Projet Bulles infra



**Rapport de Projet**

Zeqiri Amir – CIN2B

ETML, Vennes – A13

40 Périodes

Maître : M. Chenaux

Table des matières

[1 Description du projet dans son ensemble 3](#_Toc183610197)

[1.1 Titre 3](#_Toc183610198)

[1.2 Sujet 3](#_Toc183610199)

[2 RAISONNEMENT 3](#_Toc183610200)

[2.1 Choix OS, installation, configuration 3](#_Toc183610201)

[2.2 Choix, croquis application 3](#_Toc183610202)

[2.3 Conceptualisation base de données 3](#_Toc183610203)

[2.4 Installation outils (Docker, VS Code, .NET 8, AvaloniaUI) 3](#_Toc183610204)

[2.5 Création application et containers Docker 4](#_Toc183610205)

[2.6 Création base de données 4](#_Toc183610206)

[2.7 Programmation application 4](#_Toc183610207)

[3 Support de cours 5](#_Toc183610208)

[3.1 Installation de la VM Linux Ubuntu 5](#_Toc183610209)

[3.2 Choix de l’application 8](#_Toc183610210)

[3.3 Base de données 9](#_Toc183610211)

[3.4 Installation des outils 10](#_Toc183610212)

[3.4.1 Docker 10](#_Toc183610213)

[3.4.2 Visual Studio Code 10](#_Toc183610214)

[3.4.3 .NET 8 11](#_Toc183610215)

[4 Sources & Aides 11](#_Toc183610216)

[5 Conclusion 11](#_Toc183610217)

# Description du projet dans son ensemble

## Titre

Création d’une application dockerisée

## Sujet

Faire une application "dockerisée" de deux containers avec .Net et interface graphique contenant une base de données sur Linux Ubuntu.

# RAISONNEMENT

## Choix OS, installation, configuration

En premier lieu, on va procéder au choix de l’OS avec la classe, puis je vais effectuer son installation et sa configuration. En même temps, je vais me documenter sur certaines fonctions de VirtualBox. Comme la virtualisation, les performances, la sécurité, les bonnes pratiques à avoir lors de la création d’une VM, etc.

Je réaliserai des instantanés sur Oracle VM VirtualBox à chaque étape de mon projet afin de ne rien perdre en cas de problème. En plus, j’exporterai ma machine virtuelle à la fin de chaque cours, à la fois pour sécuriser mes données et pour pouvoir continuer à travailler à la maison.

## Choix, croquis application

Je vais imaginer l’application que je souhaite développer. Pour cela, je vais réaliser un croquis et rédiger une explication que je soumettrai à la validation de mon professeur.

## Conceptualisation base de données

Ensuite, je vais conceptualiser la base de données sur Looping. Cette étape est importante pour bien comprendre le fonctionnement de l’application. Une fois la conceptualisation terminée, je pourrai récupérer le script SQL qui me permettra, plus tard, de créer la base de données dans MySQL.

## Installation outils (Docker, VS Code, .NET 8, AvaloniaUI)

L’installation des outils nécessaires est une étape clé de mon projet. Je vais installer Docker Engine en ligne de commande, contrairement à Docker Desktop que j’ai déjà utilisé auparavant.

Ensuite, j’installerai Visual Studio Code pour pouvoir coder mon application.

En même temps, j’installerai .NET 8, qui sera utilisé comme environnement de développement pour programmer mon application.

Je vais également installer AvaloniaUI, un framework permettant de créer des interfaces graphiques multiplateformes. Cet outil sera indispensable pour concevoir l'interface utilisateur de mon application.

Enfin, je vérifierai que tous ces outils sont correctement configurés et fonctionnent ensemble avant de passer à l’étape suivante.

## Création application et containers Docker

À cette étape, je vais préparer la base de mon application. Je vais créer la structure principale avec les pages importantes comme la connexion, l’inscription, et l’accueil. Je vais aussi organiser les fichiers et dossiers pour que tout soit bien rangé et clair.

Comme indiqué par l’enseignant, je vais créer plusieurs containers Docker. Un premier pour le build de l’application, un second pour les tests, un troisième pour la base de données et un dernier pour déployer l’application.

## Création base de données

Dans cette étape, je vais utiliser le script SQL créé lors de la conceptualisation sur Looping pour générer ma base de données dans MySQL.

Je vais m’assurer que toutes les tables et la relation (clés primaires et étrangères) sont bien mises en place.

Je vais ensuite ajouter des données fictives pour tester les fonctionnalités principales, comme ajouter, modifier, supprimer ou afficher des informations (CRUD).

Je configurerai aussi la connexion entre la base de données et mon application via un conteneur Docker. Pour ça, je vais régler les paramètres nécessaires, comme les utilisateurs, les permissions, et les variables d’environnement.

Une fois terminé, je testerai la base pour m’assurer qu’elle marche bien avec l’application et je corrigerai les éventuels problèmes.

## Programmation application

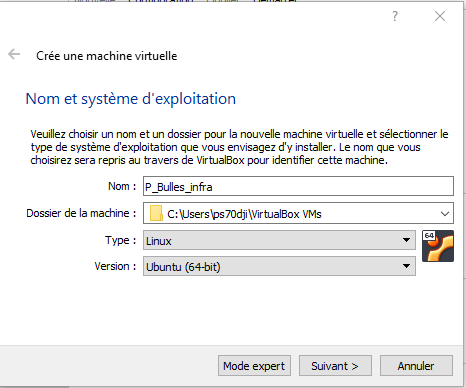
Une fois que tout sera prêt, je vais vraiment commencer à programmer l’application. Je vais coder les fonctionnalités principales comme la gestion des utilisateurs (inscription, connexion, modification), la création et la gestion des tâches, et l’affichage des données depuis la base de données.

