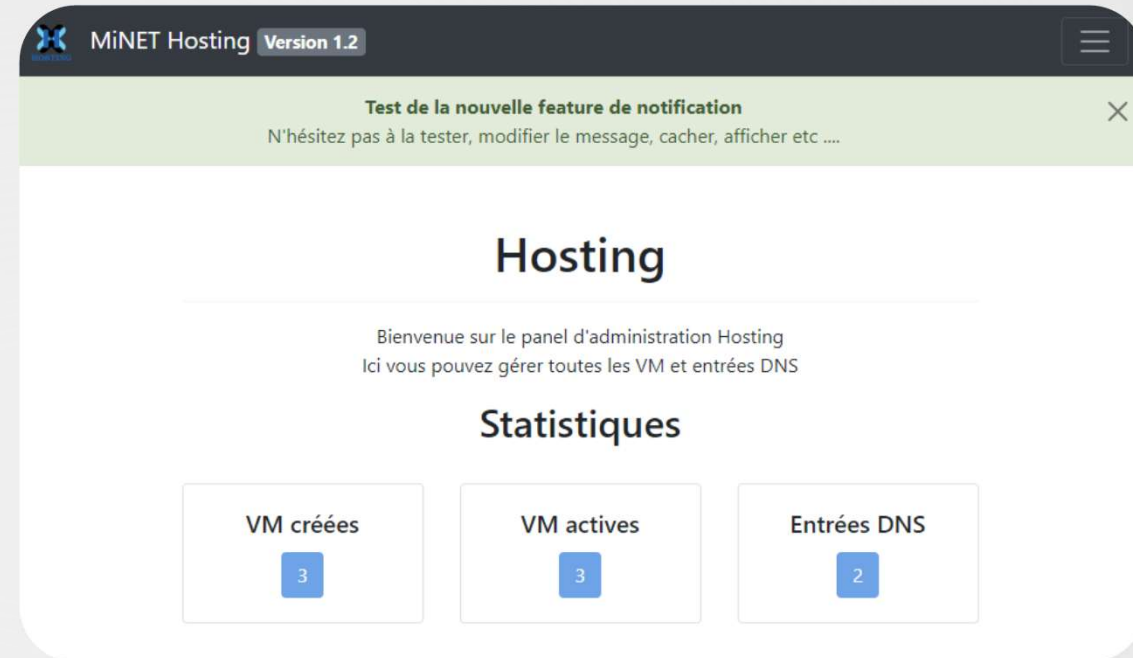
[Ajouter une entrée](#)

Entrées DNS en attente

User	Host	Destination
tburellier	MyDNSEntry	157.159.195.9
tburellier	MyDNSEntry2	157.159.195.9

4

C'est quoi le fonctionnement ?



On commence en se connectant au **site hosting**. C'est le frontend, la partie avec lequel interagit le client, réalisé en **Angular**

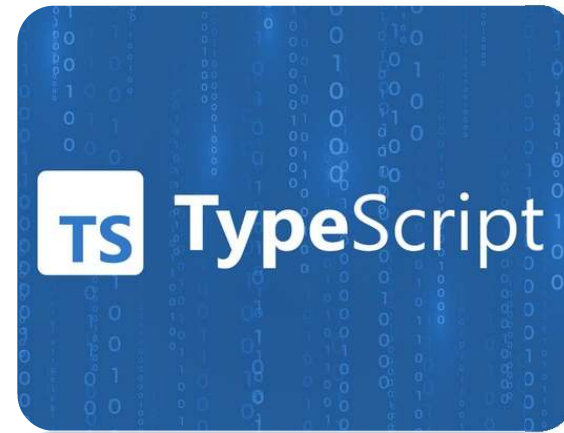
Angular c'est un framework web

**Un framework possède pleins
d'outils préconçus pour faire
rapidement des gros sites web**

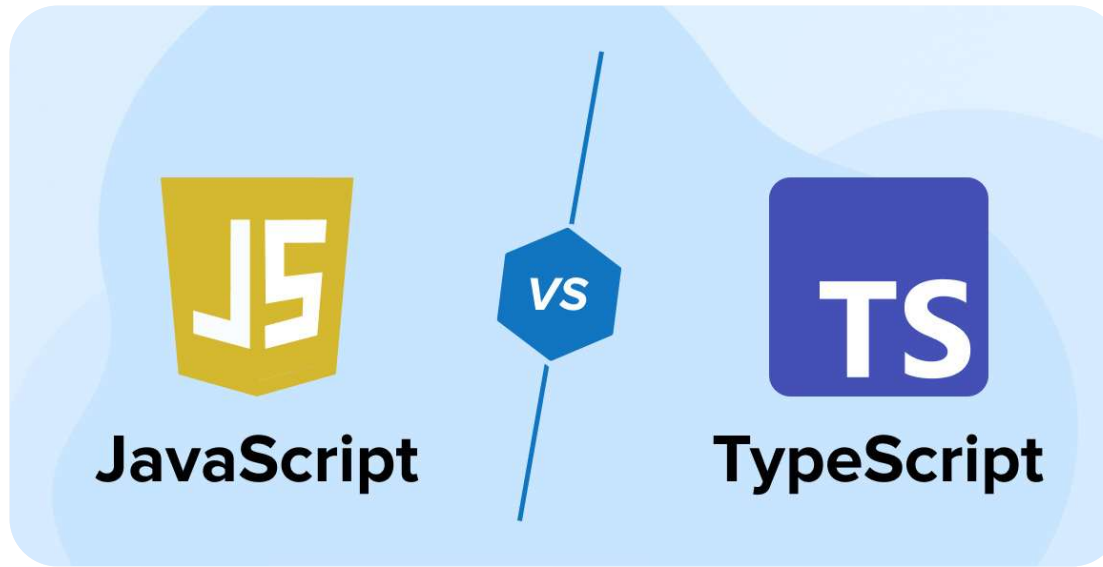
Pas besoin de réinventer la roue



ANGULAR



**Ça se code en
HTML, CSS et en
Typescript**



Aucun typage : ne te diras pas si tu fais n'importe quoi

Aucune structure : difficile de s'y retrouver

Aucune optimisation : ton code sera tel que tu l'as écrit dans ton site

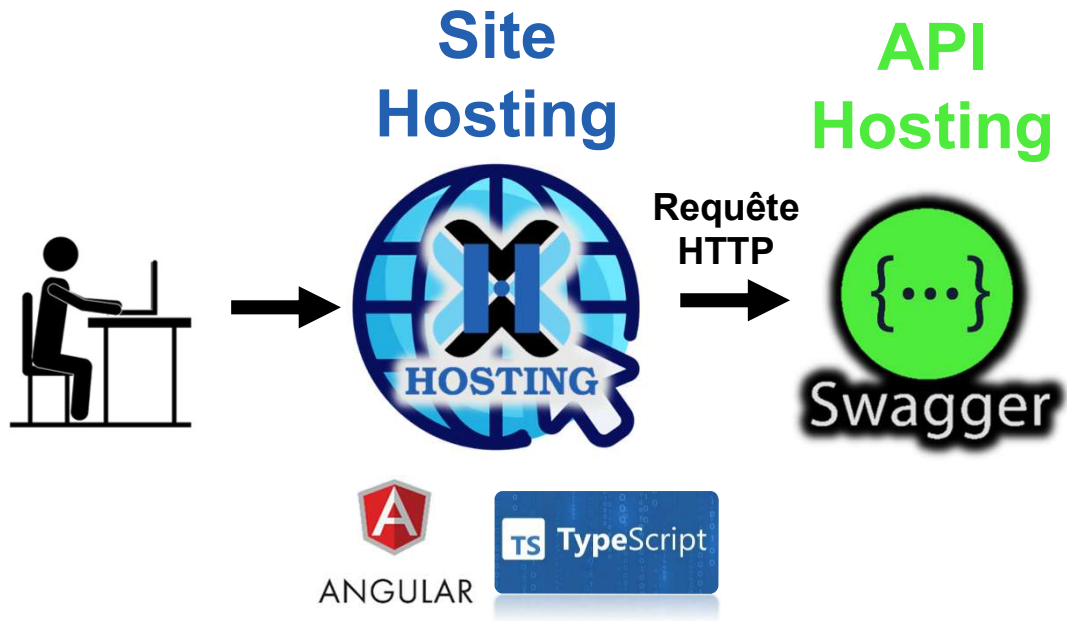
Typage strict : t'indique si y'a des erreurs

Gère la programmation orientée objet : permet de créer des classes et interfaces

Se compile de manière optimisée en javascript

Privilégié par tous les gros frameworks

C'est quoi le fonctionnement ?



VM: hosting-dev-seaweedbrain

Statut : allumée et en fonctionnement

Information sur la VM

ID : 153	Propriétaire : testchepinsky
Type : bare_vm	Créée le : 2024-06-20
Statut : running	Dernière backup : 2024-05-25 04:12
Nom : hosting-dev-seaweedbrain	Uptime : 64d 1h 13min 25s
Adresse IPv4 : 157.159.195.9	

Démarrage automatique : :
 Décochez cette option si vous ne souhaitez pas que votre VM soit automatiquement démarrée dans le cas d'un redémarrage de notre serveur

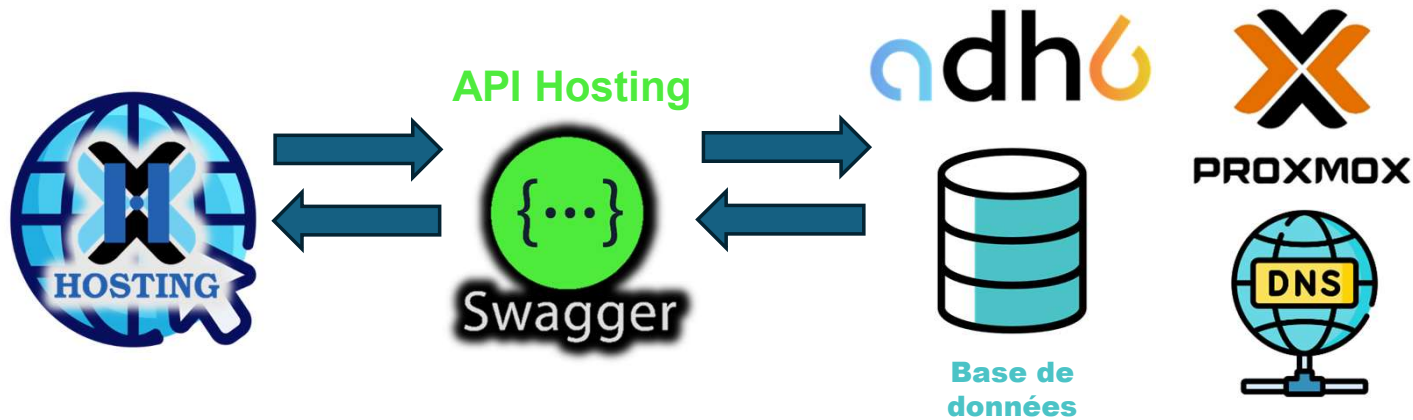
[Changer mes identifiants](#)

