

draft

GITHUB COPILOT WORKSHOP:

Auteur : [nom] – CELLENZA GO



cellenza

Developing with Github Copilot

AI pair programming

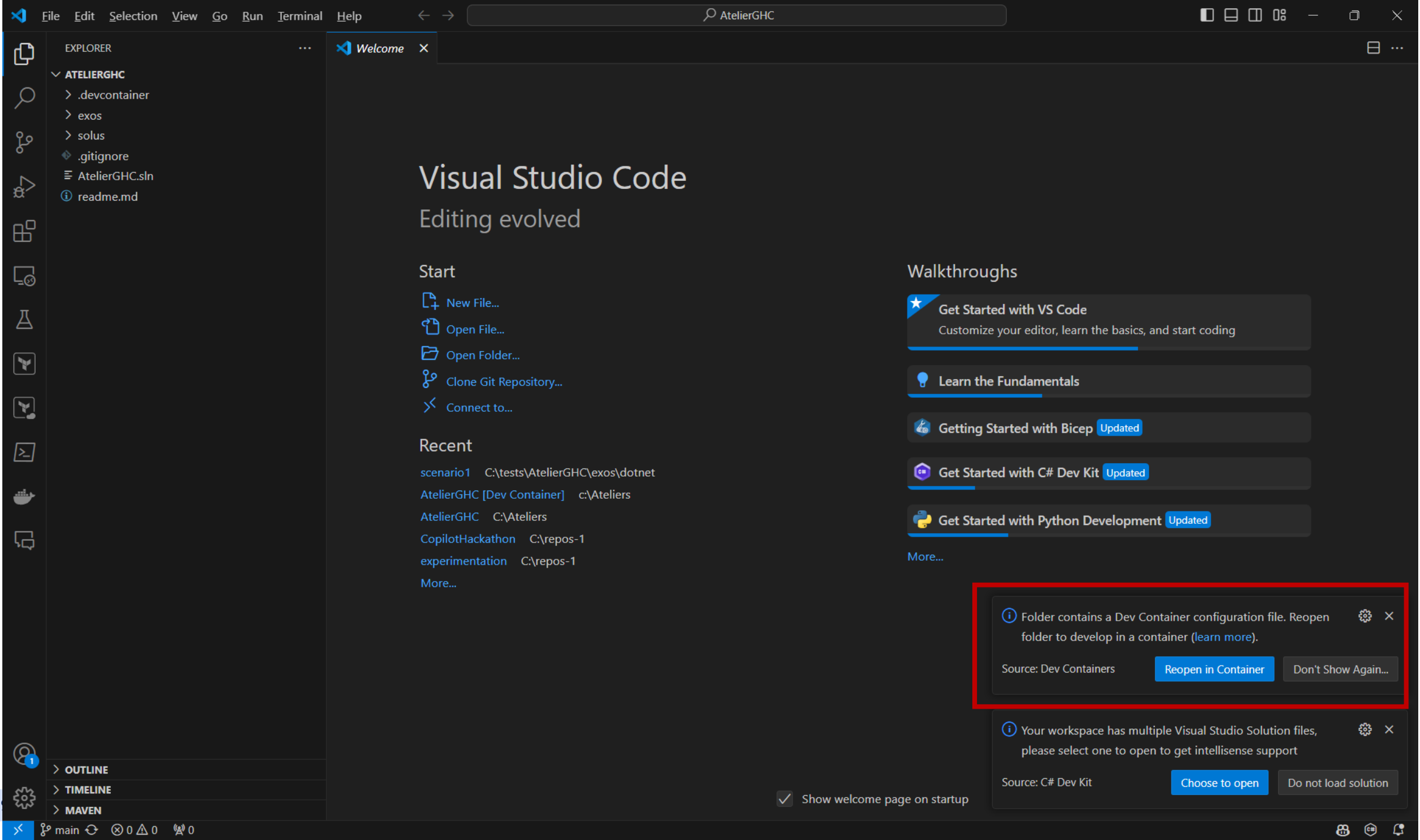
Developing with Github Copilot

- Adding features
- Document your code
- Creating Unit Tests
- Creating a docker image

If you have any errors, always ask Chat Copilot to help you !!