Je vais tester chaque partie pour m’assurer que ça fonctionne bien et corriger les problèmes si besoin.

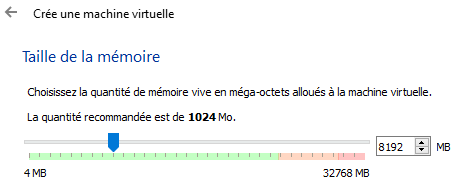
# Support de cours

## Installation de la VM Linux Ubuntu

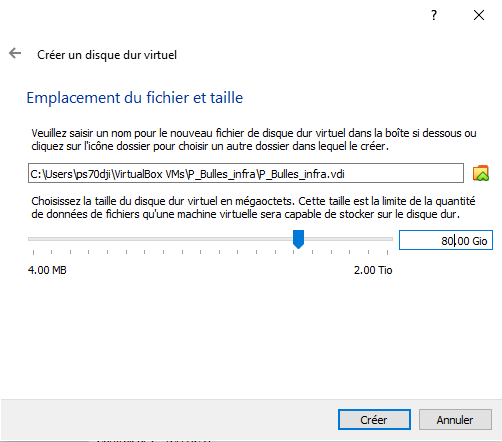
Choisir Linux Ubuntu de 64 bits



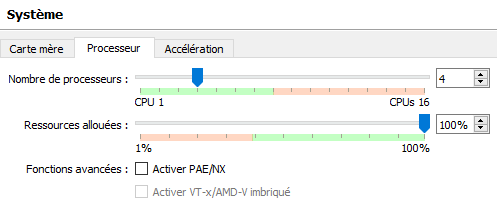
Pour que la VM puisse correctement fonctionner, il est important de choisir une quantité minimale de mémoire (4Go). Selon moi, comme nous possédons des machines à 32Go de RAM, nous pouvons nous permettre d’en choisir 8 pour la VM.



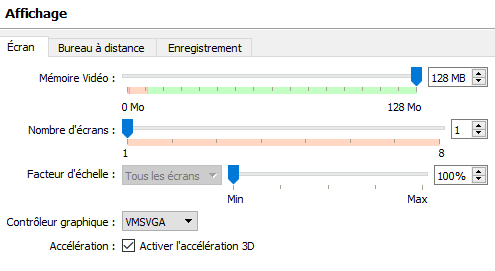
Allouer dynamiquement 80Go. Le minimum requis est de 25Go pour le bon fonctionnement. Comme je vais installer plusieurs logiciels et coder, j’ai mis une taille bien au-dessus.



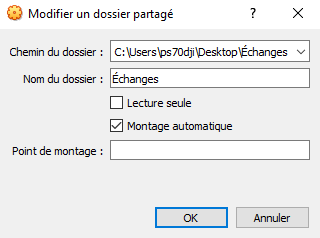
Le minimum de cœurs est de 2. J’ai mis le double comme le PC en possède 16.



Pour la mémoire vidéo j’ai décidé de mettre le maximum (128Mo). Et j’ai activé l’accélération 3D.



Pour pouvoir échanger des fichiers entre le PC et la VM, j’ai créé un dossier partagé. J’ai également activé le presse-papier partagé et le Glisser-Déposer entre machine.



Dans les fonctions avancées du processeur, on trouve PAE/NX et VT-x/AMD-V imbriqué.



**PAE** (Physical Address Extension) permet d'accéder à plus de 4 Go de RAM dans les systèmes 32 bits.

**NX** (No-eXecute) améliore la sécurité en empêchant l’exécution de code dans les zones de mémoire non exécutables.

**VT-x**/**AMD-V** imbriqué (Virtual Technology-variants / AMD-Virtualisation) Permettent la virtualisation dans la machine virtuelle, il est possible en activant cette fonction de créer des VMs dans une VM. VT-x sert pour les processeurs Intel tandis que AMD-V pour les processeurs AMD.

Comme cette fonction est grisée, je dois passer par PowerShell et l’activer en lignes de commande. Je dois, en premier temps, me déplacer dans les fichiers de VirtualBox avec cette commande :

cd 'C:\Program Files\Oracle\VirtualBox\'

Et cette commande activera la fonction :

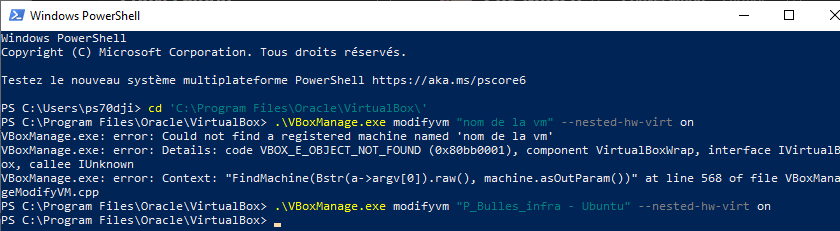
.\VBoxManage.exe modifyvm "P\_Bulles\_infra - Ubuntu" --nested-hw-virt on

**VBoxManage.exe** : C'est le programme de commande de VirtualBox.

**modifyvm** : Indique qu'on veut modifier une VM existante.

**"P\_Bulles\_infra - Ubuntu"** : C'est le nom de la VM à modifier.

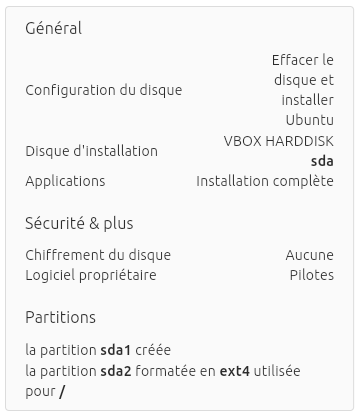
**--nested-hw-virt on** : Active la virtualisation imbriquée expliquée si-dessus.