Informations sur le Système

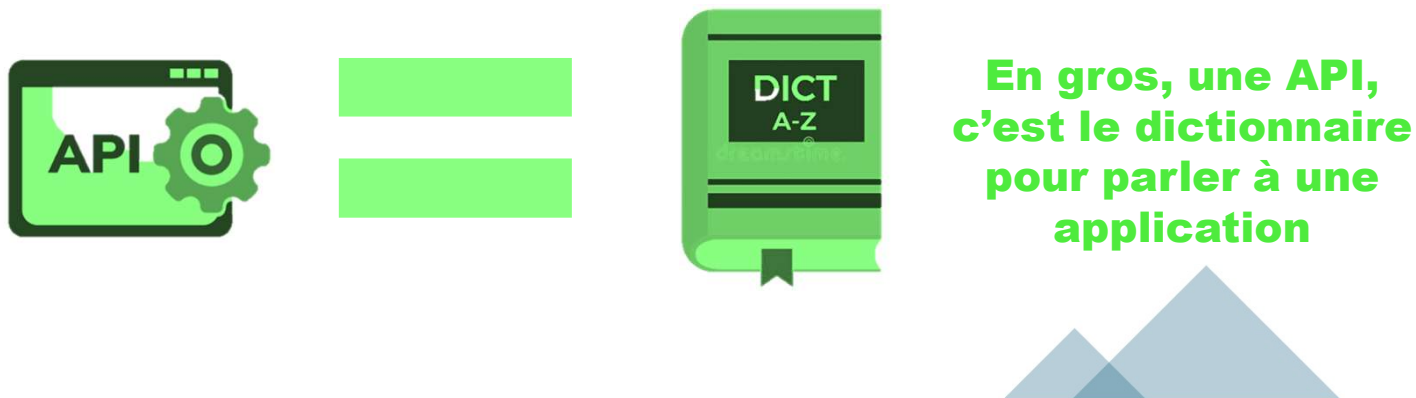
CPU : 2	1.8%
RAM (GB) : 4	0.3 / 4 GB
Espace disque (GB) : 10	

Quand j'affiche du contenu, genre les infos de **cette vm**, ou que je clique sur un bouton, ça exécute des fonctions **Typescript** qui exécute des requêtes **HTTP** vers l'API d'Hosting

Une API permet à n'importe quelle application A de communiquer avec une application B via des lignes de commandes



Dans notre cas, c'est le frontend Hosting qui veut accéder aux VMs proxmox, au DNS MiNET, ou aux adhérents adh6



User

Allez sur api-hosting.minet.net/2.0/ui

GET /account_state/{username} get the freeze state of a user account

GET /expired get the list of users with expired cotisation

GET /ips get the list of a user ip addresses

GET /max_account_ressources max ressources a single account can have

GET /notification Try to fetch a notification to display for the user

PATCH /notification Modify the notification (for admin only)

DNS

GET /dns get all user's dns entries. For admin, it returns all dns entries

POST /dns create a dns entry

POST /dns/validation validate a dns entry by id

DELETE /dns/{dnsid} delete dns entry by id in database, also delete the dns entry in the proxmox server if it exists (validated)

GET /dns/{dnsid} get a dns entry by id

VM

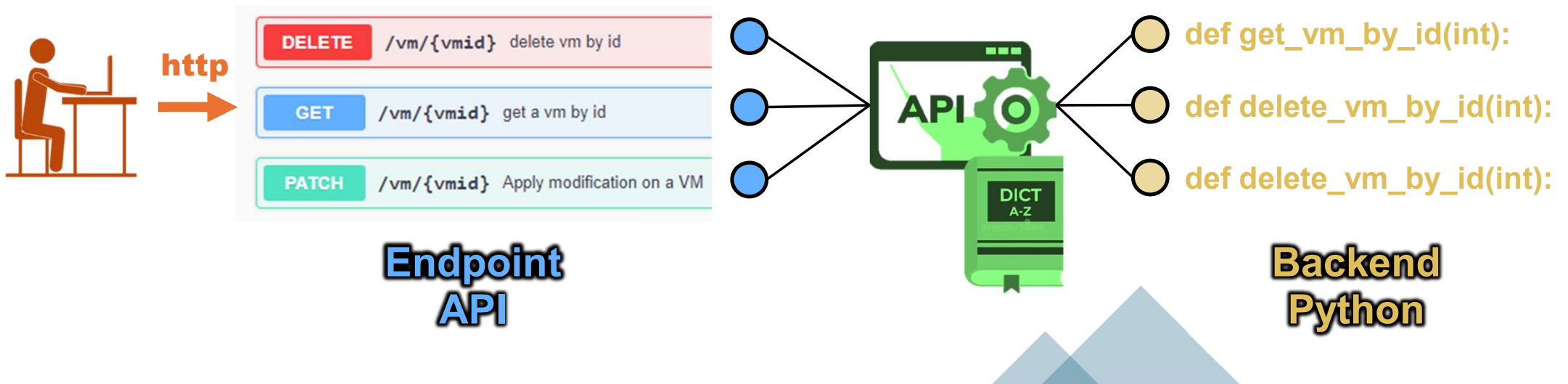
GET /history/{vmid} get the ip history of a vm

Voici l'ensemble des endpoints HTTP qui ont été codés pour hosting

Un EndPoint, c'est une action que l'API nous permet de réaliser en faisant une requête http

(C'est une définition dans le dictionnaire pour dialoguer avec l'application)

Et derrière chaque EndPoint, il y a une fonction python dans le backend



Par exemple, une requête http GET en typescript ça ressemble à ça :

```
this.http.get(« https://api-hosting.minet.net/vm/146 »);
```

(récupère les infos de la vm numéro 146 sur proxmox)

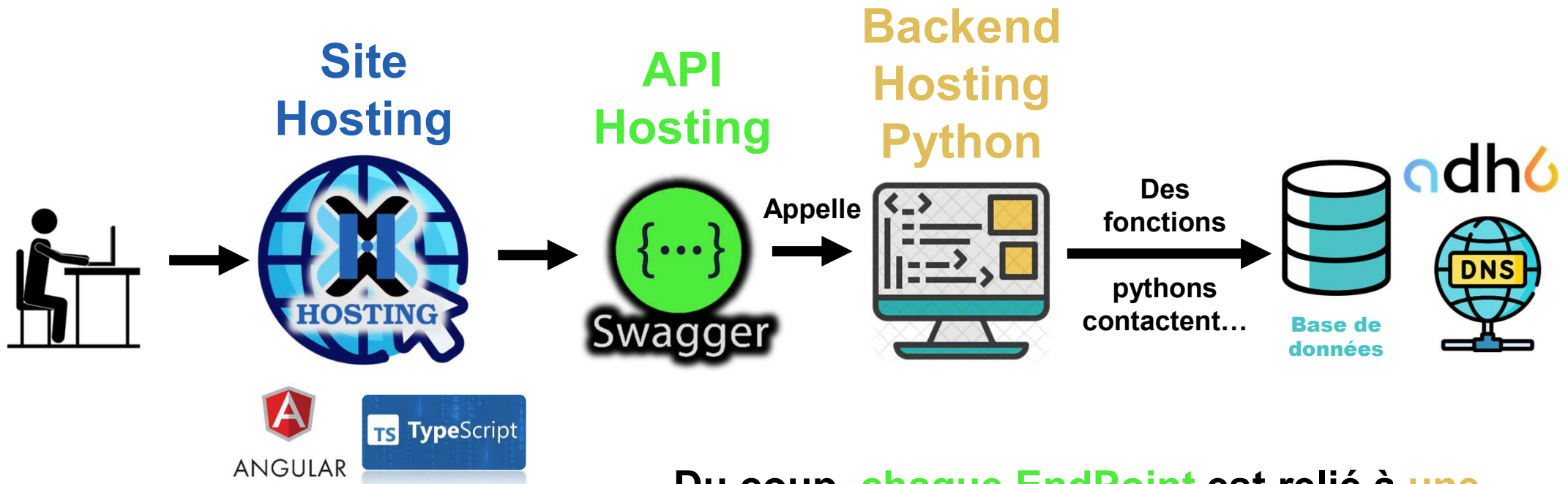
**Note : On pourrait faire ça dans n'importe quel langage, depuis n'importe quelle machine
Une API, c'est un dictionnaire qui fonctionne dans toutes les langues**

DELETE /vm/{vmid} delete vm by id

GET /vm/{vmid} get a vm by id

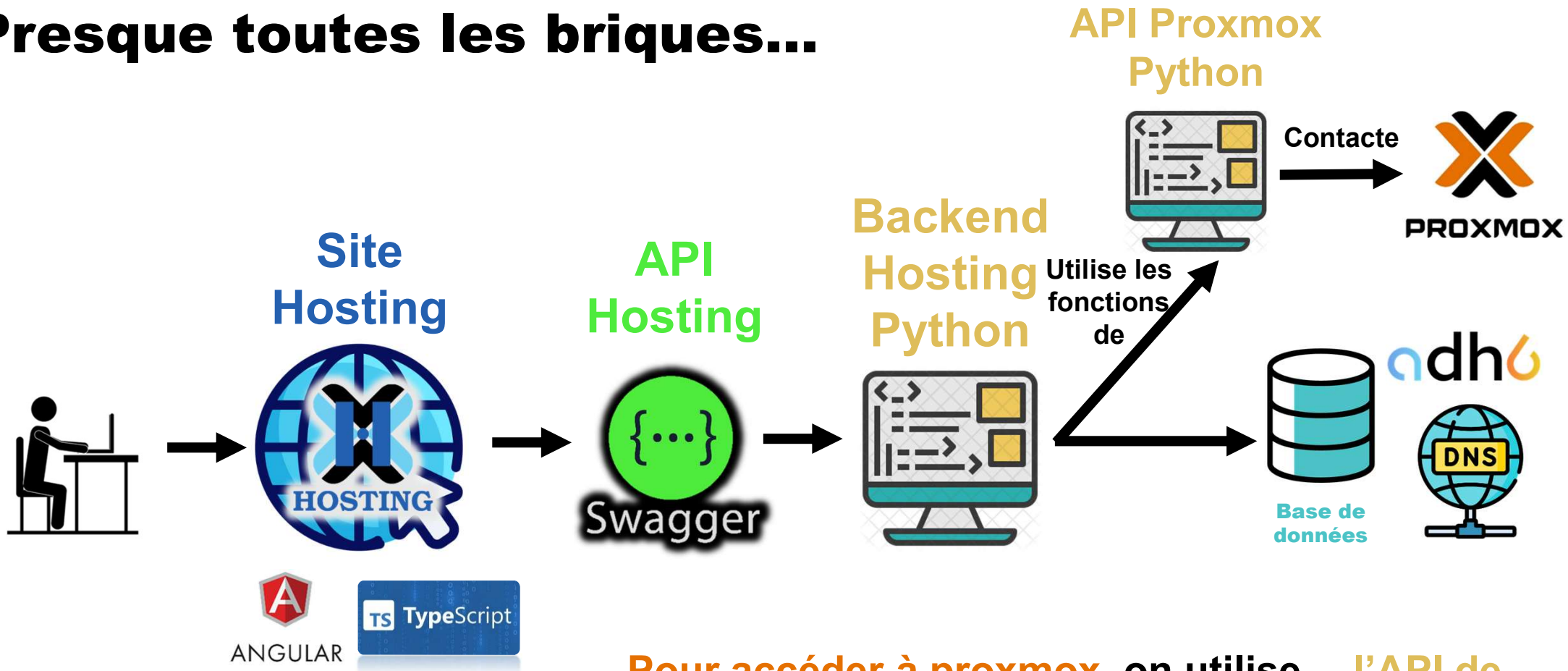
PATCH /vm/{vmid} Apply modification on a VM

Maintenant on a toutes les briques !