SC1: Exercise 1: Checking the dotnet project

- Clone the project : [<https://github.com/EricVernie/AtelierGHC.git>]
- Got the directory
 - \AtelierGC
- *Launch VS Code*
 - Type : *Code* .
- The first time VS Code ask you to "Reopen in container" *
- Go to the directory
 - .\exos\dotnet\scenario1\MinimalAPI
- Check that everything is working
 - In the VSCode terminal (*CTRL+ ù*)
 - `dotnet run --project .\MinimalAPI\MinimalAPI.csproj` (Windows)
 - `dotnet run --project ./MinimalAPI/MinimalAPI.csproj` (linux)



SC1: Exercice 2 : Creating a Hello World! EndPoint

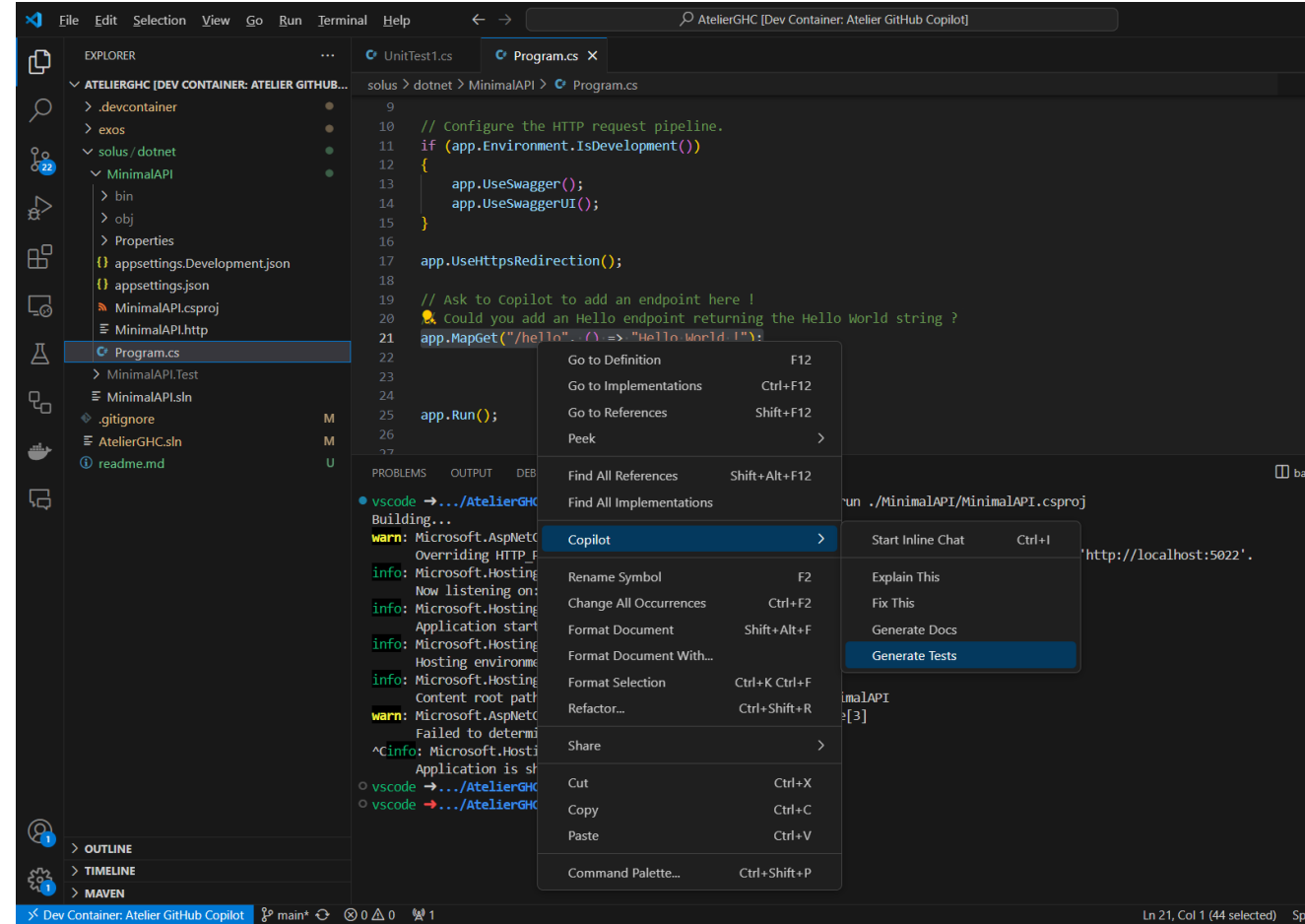
- Remember, it is possible to use Copilot either
 - By Chat 
 - `CTRL + i` (inline chat)
 - Typing in natural language
- Open the file *MinimalAPI\program.cs*
 - Prompt: "Add the /Hello endpoint that returns the Hello World string!"
- Start the project
 - `dotnet run --project ./MinimalAPI/MinimalAPI.csproj`
- Open the file *MinimalAPI\MinimalAPI.http*
 - Prompt: "Adds a call to the /Hello endpoint"
 - Test the call by clicking :send request