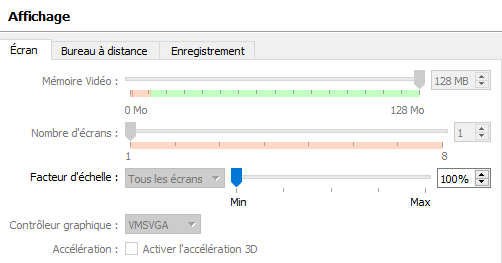
Voici la configuration de Linux Ubuntu :

Nom : amir

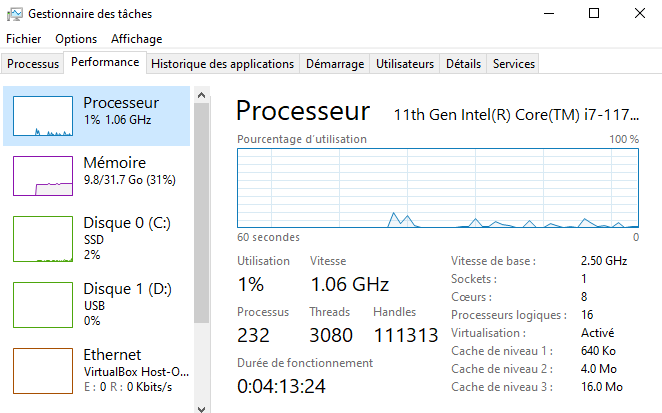
Mot de passe : 123



Après l’installation et la configuration de la VM, des bugs d’affichage étaient présents. Pour ce faire, il suffit de désactiver l’accélération 3D dans les paramètres d’affichage de VirtualBox.



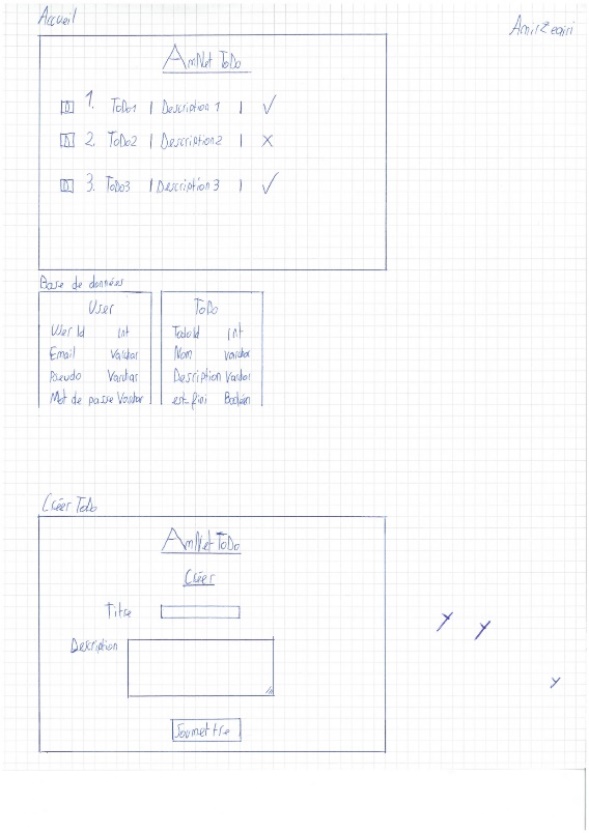
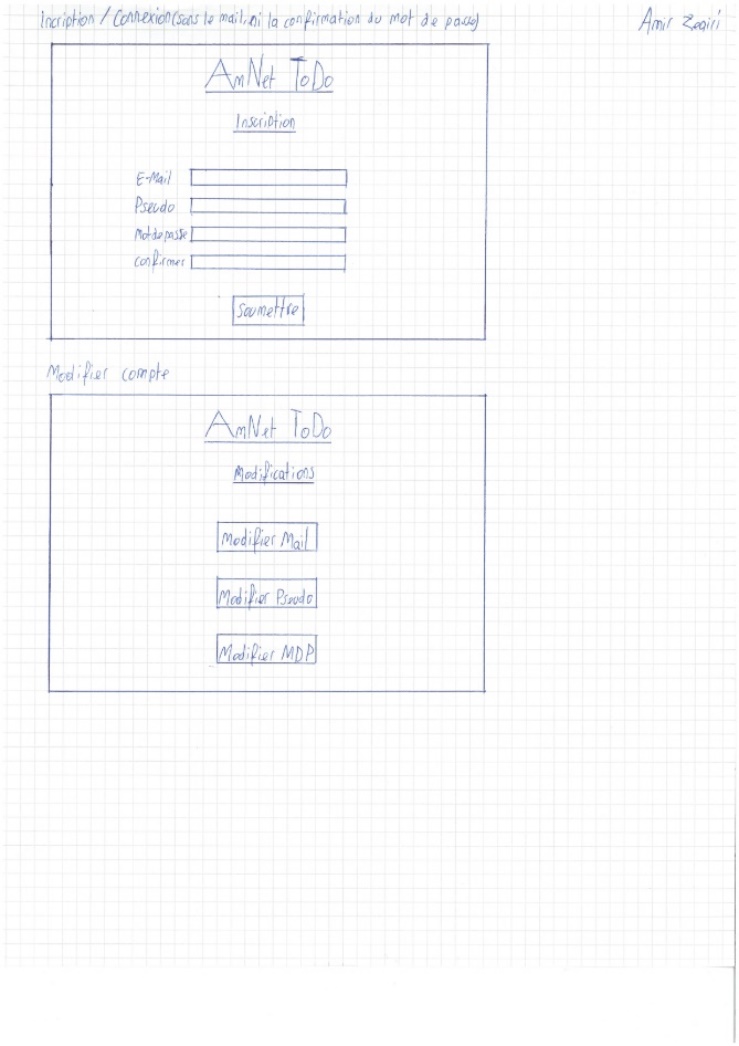
Pour vérifier si la virtualisation du PC est activée, je me suis rendu dans le gestionnaire des tâches > Performance > Processeur et il est écrit si oui ou non la virtualisation est activée.