Du coup, **chaque EndPoint** est relié à **une fonction Python dans le backend d'hosting**. Et ensuite on utilise tout un tas de bibliothèques Python pour contacter **le DNS, la BDD ou adh6**

Presque toutes les briques...



Pour accéder à proxmox, on utilise... l'API de proxmox. C'est aussi une librairie python, avec des fonctions déjà toutes faites pour obtenir toutes les infos dispos sur Proxmox

The screenshot shows the Proxmox API viewer interface. On the left is a resource tree with a search bar and a list of folders including 'nodes', 'qemu', and 'status'. The 'status' folder is highlighted with a red box. The main panel shows the path `/nodes/{node}/qemu/{vmid}/status/current` and a `GET` method. Below this, there is a description: 'Get virtual machine status.' and a usage section with the following information:

HTTP: `GET /api2/json/nodes/{node}/qemu/{vmid}/status/current`

CLI: `pvesh get /nodes/{node}/qemu/{vmid}/status/current`

Parameters:

Name ↑	Type	Default	Format
Required			
node	string		<string>
vmid	integer		<integer>

Returns: object

Name ↑	Type	Default	Format
Obligatory			
ha	object		
status	enum		stopped
vmid	integer		pve-vmid
Optional			
agent	boolean		<true/false>
clipboard	enum		vnc
cpus	number		
lock	string		
maxdisk	integer		
maxmem	integer		
name	string		
usage	object		

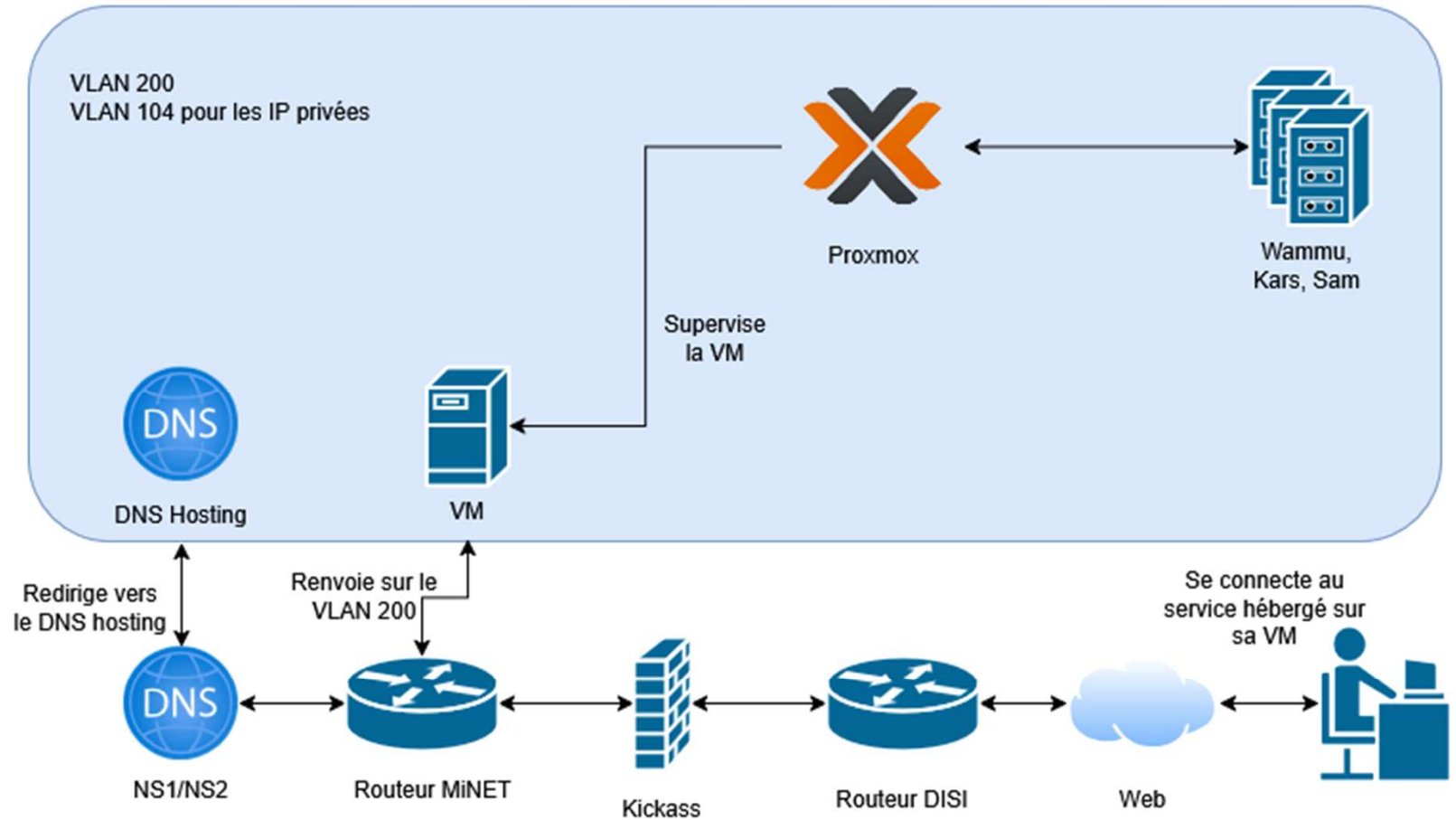
Allez sur ce site :

pve.proxmox.com/pve-docs/api-viewer

Contient tous les EndPoints de l'API proxmox

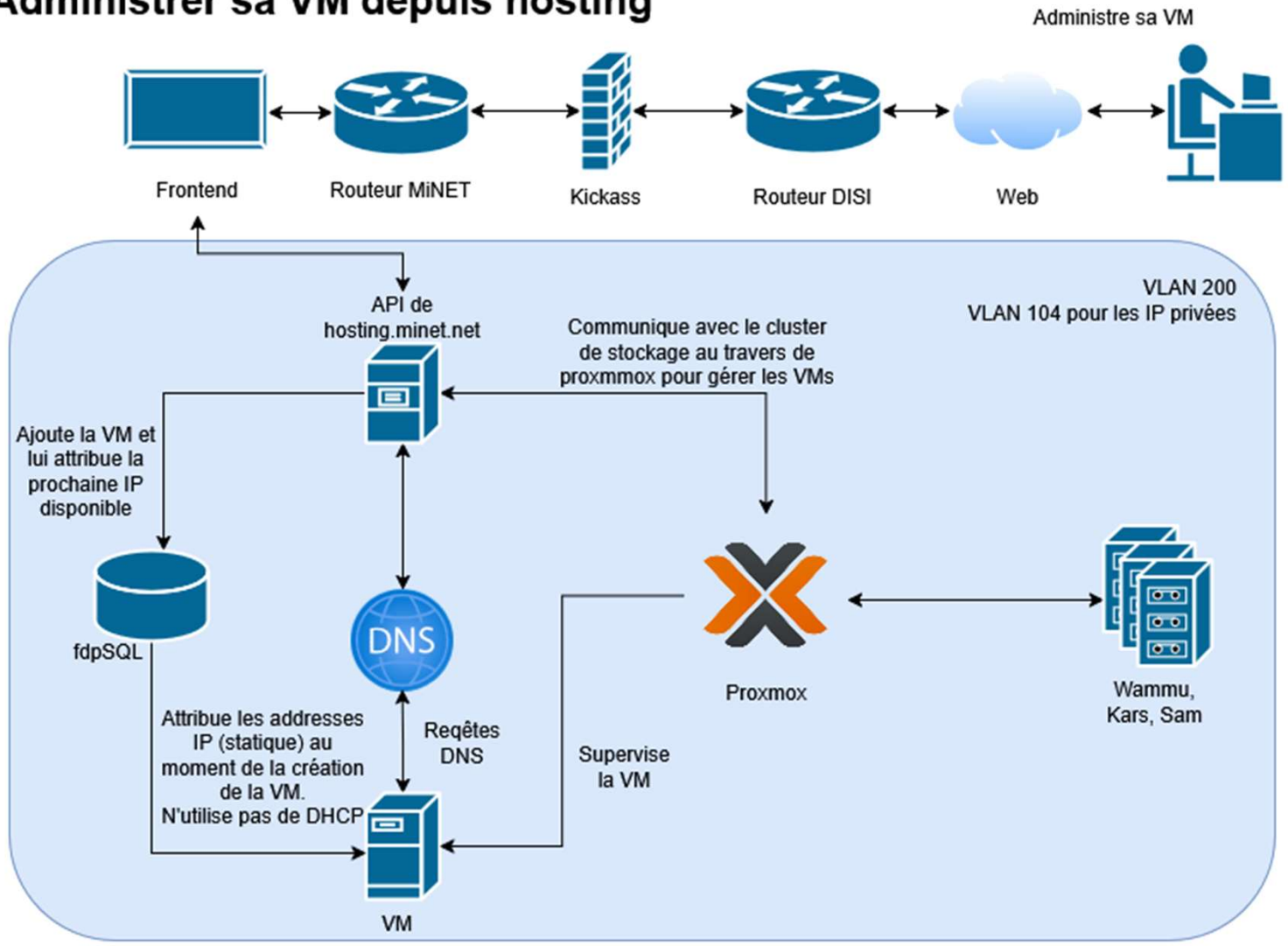
Par exemple celui-là c'est pour obtenir les infos de la vm

Accéder au service hébergé sur sa VM





Administrer sa VM depuis hosting



Et du coup oui ! Derrière hosting, il y a juste un proxmox avec toutes les VMs des adhérents. Comme sur la Dev ou la Prod

PROXMOX Virtual Environment 7.4-17 api

Documentation Create VM Create CT root@pam

Server View Node 'kars'

Reboot Shutdown Shell Bulk Actions Help

Search:

Type	Description	Disk usage...	Memory us...	CPU usage	Uptime	Host CPU ...	Host Mem...	Tags
lxc	134 (packer)				-			
lxc	155 (hosting-dev)	77.4 %	7.1 %	0.2% of 2 ...	63 days 07:3...	0.0% of 48...	0.2 %	
lxc	1097 (test-db-2)				-			
lxc	1098 (test-db-1)				-			
lxc	5000 (checkssh)				-			
qemu	101 (asint-com)	0.0 %	29.5 %	4.8% of 2 ...	63 days 07:3...	0.2% of 48...	0.3 %	
qemu	104 (k8s-master1)	0.0 %	72.6 %	10.6% of 2 ...	63 days 07:3...	0.4% of 48...	1.5 %	
qemu	105 (api)	0.0 %	34.0 %	0.9% of 2 ...	63 days 07:3...	0.0% of 48...	1.1 %	
qemu	106 (k8s-master2)	0.0 %	70.1 %	7.4% of 2 ...	63 days 07:3...	0.3% of 48...	1.5 %	
qemu	110 (site-web-bpm)				-			
qemu	112 (lpe-siteweb)	0.0 %	34.5 %	0.8% of 2 ...	63 days 07:3...	0.0% of 48...	0.7 %	
qemu	114 (bridge-messenger-disc...)				-			
qemu	116 (serveur-modpack)	0.0 %	95.6 %	11.6% of 6 ...	63 days 07:3...	1.4% of 48...	4.6 %	
qemu	118 (websites)	0.0 %	73.7 %	1.0% of 2 ...	63 days 07:3...	0.0% of 48...	1.6 %	
qemu	119 (bpm-vitrine)				-			
qemu	121 (forma-web)	0.0 %	16.1 %	0.2% of 6 ...	63 days 07:3...	0.0% of 48...	0.8 %	
qemu	125 (automationtasks)	0.0 %	77.3 %	41.5% of 2 ...	63 days 07:3...	1.7% of 48...	1.6 %	
qemu	129 (pastas-legacy)	0.0 %	39.9 %	9.5% of 1 ...	63 days 07:3...	0.2% of 48...	0.2 %	
qemu	132 (ctf)	0.0 %	28.9 %	0.4% of 3 ...	63 days 07:3...	0.0% of 48...	0.6 %	
qemu	133 (zero-totp)	0.0 %	57.2 %	1.2% of 2 ...	63 days 07:3...	0.1% of 48...	2.4 %	
qemu	135 (phobos)	0.0 %	59.8 %	2.7% of 2 ...	63 days 07:3...	0.1% of 48...	1.3 %	
qemu	136 (rdln)				-			

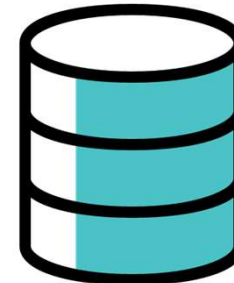
Tasks Cluster log

Start Time	End Time	Node	User name	Description	Status
May 23 23:48:06		wammu	root@pam	VM 128 - Migrate	
Mav 23 23:48:09	Mav 23 23:48:11	kars	root@pam	VM 128 - Start	OK

Il faut savoir qu'il y a TROIS Hosting

Hosting.minet.net :
production

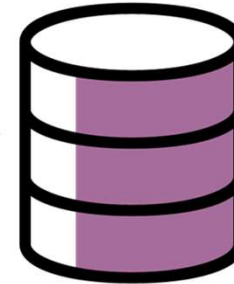
C'est le site qu'on connaît



Base de données de production : toutes les VMs des adhérents

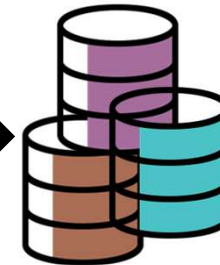
Hosting-dev.minet.net :
développement

C'est un site qui est aussi accessible en permanence, mais qui contient du code pas encore push sur main sur gitlab



Base de données de dev : c'est sur le même cluster, mais pas la même bdd

Hosting-local.minet.net : vous quand vous lancez le projet sur votre pc



On peut choisir quelle base utiliser en local. Et y'a une base dédiée au test de code

Grâce à Angular, on peut afficher des trucs totalement différents en fonction de qui est connecté

Y'a en gros 4 modes :

Home VMs DNS Delete VM Notification

Test de la nouvelle feature de notification
N'hésitez pas à la tester, modifier le message, cacher, afficher etc ...

Hosting

Bienvenue sur le panel d'administration Hosting
Ici vous pouvez gérer toutes les VM et entrées DNS

Statistiques

VM créées

3

VM actives

3

Entrées DNS

2

Mode Admin



venue sur le nouveau service d'hébergement de MiNET. En cas de problème n'hésitez pas à ouvrir un ticket sur [webmaster!](#)
ce est gratuit si vous possédez déjà un compte MiNET. Autrement, veuillez utiliser [payment.minet.net](#) pour vous abonner.

Démarrer

virtuelle ou un
à l'emploi.



Créer un enregistrement DNS
pointant vers votre machine
virtuelle.

98%

d'uptime en 2023

Mode Pas Connecté

Hosting

Vous pouvez créer ici des machines virtuelles qui hébergeront votre site web ou le service de votre choix.

Avant d'avoir lu nos règles et notre manuel d'utilisation avant toute manipulation. N'hésitez pas à ouvrir un ticket sur [tickets.minet.net](#) pour toute question.

3 CPUs restant

3 GO de RAM restant

10 GO de stockage restant

Choisissez votre Machine Virtuelle

Présentation des types de machine virtuelle possibles

Charge
Exemple VM vierge avec Debian 11, utilisez la comme stockage, machine de calcul ou ce que vous souhaitez !

Server-Web

Adresse IP prêt à utilisation avec un serveur-FTP.

Type

Nom de la VM

Mode Adhérent

Mon DNS

Nath

Hosting



Votre compte est gelé !

Vous pourrez toujours gérer vos VMs pendant plusieurs jours. Après 30 jours d'expiration, vos VMs seront définitivement détruites. ATTENTION : vous avez seulement 30 jours pour renouveler votre cotisation avant l'extinction de vos VMs

Votre cotisation a expiré

Mode Cotisation expiré

Freeze State

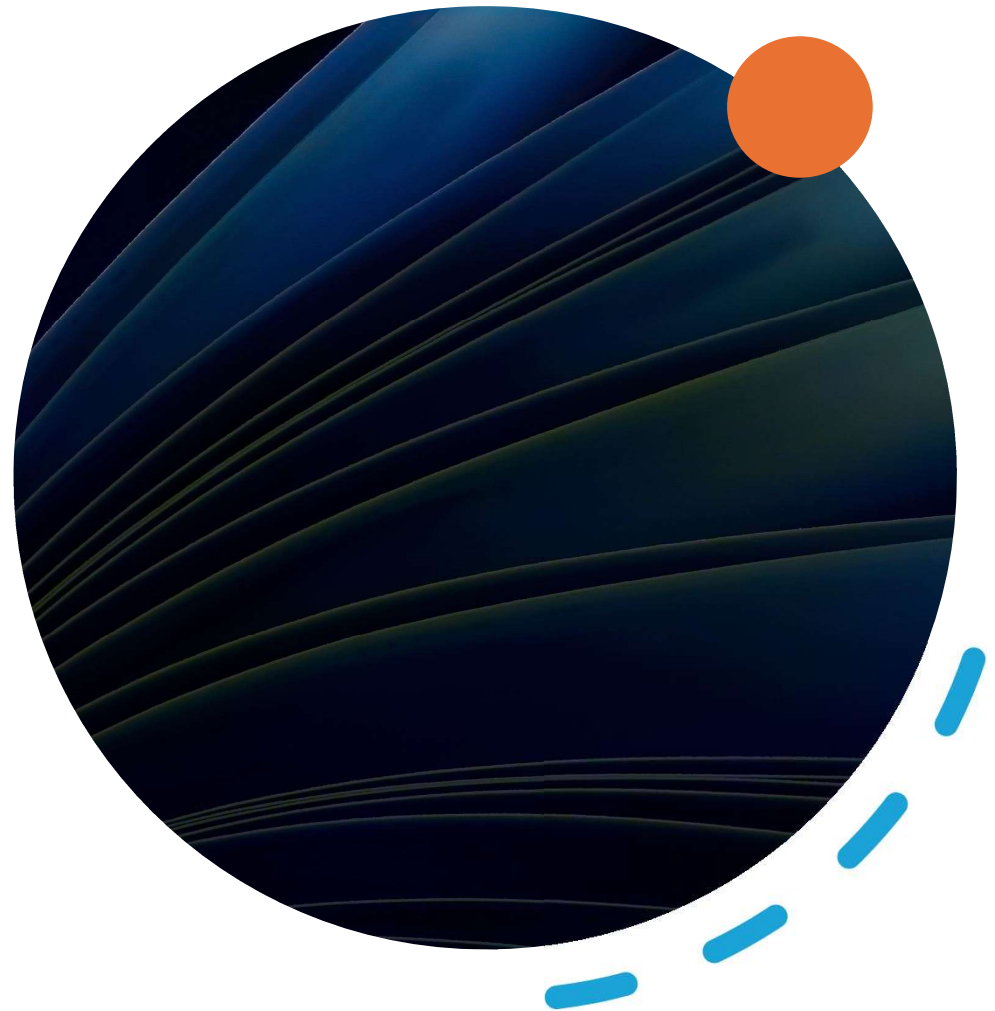
Un adhérent a un freeze state de 1.0 quand il cotise

Dès que sa cotisation expire, il passe à un freeze state de 2.1

Le freeze state d'un adhérent expiré va changer toutes les semaines et il recevra un mail : 2.1 → 2.2 → 2.3 → 2.4 → 3.1 → 3.2 → 3.3 → 3.4 → 4.1 → ...

A partir d'un freeze state de 4, on peut supprimer les VMs de l'adhérent

Les freezes states sont gérés sur le jenkins de minet (faut trouver l'IP sur le wiki)



**Passons à du
concret, on va voir
comment bien
débuter sur hosting**

**Cloner le
repo sur
Github :**

**Le projet est mirroré sur Github pour
être accessible à tous :**

<https://github.com/minet/hostingproxmox>



**Ensuite faut
installer
tout un tas
de trucs :**

\$ sudo apt install npm

\$ npm install -g @angular/cli

**<https://doc.ubuntu-fr.org/nodejs>
(depuis un PPA)**



Setup le projet

Hosting est divisé en 2 dossiers : le frontend avec Angular, le backend avec Flask

Installer les dépendances angular utilisée dans le projet :

```
$ cd frontend && npm install
```

Créer un environnement virtuel python avec les dépendances utilisées par le projet :

```
$ cd backend && python3 -m venv venv && source venv/bin/activate  
&& pip3 install -r requirements.txt
```

Setup le projet

Pour finir il faut créer un fichier nommé « **.env** » dans la racine du projet. Il sert à **stocker les variables d'environnement** pour se connecter aux différents serveurs d'Hosting

Pour des raisons de sécurité, je ne peux pas les donner ici, et c'est pas grave puisqu'on peut faire cette formation sans. Mais il faudra les demander si vous voulez vraiment travailler sur hosting