SC1 : Exercice 3 : Adding new features

- **/daysbetweendates**
 - Calculate the number of days between two dates, with parameters in query
- **/validateemail**
 - Validate the email using a regular expression
- **/openid**
 - Can you make a web api call using HttpClientFactory in order to retrieve the openid information from Microsoft Entra?
- **/parseurl**
 - Retrieves a parameter from querystring call someurl, parse the url and return the protocol, host, port, querystring and hach, return the parsed host
- **/tellmeajoke**
 - Calls jokeapi and returns a single joke, Deserialize to dynamic. Make sure to use IHttpClientFactory to create the HttpClient instance.
- **/randomeeuropeancountry**
 - Make a strings array of european country and its iso codes, return the country and its code from the array
- **/listfiles**
 - Return the list of files in the current directory
- **/memoryusage**
 - Returns memory consumption in GB rounded to 2 decimal places
- **It's up to you to try with other examples**

SC1: Exercise 4: Adding Tests

- Select the *daysbetween* EndPoint
- Right-click
 - Copilot ->Generate Tests
- Or
 - CTRL+ i, Type: /tests
- Generate the tests in the file
 - .\MinimalAPI.Test\ProgramTest.cs
- Start Testing
 - Dotnet test

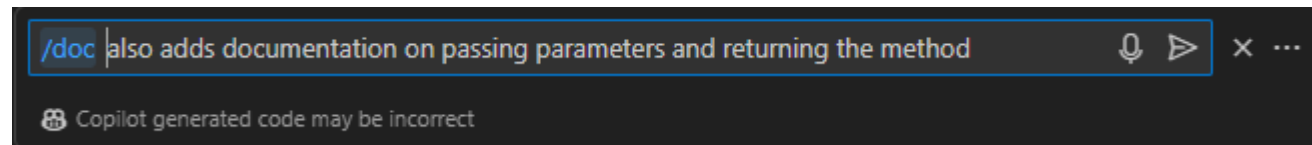


SC1: Exercise 5a: Document the Code

- Inside the *Program.cs* file

- ***ctrl + i***

- **Write:** `/doc` "also adds documentation on passing parameters and returning the method"



The screenshot shows a code editor with a dark theme. A text input field contains the text `/doc` followed by a space and the string "also adds documentation on passing parameters and returning the method". To the right of the input field are icons for voice search, a play button, a close button (x), and a menu button (three dots). Below the input field, a small icon of a person with a question mark is followed by the text "Copilot generated code may be incorrect".