## Choix de l’application

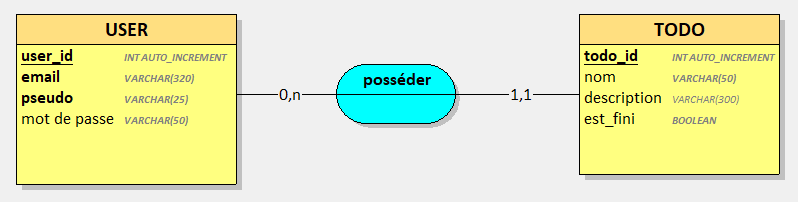
J’ai choisi de faire une application de ToDo list qui s’appelle "AmNet ToDo". Elle aura plusieurs pages : inscription, connexion, modification du compte, création de tâches et une page pour afficher les tâches de l’utilisateur. L’appli sera développée en C# avec .NET en interface graphique (Windows Form).

Voici les croquis faits sur papier :



## Base de données

Modèle conceptuel de données (MCD) :



Modèle logique de données (MLD) :



Dans la base de données de l’application, se trouve deux tables et une relation. La première table est nommée ‘t\_user’ et contiendra les informations de l’utilisateur, la deuxième est nommée ‘t\_todo’ et contiendra les informations de la tâche todo. La relation entre les deux tables est nommée ‘t\_posseder’.

Un utilisateur peut posséder aucune ou plusieurs tâches todo, tandis qu’elle, est possédée par un seul utilisateur obligatoirement. Certaines données sont obligatoires et uniques, comme le l’email, le pseudo, etc. Certaines sont uniquement obligatoire mais pas forcément unique, comme le nom de la tâche, le mot de passe de l’utilisateur, etc.  
La base de données a été conceptualisée en respectant les normes de conventions de codage de l’ETML et sera stockée dans un des deux containers Docker.

## Installation des outils

### Docker

Dans le terminal de Ubuntu, j’ai lancé plusieurs commandes qui installent Docker. Ces commandes je les ai trouvés dans le site de Docker [sourcé à la fin du rapport](#_Sources_&_Aides).

Voici les commandes :

Ces lignes-là, installe Docker

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

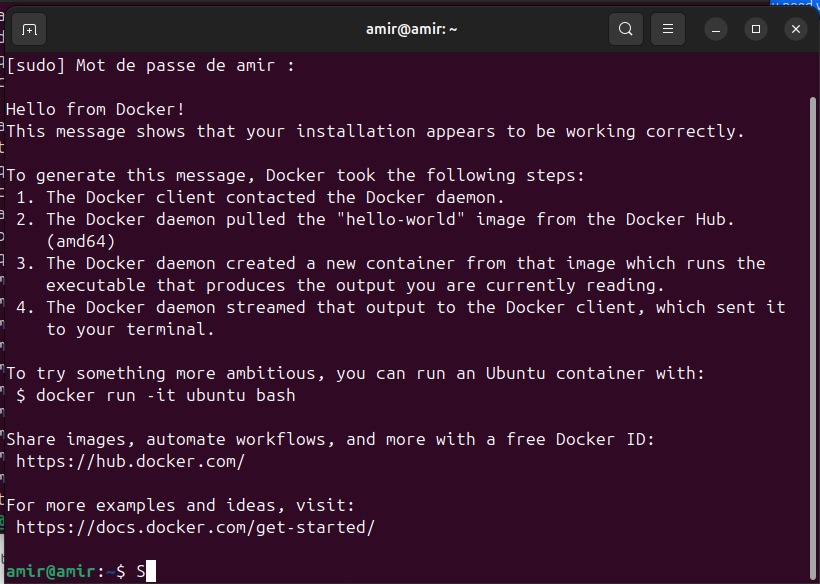
sudo apt-get update

Cette ligne-sous, installe, les packages Docker

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

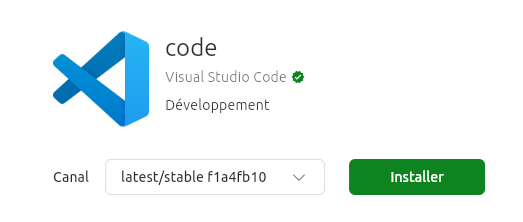
Cette ligne-ci dessous, vérifie l’installation en exécutant l’image ‘Hello World’

sudo docker run hello-world



### Visual Studio Code

Je me suis rendu dans le centre d’application, sur code et j’ai lancé l’installation.