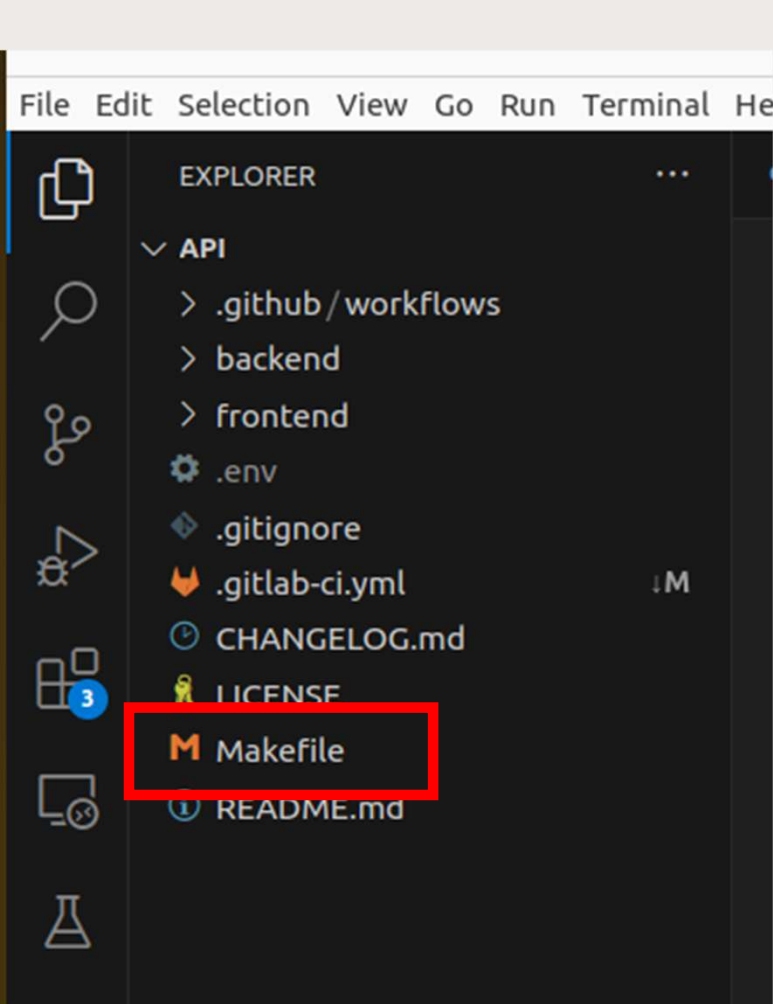
Copier-coller ça dans le **.env** :

```
export KEYRING_DNS_SECRET=<KEYRING_DNS_SECRET>
export PROXMOX_API_KEY_NAME=<PROXMOX_API_KEY_NAME>
export PROXMOX_API_KEY=<PROXMOX_API_KEY>
export PROXMOX_BACK_DB=<PROXMOX_BACK_DB>
export ADH6_API_KEY=<ADH6_API_KEY>
export PROXMOX_BACK_DB_DEV=<PROXMOX_BACK_DB_DEV>
export ENVIRONMENT= « DEV »
```


Le makefile

C'est un fichier qui permet d'exécuter pleins de commandes rapidement

Notamment ça sert pour faire toutes les commandes pour lancer le projet d'un coup



```
M Makefile
1  .DEFAULT_GOAL := run
2
3  install_server:
4      rm -rf backend/venv
5      python3 -m venv backend/venv
6      backend/venv/bin/pip install -r backend/requirements.txt
7
8  install_frontend:
9      cd frontend && npm install
10
11 install_all: install_server install_frontend
12
13 run_server:
14     . ./env && . backend/venv/bin/activate && cd backend && python3 -m proxmox_api
15
16 run_frontend:
17     cd frontend && ng serve --host=127.0.0.1 --disable-host-check
18
19 run:
20     echo "Starting server ..."
21     make run_server & \
22     echo "Starting frontend on http://hosting-local.minet.net:4200/ ..." && \
23     make run_frontend
24
25 clean:
26     rm -rf pycache
```

On va PAS pouvoir utiliser le backend

Du coup on va faire `$ make run_frontend`

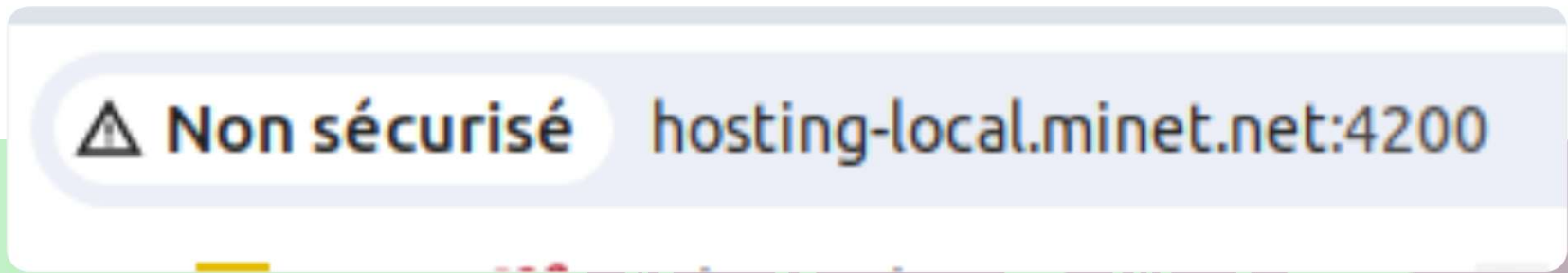
Normalement on pourrait utiliser make pour lancer les 2

Faut modifier le `/etc/hosts` pour ajouter :

`127.0.0.1 hosting-local.minet.net`

Puis faut aller sur `hosting-local.minet.net:4200`
(`127.0.0.1:4200` fonctionne pas)

```
GNU nano 6.2
127.0.0.1 localhost
127.0.1.1 Valt
127.0.0.1 hosting-local.minet.net
192.168.103.2 gitlabint.priv.minet.net
192.168.102.2 gitlabint.priv.minet.net
```






Entrez votre identifiant et votre mot de passe.

Identifiant :

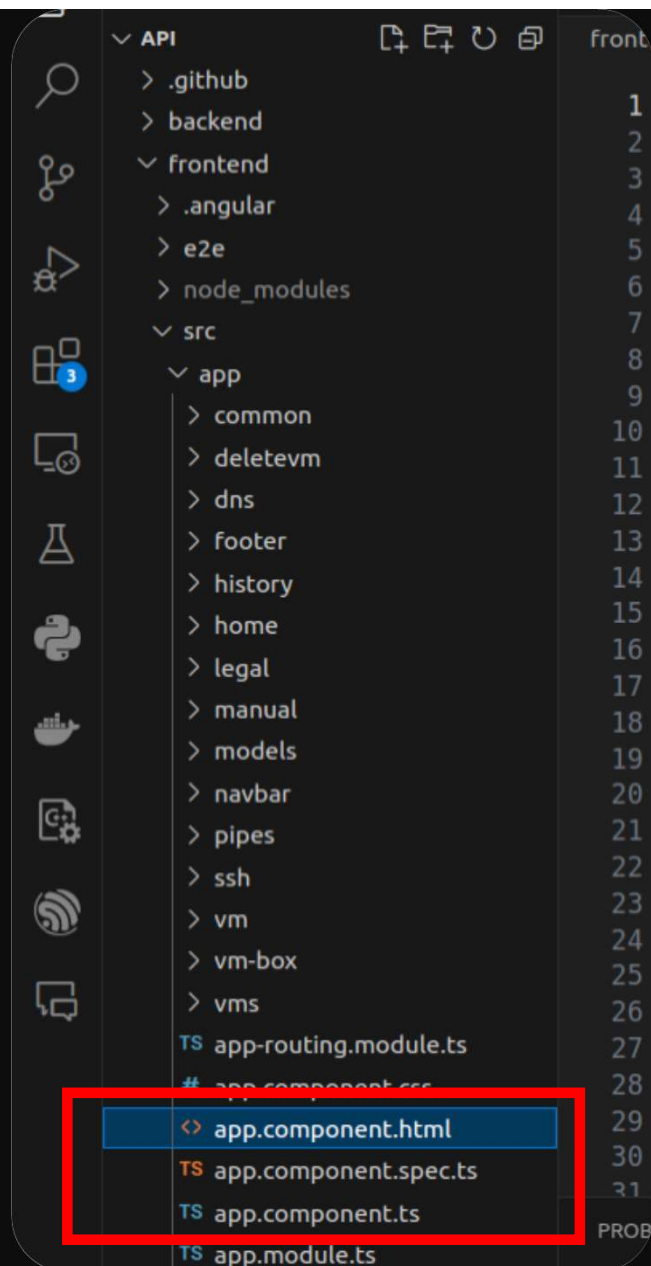
Mot de passe :

SE CONNECTER



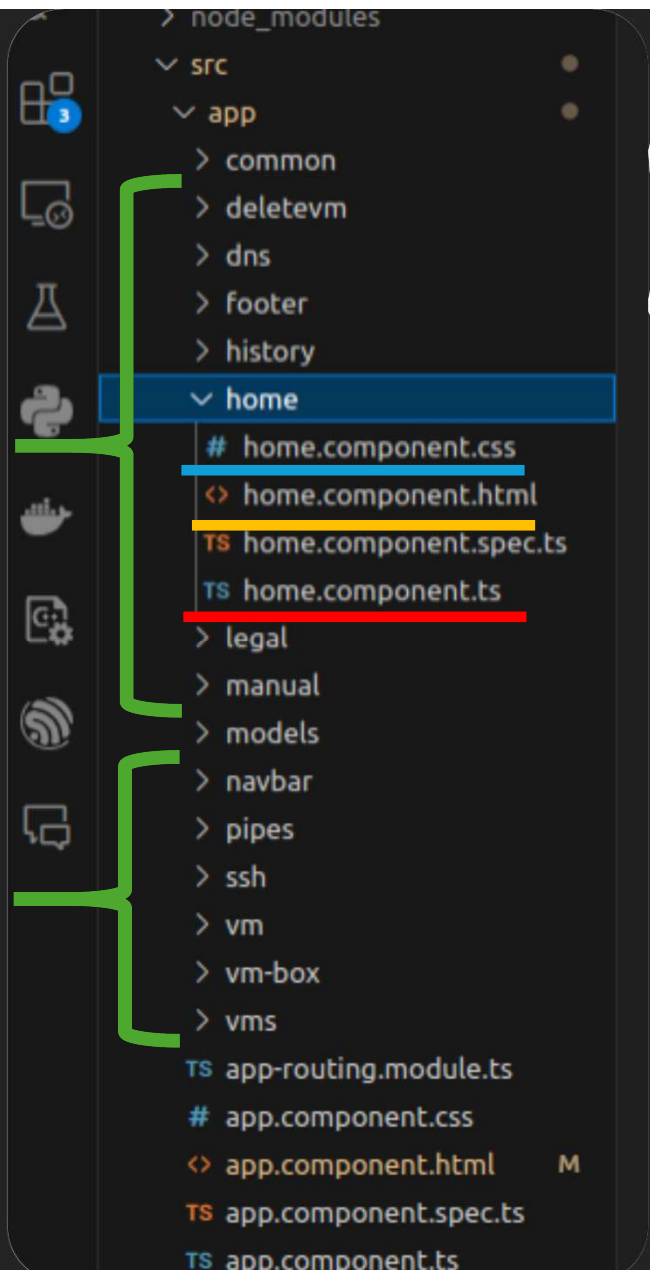
**Là vous avez accès
à la page d'accueil
et vous pouvez
vous connecter au
cas de minet avec
vos creds
d'adhérent**