SC1 : Exercice 5b: Add Documentation

- Create a new file called *readme.md*
- Open the file, Hit CTRL + I
- Type : *Add documentation for the daysbetween dates*

SC1 : Exercice 6 : Creating a docker container

- Open the *Dockerfile*

- Prompt :

- Build a docker image with .NET 8 and run the app on port 8080

- Open the file *buildimage.sh*

- Prompt :

- Build a Docker image and start a container on port 8080

- In the VS Code Terminal

- Run the command:

- `bash buildimage.sh`

ATELIER GITHUB COPILOT :

Auteur : [nom] – CELLENZA GO



cellenza

Développer avec Github Copilot

Votre binôme de développement

Développer avec Github Copilot


- Ajout de fonctionnalités
- Documenter son code
- Création de tests unitaire
- Création d'une image docker

Si vous avez des erreurs, toujours demander à Chat Copilot de vous aider !!!

SC1: Exercise 1: Vérification du projet dotnet

- Clone the project : [<https://github.com/EricVernie/AtelierGHC.git>]
- Allez dans le répertoire
 - \AtelierGC
- **Lancez VS Code**
 - Type : *Code* .
- La première fois que VS Code vous demande de "Reopen in container" *
- Positionnez-vous dans le répertoire
 - .\exos\dotnet\scenario1\MinimalAPI
- Dans le terminal VS Code (**CTRL+ `**) tapez
 - `dotnet run --project .\MinimalAPI\MinimalAPI.csproj` (Windows)
 - `dotnet run --project ./MinimalAPI/MinimalAPI.csproj` (linux)

SC1: Exercice 2 : Création d'un EndPoint Hello World!

- **Rappelez-vous, il est possible de faire une demande soit**
 - Par le chat 
 - Soit en activant *CTRL + i* (Chat en ligne)
 - Soit dans une ligne de commentaire
- **Ouvrez le fichier *MinimalAPI\program.cs***
 - Prompt: "Ajoute-le endpoint /Hello qui retourne la chaine Hello World !"
- **Démarrez le projet**
 - `dotnet run --project ./MinimalAPI/MinimalAPI.csproj`
- **Ouvrez le fichier *MinimalAPI\MinimalAPI.http***
 - Prompt: "Ajoute un appel au endpoint /Hello"
 - Testez l'appel en cliquant sur: `send request`

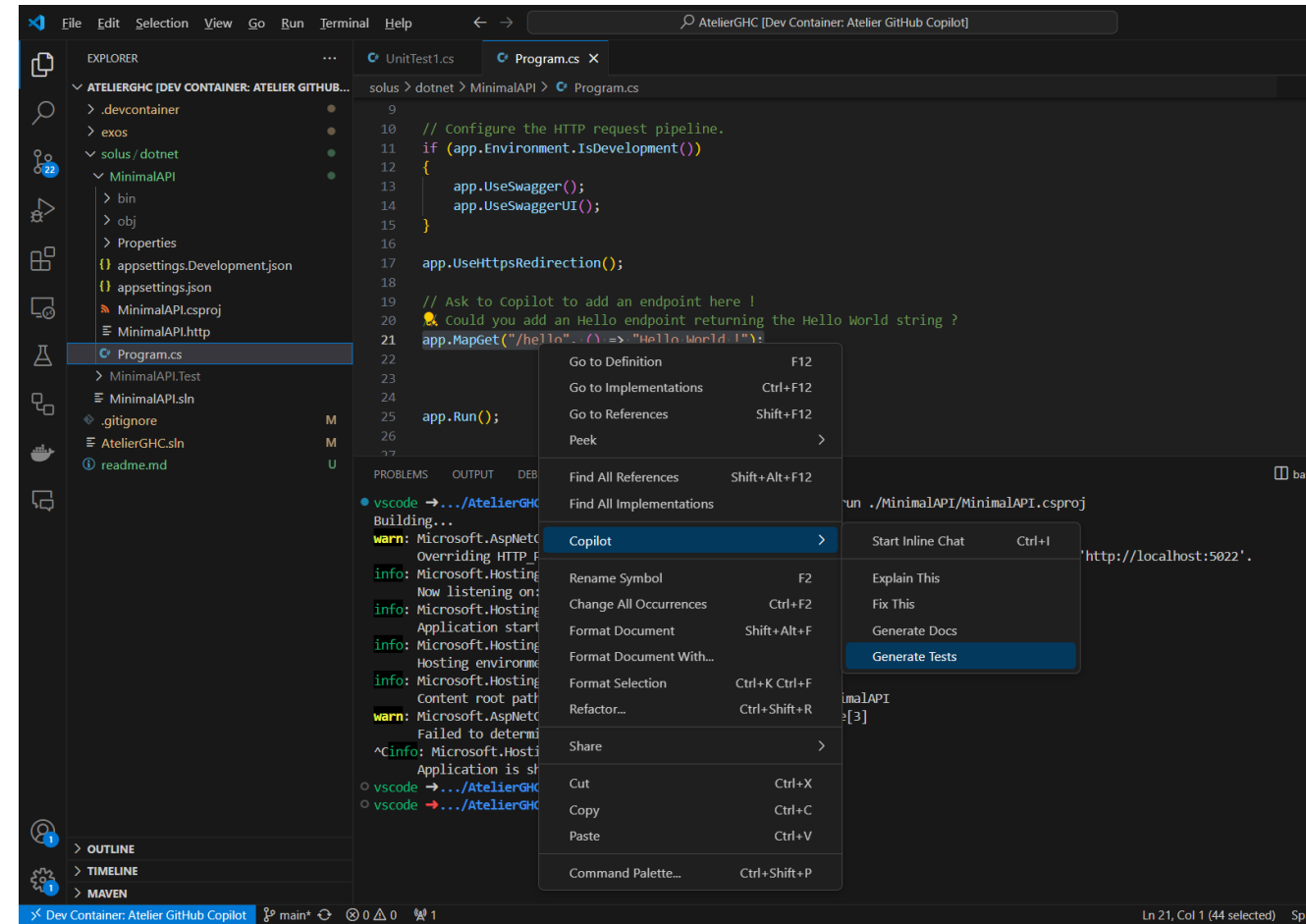
SC1 : Exercice 3 : Ajout de nouvelles fonctionnalités