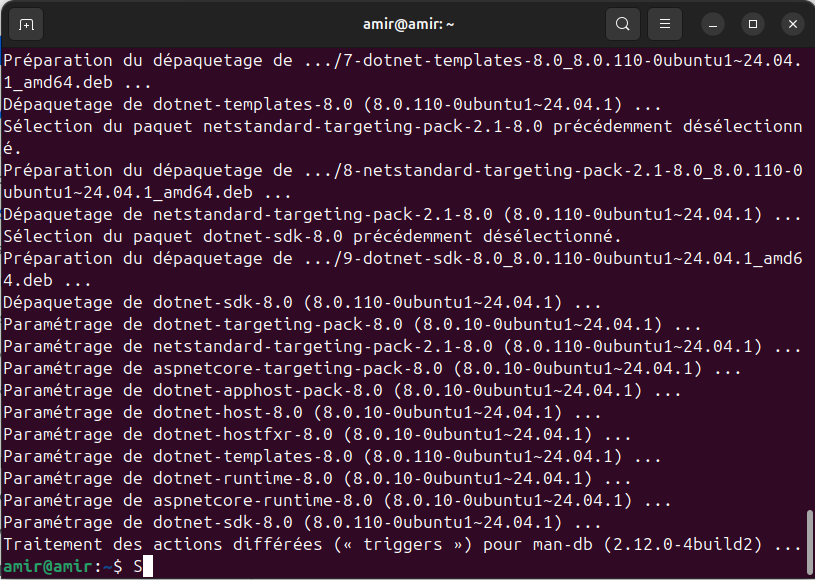
### .NET 8

Dans le terminal de Ubuntu, j’ai lancé une commande qui installent .NET 8. Cette commande je l’ai trouvé dans le site de Microsoft Learn [sourcé à la fin du rapport](#_Sources_&_Aides).

Voici les commandes qui installe .NET 8 :

sudo apt-get update && \

sudo apt-get install -y dotnet-sdk-8.0



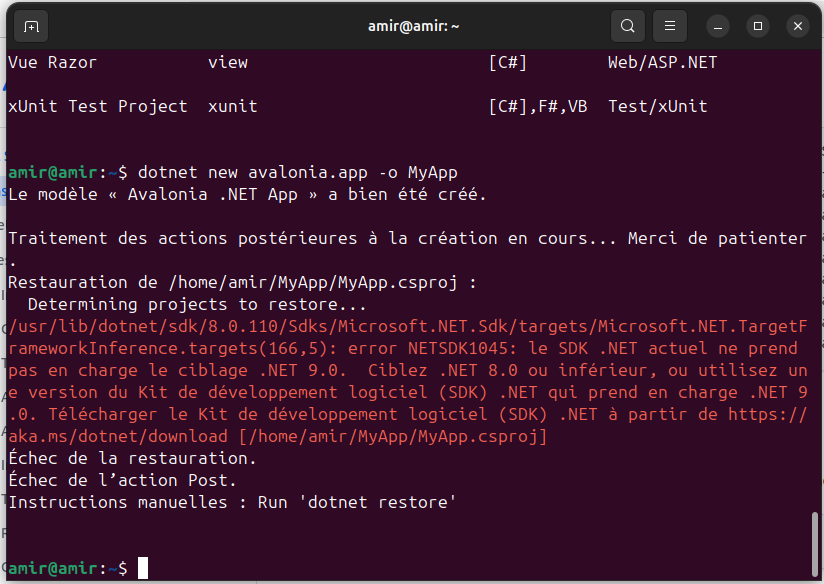
### AvaloniaUI

Je me suis rendu sur le site de AvaloniaUI [sourcé à la fin du rapport](#_Sources_&_Aides) pour suivre les étapes d’installation. J’ai commencé par installer le framework en lançant certaines commandes

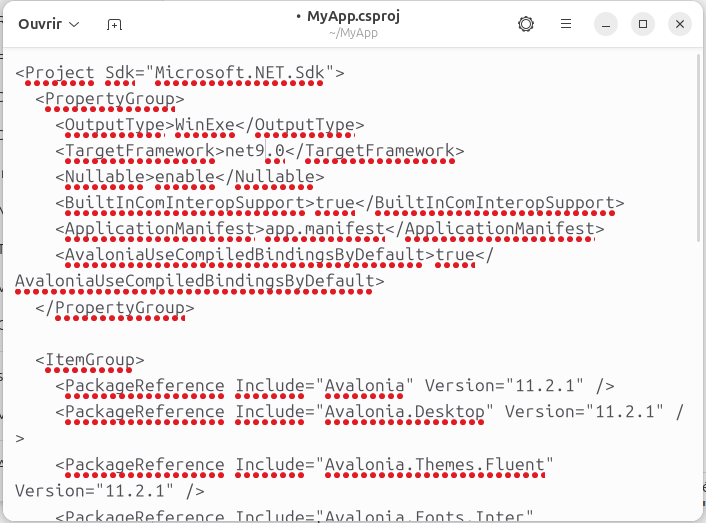
Voici les commandes :  
  
Ces lignes-là, installe les templates de Avalonia et crée une liste   
dotnet new install Avalonia.Templates  
dotnet new list

Cette commande crée une nouvelle app  
dotnet new avalonia.app -o MyApp

Après avoir exécuté ces commandes, un message d’erreur m’est apparu :