**(pas ldap, vous
êtes pas admin
hosting)**



Pour comprendre comment tout ça fonctionne, commençons par revenir à la racine de l'app : **app.component.html et **app.component.ts****

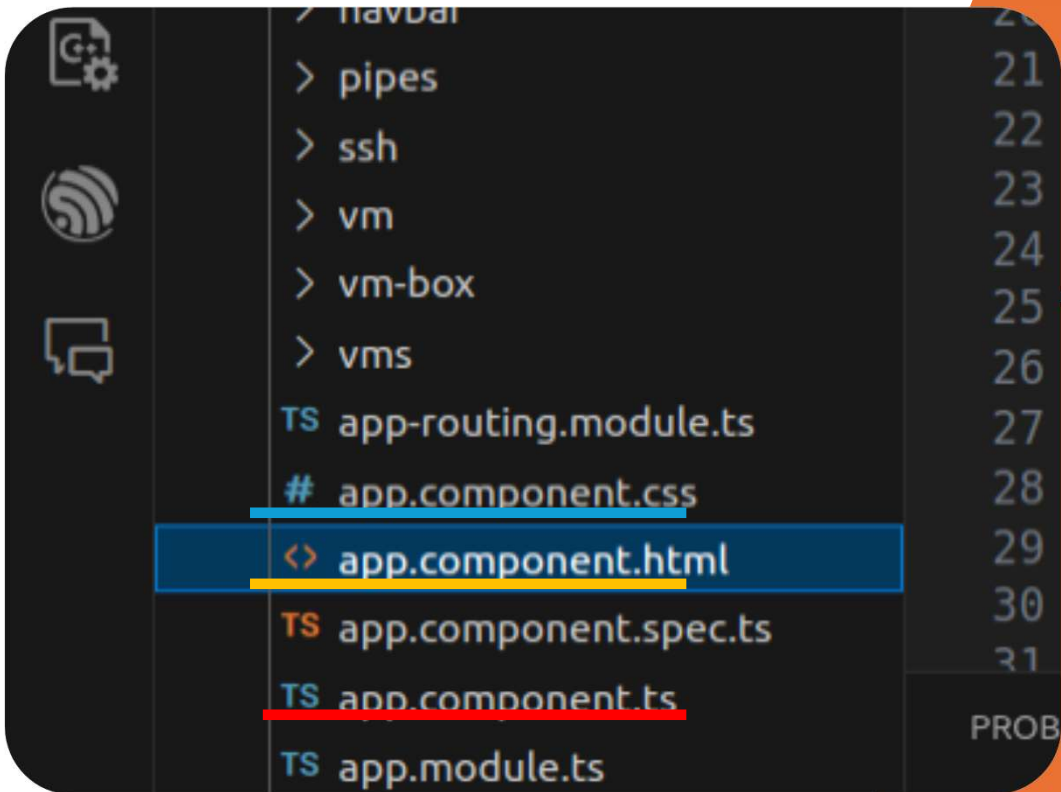
C'est là que démarre notre app



Faut savoir que Angular fonctionne avec un système de component. Un component c'est **un fichier html**, couplé à **un fichier typescript** et **un fichier css**. Les 3 fonctionnent ensemble et sont connectés

Et un component peut ensuite être inséré et réutilisé n'importe où

Par exemple, **chacun des dossiers** que vous voyez là sont des components

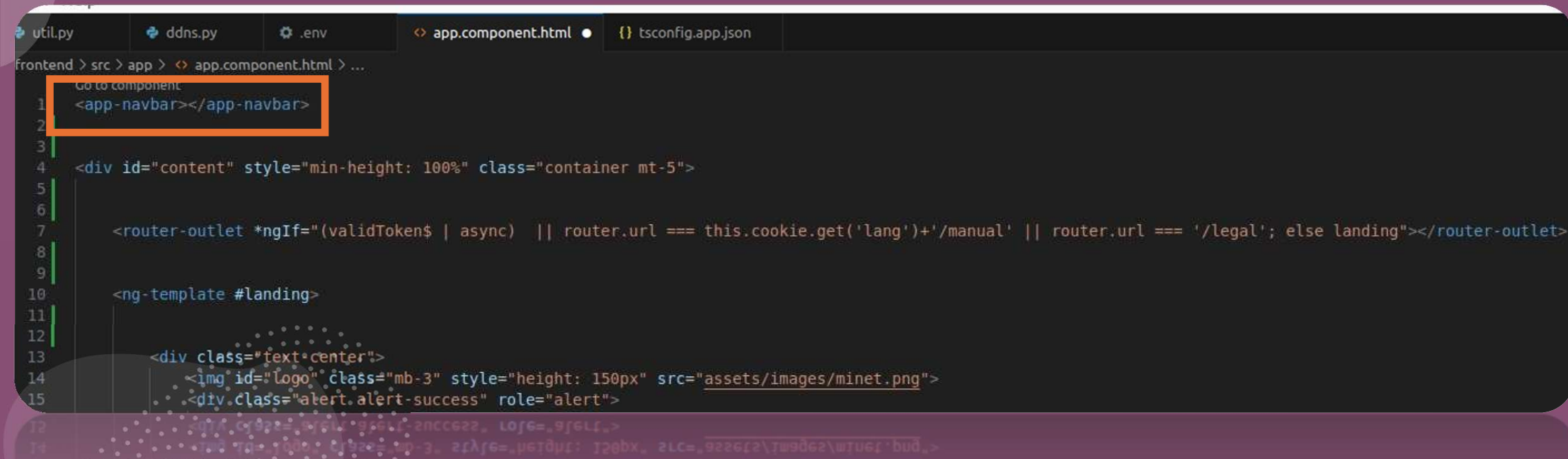


```
> navBar 20
> pipes 21
> ssh 22
> vm 23
> vm-box 24
> vms 26
TS app-routing.module.ts 27
# app.component.css 28
<> app.component.html 29
TS app.component.spec.ts 30
TS app.component.ts 31
PROB
```

Et du coup, y'a un component au-dessus des autres, c'est le **app.component**. C'est un peu le component qui s'affiche par défaut et sur lequel va venir se greffer tout le reste

Tout en haut, on voit qu'on insère un **app-navbar component**

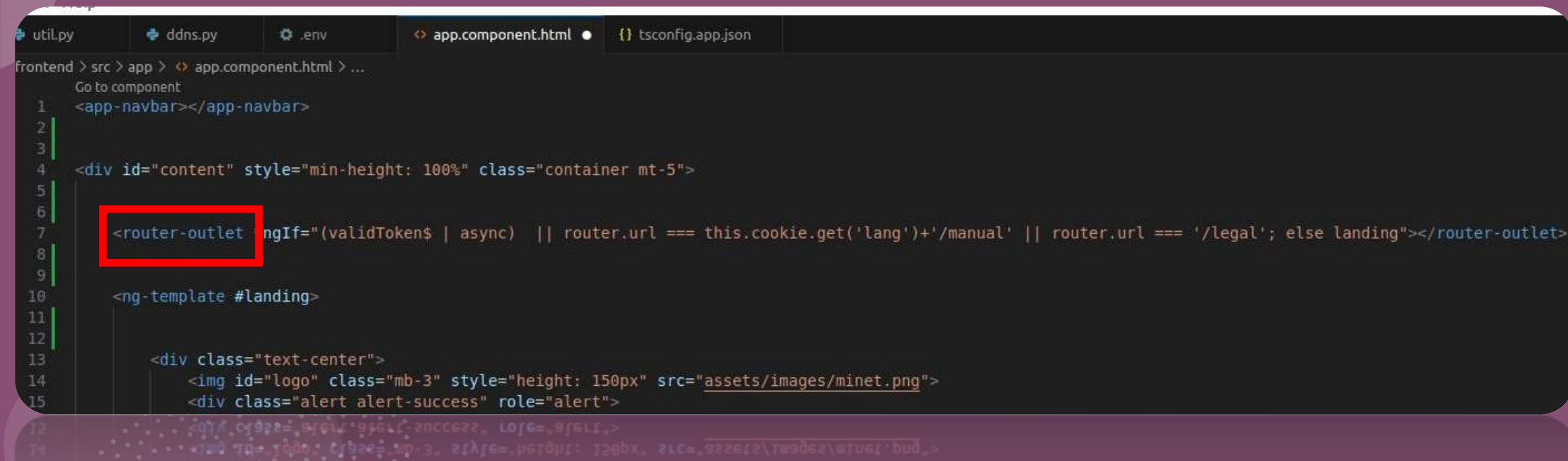
De même tout en bas, on a **le footer-component**. Vu que c'est le **component de base**, ça veut dire que peu importe où on sera dans l'application, on aura un navbar, et un footer. Ils sont définis dans les **dossiers « navbar » et « footer »**



```
util.py | ddns.py | .env | app.component.html | tsconfig.app.json
frontend > src > app > app.component.html > ...
Go to component
1 <app-navbar></app-navbar>
2
3
4 <div id="content" style="min-height: 100%" class="container mt-5">
5
6
7 <router-outlet *ngIf="(validToken$ | async) || router.url === this.cookie.get('lang')+'/manual' || router.url === '/legal'; else landing"></router-outlet>
8
9
10 <ng-template #landing>
11
12
13 <div class="text-center">
14 
15 <div class="alert alert-success" role="alert">
16
17
18
```

Ensuite on a le **"router-outlet"**. Ça permet d'afficher un **composant différent** en fonction de l'url de votre site

Par exemple, si mon url c'est **"/vms"**, je vais afficher le **composant vms**. Et si mon url c'est **"/dns"**, je vais afficher le **composant dns**