- **/daysbetweendates**
 - Calcul de jours entre deux dates, avec passage de paramètre dans la requête
- **/validateemail**
 - Valider l'email en utilisant une expression régulière
- **/openid**
 - Peux-tu me faire un appel d'api web en utilisant HttpClientFactory afin de récupérer les informations openid de Microsoft Entra ?
- **/parseurl**
 - Récupère un paramètre à partir de l'appel de chaîne de requête someurl, analyse l'url et récupère le protocole, l'hôte, le port, la chaîne de requête et hash, renvoie l'hôte analysé.
- **/tellmeajoke**
 - Appelle jokeapi et renvoie une seule blague, Désérialisée en dynamique. Assurez-vous d'utiliser IHttpHttpClientFactory pour créer l'instance HttpClient.
- **/randomeuropeancountry**
 - Faire un tableau de pays européen et ses codes iso, renvoyer le pays et son code à partir du tableau
- **/listfiles**
 - Retourner la liste des fichiers du répertoire courant
- **/memoryusage**
 - Retourne la consommation mémoire en GO arrondie à 2 décimal

- **A vous d'essayer avec d'autres exemples**

SC1: Exercice 4 : Ajout de tests

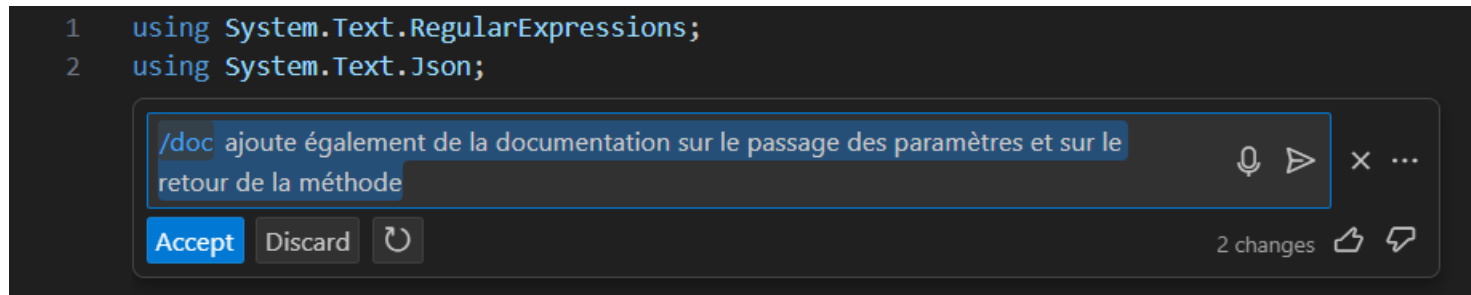
- Sélectionnez le EndPoint *daysbetween* dates
- Cliquez droit
 - Copilot ->Generate Tests
- Ou
 - *CTRL+ i* , Tapez : */tests*
- Générez les tests dans le fichier
 - *.\MinimalAPI.Test\ProgramTest.cs*
- Démarrez les tests
 - Dotnet test



SC1 : Exercice 5 : Documentez le code

- Dans le fichier *Program.cs*

- Invoquez copilot : **ctrl + i**
- Tapez : **/doc** "ajoute également de la documentation sur le passage des paramètres et sur le retour de la méthode"



The screenshot shows a Visual Studio code editor with a dark theme. The code editor contains two lines of C# code:

```
1 using System.Text.RegularExpressions;
2 using System.Text.Json;
```

Below the code editor, a Copilot chat window is open. The input field contains the text: `/doc` ajoute également de la documentation sur le passage des paramètres et sur le retour de la méthode. The chat window has a blue border and a blue background. At the bottom of the chat window, there are three buttons: "Accept" (highlighted in blue), "Discard", and a refresh icon. To the right of the buttons, it says "2 changes" followed by a thumbs up and a thumbs down icon.

SC1 : Exercice 6 : Création d'un conteneur docker

- Ouvrez le fichier Dockerfile

- Prompt :

- Construit une image docker avec .NET 8 et exécute l'app sur le port 8080

- Ouvrez le fichier buildimage.sh

- Prompt :

- build une image docker et démarre un container sur le port 8080

- Dans le terminal VS Code

- Exécutez la commande :

- `bash buildimage.sh`

SC1: Correction d'erreur/ Explication de code

- Dans le fichier `.\MinimalAPI\Program.cs`, cherchez le EndPoint *parallelloop*
 - La méthode incrémente une variable `i` dans une boucle qui a la particularité de s'exécuter sur plusieurs tâches, enfin elle retourne le résultat.
- Dans le terminal de VSCode (**CTRL+ ù**) exécutez la commande
 - `dotnet run -project ./MinimalAPI/MinimalAPI.csproj`
- Ouvrez le fichier `.\MinimalAPI\MinimalAPI.http` ajoutez les lignes suivantes :


```
###  
Get http://localhost:5022/parallelloop?iteration=1000000
```
- Cliquez sur : **Send Request**
- Que constatez-vous ?
- Pour corriger le problème, sélectionnez le EndPoint puis **CTRL+ i /fix**
 - Cette commande `/fix` cherchera à corriger l'erreur de programmation
- Tapez `/explain` pour une explication du code

SC1: Error fix

- In the `.\MinimalAPI\Program.cs` file, look for the `parallelloop` endpoint
 - The method increments a variable `i` in a loop that has the particularity of running on several tasks, finally it returns the result.
- In the VSCode terminal (**CTRL+ ù**) run the command
 - `dotnet run -project ./MinimalAPI/MinimalAPI.csproj`
- Open the `.\MinimalAPI\MinimalAPI.http` file and add the following lines:

###

Get <http://localhost:5022/parallelloop?iteration=1000000>

- **Click: Send Request**
- What do you see?
- To fix the problem, select the EndPoint and then **CTRL+ i /fix**
 - This `/fix` command will attempt to fix the programming error
- Type `/explain` for an explanation of the code



cellenza

Paris | Lyon | Nantes
Tél. +33(0)1 45 63 14 29
cellenza.com

Rejoignez-nous sur :