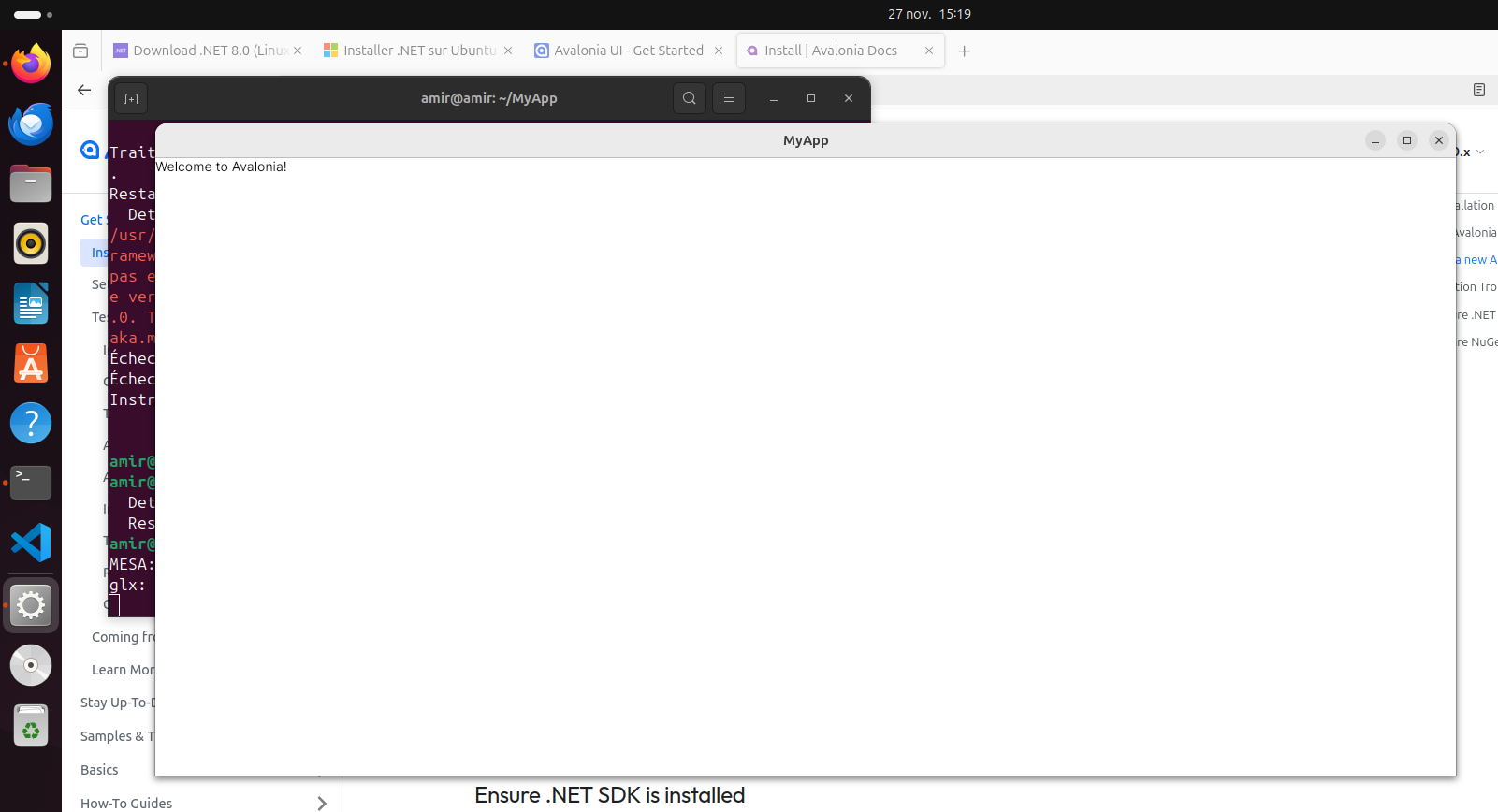
On voit que l’erreur est due à la version du .NET. Avalonia ne supporte pas .NET 9, pourtant j’ai installé la version 8. Pour cela je me suis rendu dans MyApp.csproj et j’ai modifié la version de manière brute.



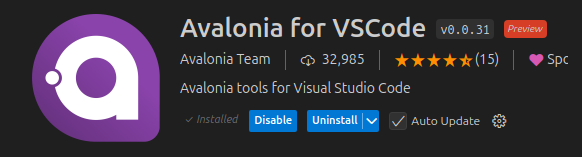
À la place de 9.0 (entouré ci-dessus), je mets 8.0. Ensuite je continue à exécuter les prochaines commandes :

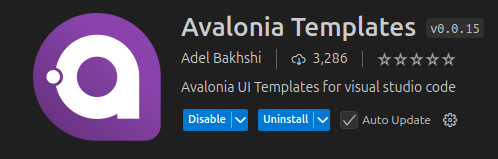
cd MyApp  
dotnet restore  
dotnet run

Et voici l’application lancée avec Avalonia et .NET8 :



Sur Visual Studio Code, j’ai installé les deux extensionsde Avalonia.