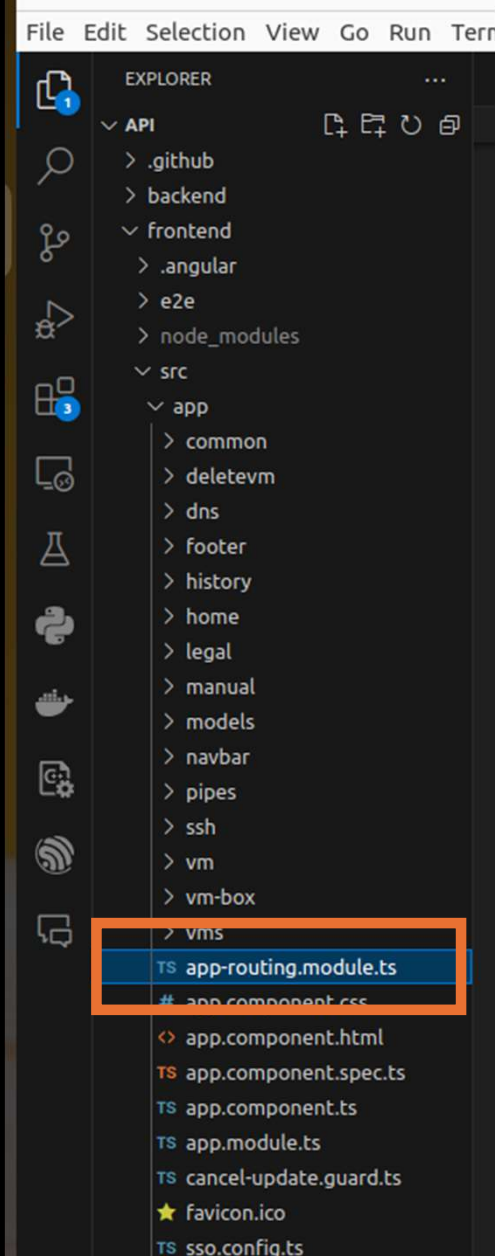
```
util.py | ddns.py | .env | app.component.html | tsconfig.app.json
frontend > src > app > app.component.html > ...
Go to component
1 <app-navbar></app-navbar>
2
3
4 <div id="content" style="min-height: 100%" class="container mt-5">
5
6
7 <router-outlet [ngIf="(validToken$ | async) || router.url === this.cookie.get('lang')+'/manual' || router.url === '/legal'; else landing]"></router-outlet>
8
9
10 <ng-template #landing>
11
12
13 <div class="text-center">
14 
15 <div class="alert alert-success" role="alert">
16
17
18
```



```
6 import {DnsComponent} from './dns/dns.component';
7 import {LegalComponent} from './legal/legal.component';
8 import {ManualComponent} from './manual/manual.component';
9 import {HistoryComponent} from './history/history.component';
10 import {DeletevmComponent} from './deletevm/deletevm.component';
11 import { CancelUpdateGuard } from './cancel-update.guard';
12
13
14 const routes: Routes = [
15   {path: '', component: HomeComponent},
16   {path: 'vms', component: VmsComponent},
17   {path: 'vms/vmid', component: VmComponent, canActivate: [CancelUpdateGuard]},
18   {path: 'dns', component: DnsComponent, canActivate: [CancelUpdateGuard]},
19   {path: 'legal', component: LegalComponent},
20   {path: 'manual', component: ManualComponent},
21   {path: 'history', component: HistoryComponent, canActivate: [CancelUpdateGuard]},
22   {path: 'deletevm', component: DeletevmComponent},
23   {path: '**', redirectTo: ''},
24 ];
25
26
27 @NgModule({
28   imports: [RouterModule.forRoot(routes, {relativeLinkResolution: 'legacy'})],
29   exports: [RouterModule]
30 })
31 export class AppRoutingModule {
32 }
33
```

Ce qui va être affiché à la place du **router-outlet**, c'est défini dans le **app-routing.module**

En gros, quand on est sur telle URL, on affiche tel component



Pour finir, vous pouvez remarquer le `*ngIf`. Angular possède une feature trop bien : vous pouvez afficher du HTML à certaines conditions !

C'est très pratique, par exemple, pour afficher des boutons visibles que pour les admins. Ou que pour les gens authentifiés

```
util.py  ddns.py  .env  app.component.html  tsconfig.app.json
frontend > src > app > app.component.html > ...
Go to component
1 <app-navbar></app-navbar>
2
3
4 <div id="content" style="min-height: 100%" class="container mt-5">
5
6
7 <router-outlet *ngIf="(validToken$ | async) || router.url === this.cookie.get('lang')+'/manual' || router.url === '/legal' else landing"></router-outlet>
8
9
10 <ng-template #landing>
11
12
13 <div class="text-center">
14 
15 <div class="alert alert-success" role="alert">
16
17
18
```

Par exemple ici : le router-outlet affiche des trucs QUE SI : vous êtes authentifié, OU vous êtes sur les routes **"/legal" ou **"/manual"**. Parce que y'a pas besoin de s'authentifier pour voir ces routes.**

```
util.py  ddns.py  .env  app.component.html  tsconfig.app.json
frontend > src > app > <> app.component.html > ...
Go to component
1 <app-navbar></app-navbar>
2
3
4 <div id="content" style="min-height: 100%" class="container mt-5">
5
6
7 <router-outlet *ngIf="(validToken$ | async) || router.url === this.cookie.get('lang')+'/manual' || router.url === '/legal' else landing"></router-outlet>
8
9
10 <ng-template #landing>
11
12
13 <div class="text-center">
14 
15 <div class="alert alert-success" role="alert">
16 </div>
17 </div>
18 </ng-template>
19 </div>
20 </pre>
```

**Et si ces conditions ne sont pas validées, alors on affiche le
ng-template "landing"**

**Ce truc est défini juste en dessous. Globalement ça correspond à la page
d'accueil. Et du coup elle sera affichée que si le router-outlet ne s'affiche pas**

```
util.py  ddns.py  .env  app.component.html  tsconfig.app.json
frontend > src > app > app.component.html > ...
Go to component
1 <app-navbar></app-navbar>
2
3
4 <div id="content" style="min-height: 100%" class="container mt-5">
5
6
7 <router-outlet *ngIf="(validToken$ | async) || router.url === this.cookie.get('lang')+'/manual' || router.url === '/legal'; else landing"></router-outlet>
8
9
10 <ng-template #landing>
11
12
13 <div class="text-center">
14 
15 <div class="alert alert-success" role="alert">
16
17 </div>
18 </div>
19 </ng-template>
```

Ok on va vraiment hostinger !



Votre mission si vous l'acceptez (ça parait dur... enfait c'est 5 lignes de code) :



Créer un nouveau component et le rendre accessible via le router (par flemme, on rentrera l'url à la main pour y accéder)



Créer une variable int, puis une fonction qui augmentent la valeur de cette variable dans le typescript. Faut qu'à chaque fois qu'on entre sur la page, la valeur de la variable soit reset à 3



Créer un bouton qui appelle la fonction. Puis afficher la variable dans le html

Aide

Créer un nouveau component

\$ ng generate component nomDuComponent

Afficher une variable du typescript dans le HTML :

{{ variable }}

Associer une méthode à un bouton :

<button (onClick)="maMethode()">

ngOnInit : une méthode qui s'exécute automatiquement à chaque fois qu'on charge le component

WoW c'est beaaau

The screenshot shows a web application interface with a dark header bar. On the left of the header is a logo with a blue 'X' and the word 'HOSTING' below it. To the right of the logo, the text 'MiNET Hosting' is displayed, followed by 'Version 1.2' in a light grey rounded rectangle, and 'Home' in a plain font. The main content area is white and contains three elements: the text 'test works!', a button with the text 'Test' inside a square border, and the number '3' below the button. Three colored arrows point from text annotations on the right to these elements: a blue arrow points to 'test works!', a green arrow points to the 'Test' button, and an orange arrow points to the number '3'.

MiNET Hosting Version 1.2 Home

test works!

Test

3

Texte par défaut que j'ai mis dans mon component

Bouton cliquable

Compteur initialisé à 3. Il augmente quand on appuie sur le bouton

Pour ceux qui sont plus motivés

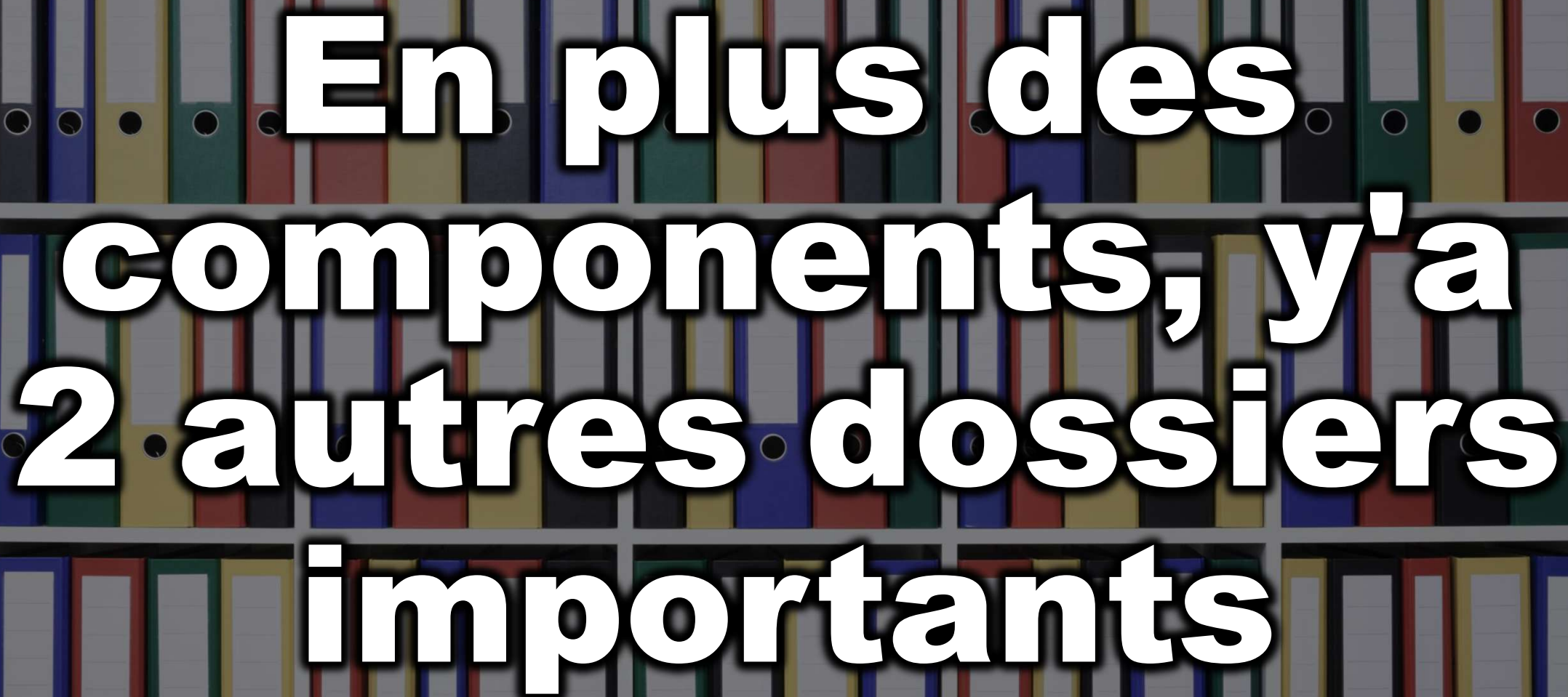
Faites 2 boutons : un pour like et un pour dislike

Du coup il faut aussi faire 2 compteurs pour like et dislike

On ne doit pas pouvoir like et dislike en même temps : si on en fait un ça annule l'autre

Lorsqu'on a like, le compteur s'affiche en vert plutôt que noir. Pareil mais en rouge pour le dislike.

On peut utiliser ngClass avec du CSS : `<div [ngClass]="{ nomPropCSS: condition}">`

A background image of a bookshelf filled with numerous colorful binders in shades of blue, green, red, and yellow. The binders are arranged in three rows, creating a dense, organized appearance.