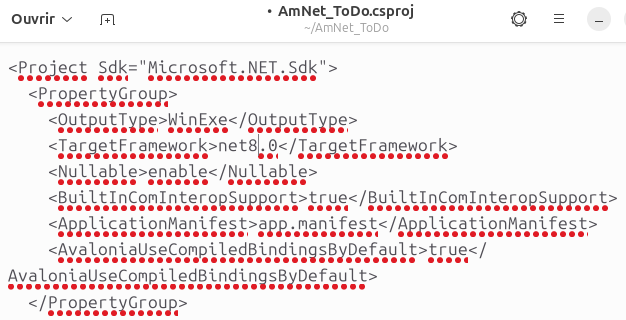


## Création de mon application

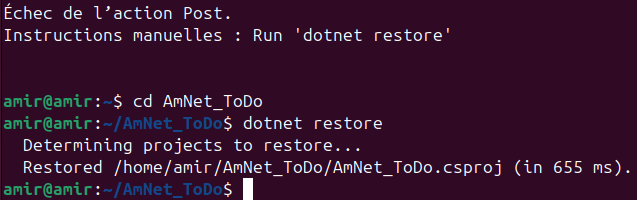
Je commence par créer mon application avec cette commande :



Après cette erreur, comme expliqué le chapitre précédent, je dois changer la version de .NET dans le fichier ‘csproj’ de mon application. Je mets donc 8.0 à la place de 9.0.



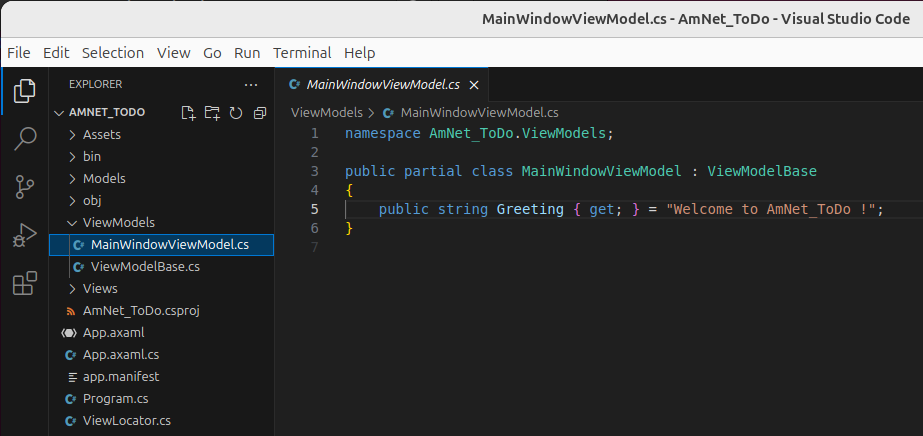
Je me place dans le dossier où se trouve mon projet et "restaure" l’application avec cette commande :



J’ouvre mon application sur Visual Studio Code avec cette commande :



Je modifie le texte de base et je suis prêt à "dockeriser" mon application

****

## "Dockerisation" de mon application

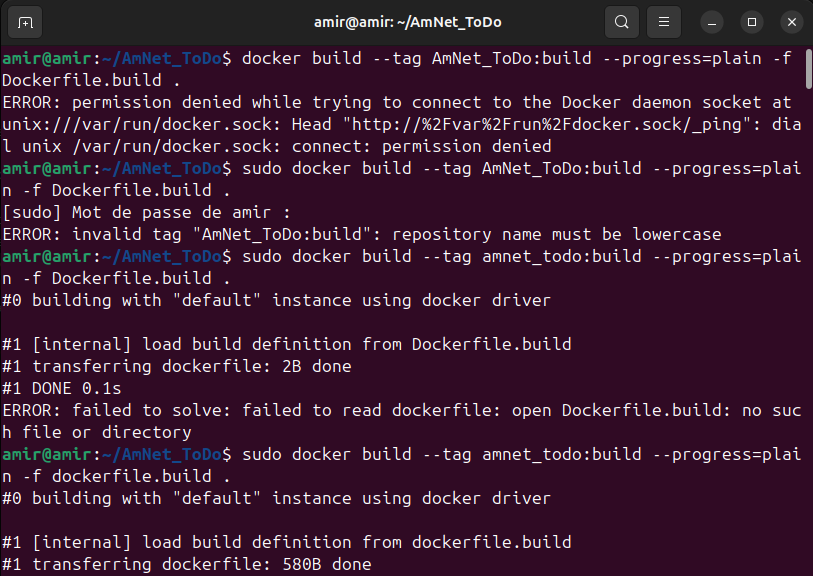
Pour "dockeriser mon application, il nous est demandé de créer trois containers. Un pour la compilation (build), un autre pour les tests, et un dernier pour le déploiement (deploy).

Commande pour créer l’image : build, test, deploy

sudo docker build --tag amnet\_todo:build --progress=plain -f dockerfile.build .

sudo docker build --tag amnet\_todo:test --progress=plain -f dockerfile.test .

sudo docker build --tag amnet\_todo:deploy --progress=plain -f dockerfile.deploy .



Commande pour créer le container à partir de l’image : build, test, deploy

sudo docker run -it --name amnet\_todo-build-container amnet\_todo:build

sudo docker run -it --name amnet\_todo-test-container amnet\_todo:test

sudo docker run -it --name amnet\_todo-deploy-container amnet\_todo:deploy

Commande pour afficher les images docker :

sudo docker images

Commande pour afficher tous les containers :

sudo docker ps -a

Commande pour afficher containers actifs :

sudo docker ps

Commande pour lancer un container : build, test, deploy

sudo docker start amnet\_todo-build-container

sudo docker start amnet\_todo-test-container

sudo docker start amnet\_todo-deploy-container

Commande pour arrêter un container : build, test, deploy

sudo docker stop amnet\_todo-build-container

sudo docker stop amnet\_todo-test-container

sudo docker stop amnet\_todo-deploy-container

Commande pour effacer le cache :

sudo docker builder prune

Commande pour supprimer un container :

sudo docker rm amnet\_todo-build-container

Commande pour supprimer une image :

sudo docker rmi 8c2e654c9cee

Commande pour se placer dans un container :

sudo docker exec -ti ce410c8a8f8d bash

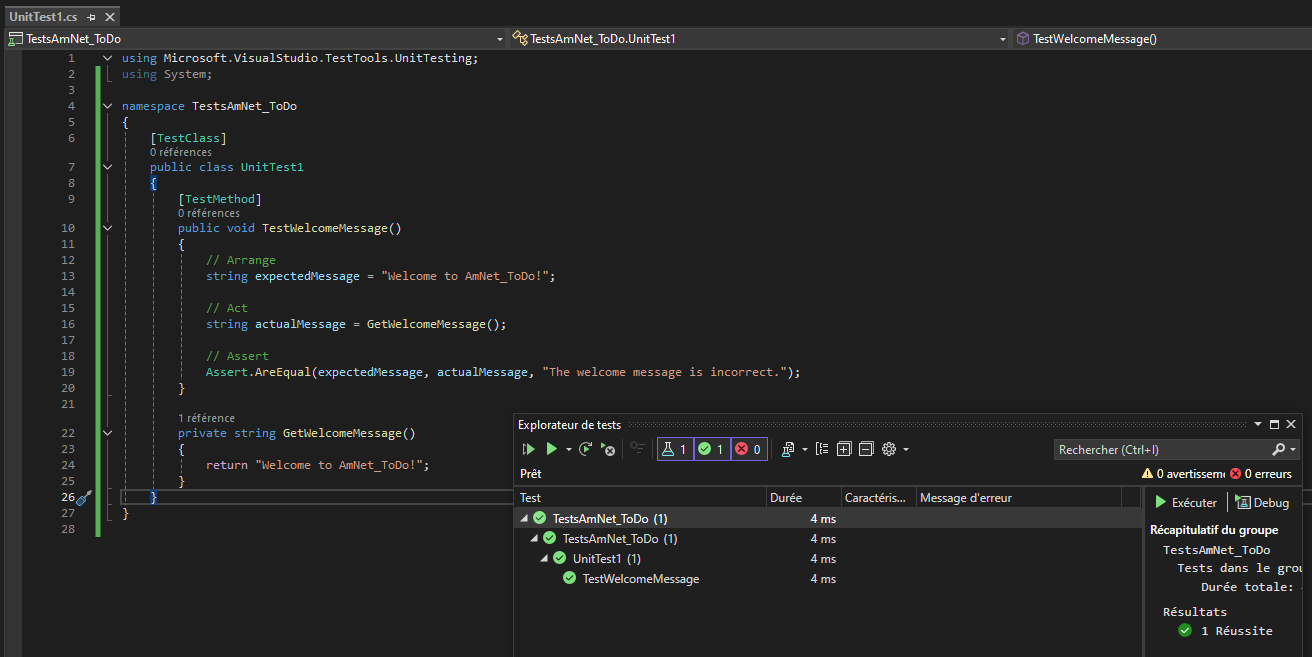
Commande pour lancer un test :

dotnet test

Commande pour lancer une application :

dotnet run

## Création du test unitaire



# Sources & Aides

[Ubuntu](https://ubuntu.com/download/desktop#system-requirements-NobleNumbat) : Aperçu de la configuration requise pour l’installation de Linux Ubuntu. Téléchargement du fichier ISO.

[Quora PAE/NX](https://www.quora.com/What-is-PAE-NX-in-VirtualBox) : Ce blog m’a permis de comprendre les fonctionnalités PAE/NX.

[It-connect](https://www.it-connect.fr/comment-activer-la-virtualisation-imbriquee-dans-virtualbox/), [Wikipédia](https://fr.wikipedia.org/wiki/Intel_VT): Ces deux sites m’ont permis de comprendre la virtualisation imbriquée.

[Superuser](https://superuser.com/questions/945910/how-to-select-paravirtualization-interface-in-virtualbox) : Ce blog m’a permis de comprendre les différentes interfaces de paravirtualisation.

[ChatGPT](https://chatgpt.com/c/6734b1df-c7f4-8008-99a6-180932d7d57f) : Cette IA m’a servi pour la conceptualisation de la base de données avec Looping.

[Conventions de codage ETML](https://eduvaud.sharepoint.com/sites/msteams_3cc8eb/Documents%20partages/Forms/AllItems.aspx?id=%2Fsites%2Fmsteams%5F3cc8eb%2FDocuments%20partages%2FGeneral%2F01%2Dnormes%2Dconventions%2FI%2DConventionsDeCodageV3%2E6%2E0%2Epdf&parent=%2Fsites%2Fmsteams%5F3cc8eb%2FDocuments%20partages%2FGeneral%2F01%2Dnormes%2Dconventions) : Lors de la conceptualisation de la base de données, j’ai fait appel à ce fichier pour nommer correctement mes tables.

[Docs Docker](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository) : La documentation de Docker m’a permis d’installer Docker Engine.

[Microsoft Learn](https://learn.microsoft.com/fr-fr/dotnet/core/install/linux-ubuntu-install?pivots=os-linux-ubuntu-2404&tabs=dotnet8#install-the-sdk) : Ce site m’a permis d’installer .NET 8

[Docs AvaloniaUI](https://docs.avaloniaui.net/docs/get-started) : La documentation de AvaloniaUI m’a permis d’installer le framework.

# Conclusion