**En plus des
components, y'a
2 autres dossiers
importants**



The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar shows a file tree with the 'models' folder expanded. The file 'vm.ts' is highlighted with a red box. The main editor area shows the content of 'vm.ts', which is a TypeScript class definition for 'Vm' with a constructor and several public properties.

```
1 export class Vm {
2   constructor(
3     public user?: string,
4     public name?: string,
5     public ram?: string,
6     public disk?: string,
7     public cpu?: string,
8     public type?: string,
9     public status?: string,
10    public autoreboot?: string,
11    public id?: number,
12    public password?: string,
13    public sshKey?: string,
14    public ip?: string,
15    public ramUsage?: string,
16    public cpuUsage?: string,
17    public uptime?: string,
18    public lastBackupDate?: string,
19    public createdOn?: string,
20    public isUnsecure?: boolean,
21    public hasError?: boolean,
22  ) { }
23 }
24
25 }
```

Les modèles :

En gros ça permet de définir des classes avec des propriétés

Genre là je définis l'objet « Vm »

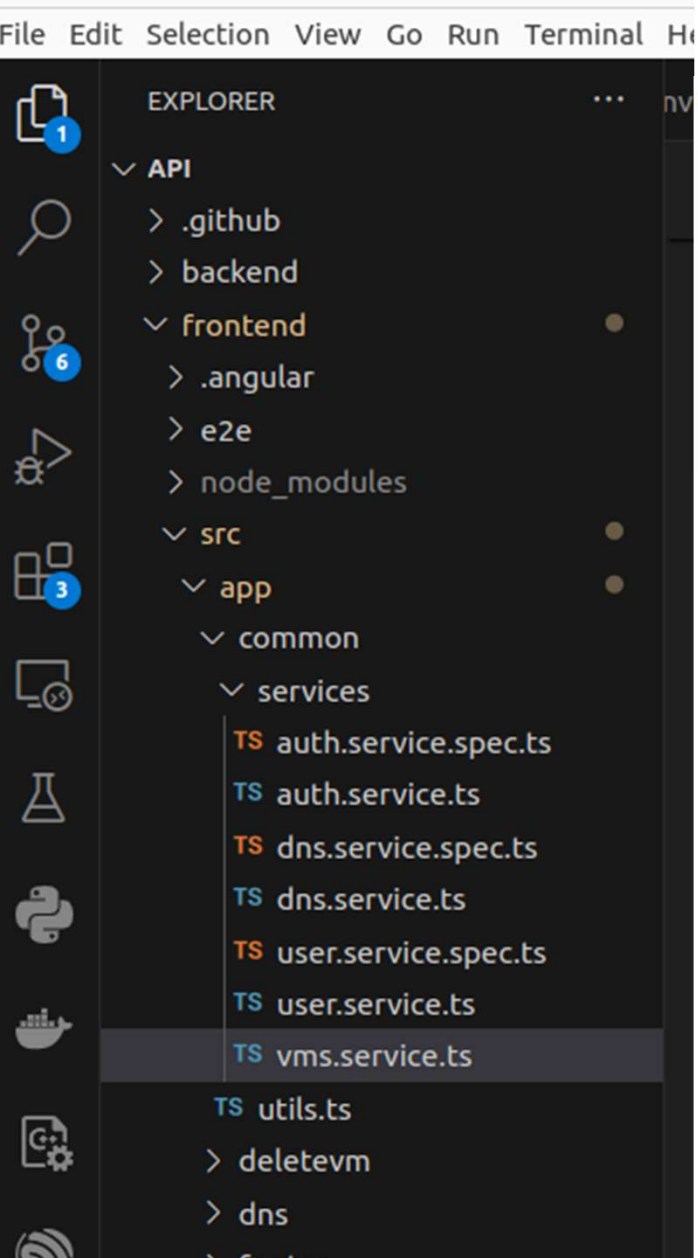
```
const expiredVms: Vm[] = [];
```

**Puis, je crée
un tableau de
VM**

```
vm.name = response.body['name'].trim();  
vm.status = response.body['status'];  
vm.ram = String(Math.floor(response.body['ram']/1000));  
vm.disk = response.body['disk'];  
vm.cpu = response.body['cpu'];  
vm.user = response.body['user'];  
vm.autoreboot = response.body['autoreboot'];  
vm.ramUsage = response.body['ram_usage'];  
vm.cpuUsage = response.body['cpu_usage'];  
vm.uptime = response.body['uptime'];  
vm.lastBackupDate = response.body['last_backup_date'];
```

**Et là je remplie
les propriétés
de ma VM**

**(cet exemple
c'est dans le
service « vms »)**



Ensuite y'a les services

Si je quitte puis revient sur un component, toutes ses variables seront réinitialisées à leur valeur de base

Les Services, ça contient des variables qui ne seront jamais réinitialisés tant qu'on ne quitte pas l'application.

Ça permet d'avoir des méthodes accessibles depuis n'importe quelle page, et de stocker des valeurs qu'on peut réutiliser sur tous les components.

Par exemple les infos sur les vms, on ne veut pas avoir à les récupérer à chaque fois qu'on change de page

Pour utiliser les services dans un autre component, il faut l'ajouter dans le constructeur du component

```
18  */
19  export class AppComponent implements OnInit{
20      title = 'Hosting';
21
22      constructor(public userService: UserService,
23                  public vmsService: VmsService,
24                  public dnsService: DnsService,
25                  public user: User,
26                  public router: Router,
27                  private titleService: Title,
28                  public cookie: CookieService)
29      {
30          const url = window.location.hostname;
```

Ok on va s'intéresser au backend maintenant

DU COUP IL VA FALLOIR OUVRIR LE DOSSIER BACKEND AUSSI

SAUF QUE SANS LES VARIABLES D'ENVIRONNEMENT DU .ENV, ON NE POURRA RIEN FAIRE DE CONCRET

PAR CONTRE ON PEUT QUAND MÊME VOIR RAPIDEMENT C'EST QUOI LA CHAÎNE D'INTERACTION POUR PASSER DU FRONTEND ANGULAR JUSQU'AUX VMS SUR PROXMOX

Voici un exemple typique pour récupérer les infos d'une VM

Essayez de retracer chaque fonction appelée :

A l'initialisation du **composant "vm"**, on appelle la **fonction typescript "updateVm"** dans le **service "vms"**

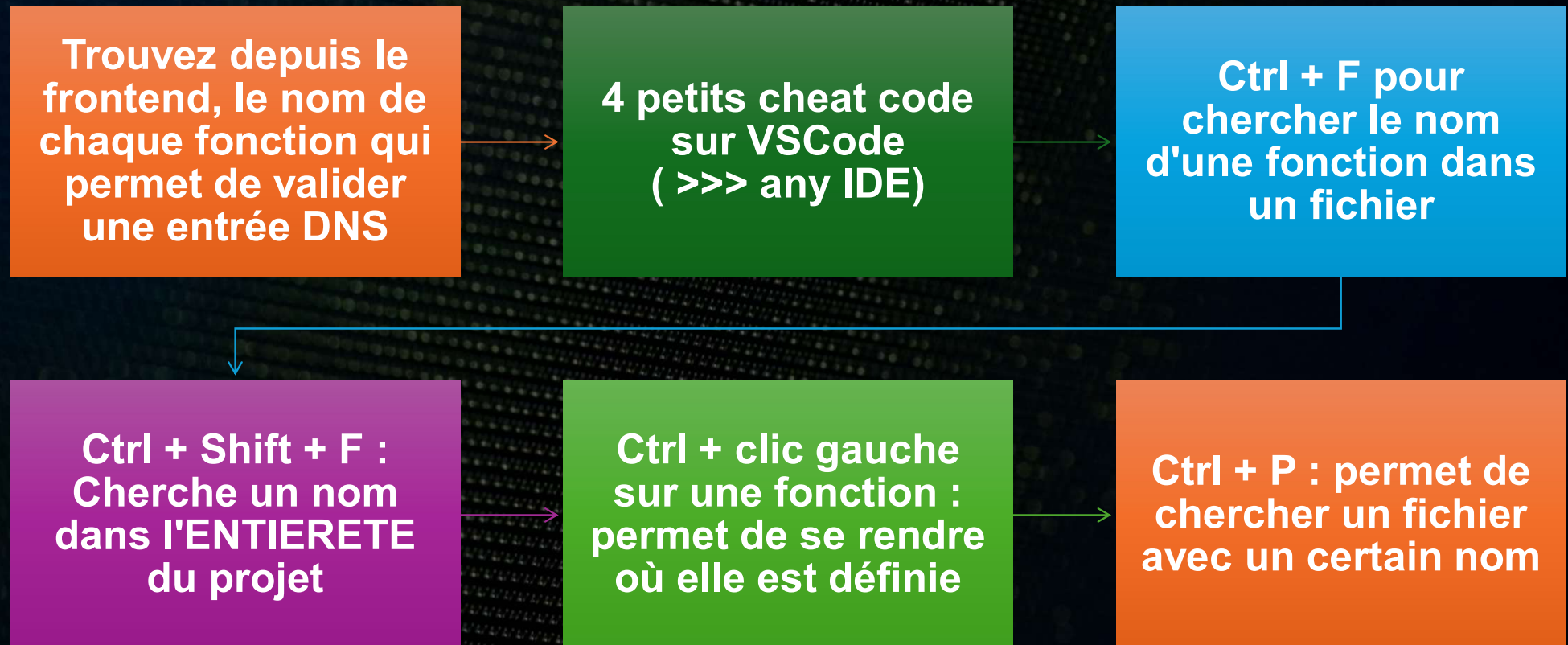
UpdateVm réalise la **requête http GET "vm/vmid"** à l'**API d'hosting** (regardez sur **api-hosting-dev.minet.net**)

Cet endpoint est défini dans **"proxmox_api > swagger"**

Le swagger appelle la **fonction "get_vm_id"** du **fichier "proxmox_api > controllers > default_controller"**

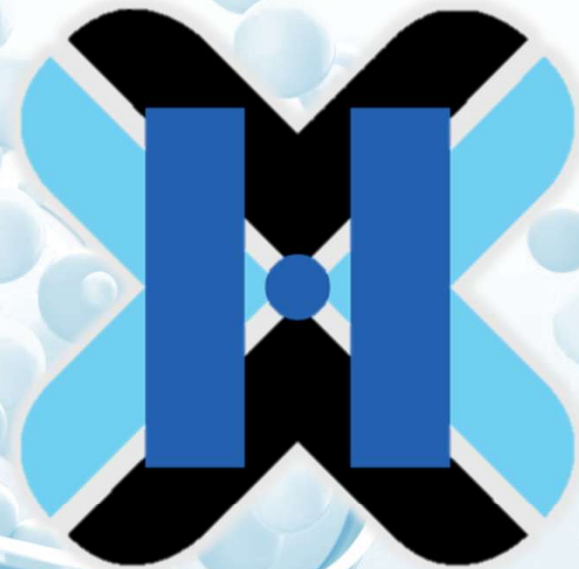
Elle fait appel à **pleins de fonction de "proxmox_api > proxmox"**

OK ce serait le dernier défi



C'est fini

**Allez regarder la
formation de
l'année dernière et
le wiki pour plus de
détails**



HOSTING

