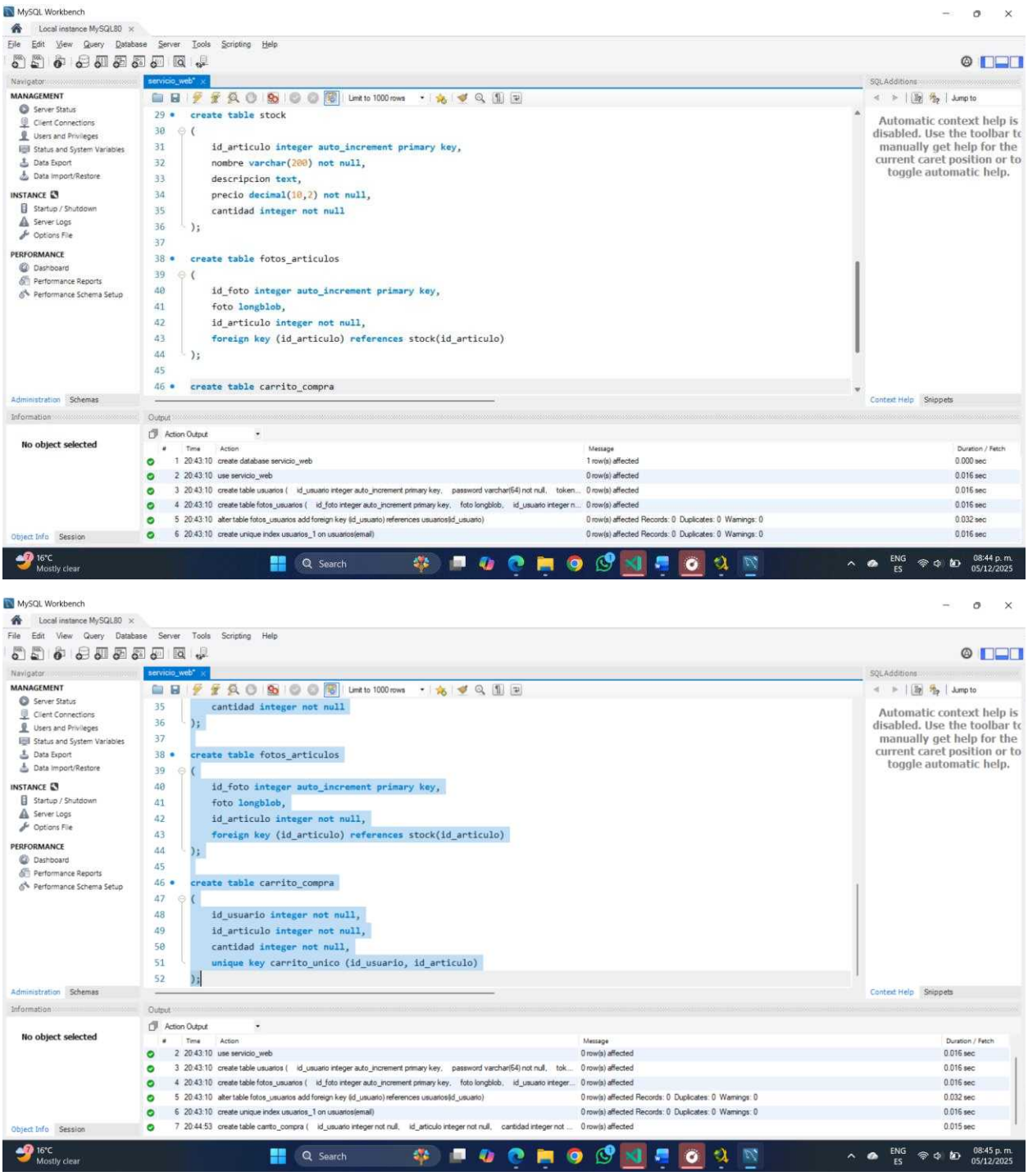


Figura 3. Creación de tablas stock, fotos_articulos y carrito_compra con sus llaves y restricciones.

4.5 Creación de usuario local y privilegios

Se creó el usuario local para desarrollo y se le otorgaron privilegios sobre la base “servicio_web”. Se accedió con este usuario para validar las operaciones del back-end.

create user x@localhost identified by '';

grant all on servicio_web.* to x@localhost;

Usuario	Host	Privilegios
x@localhost	localhost	ALL en servicio_web.*

Tabla 4 Usuarios

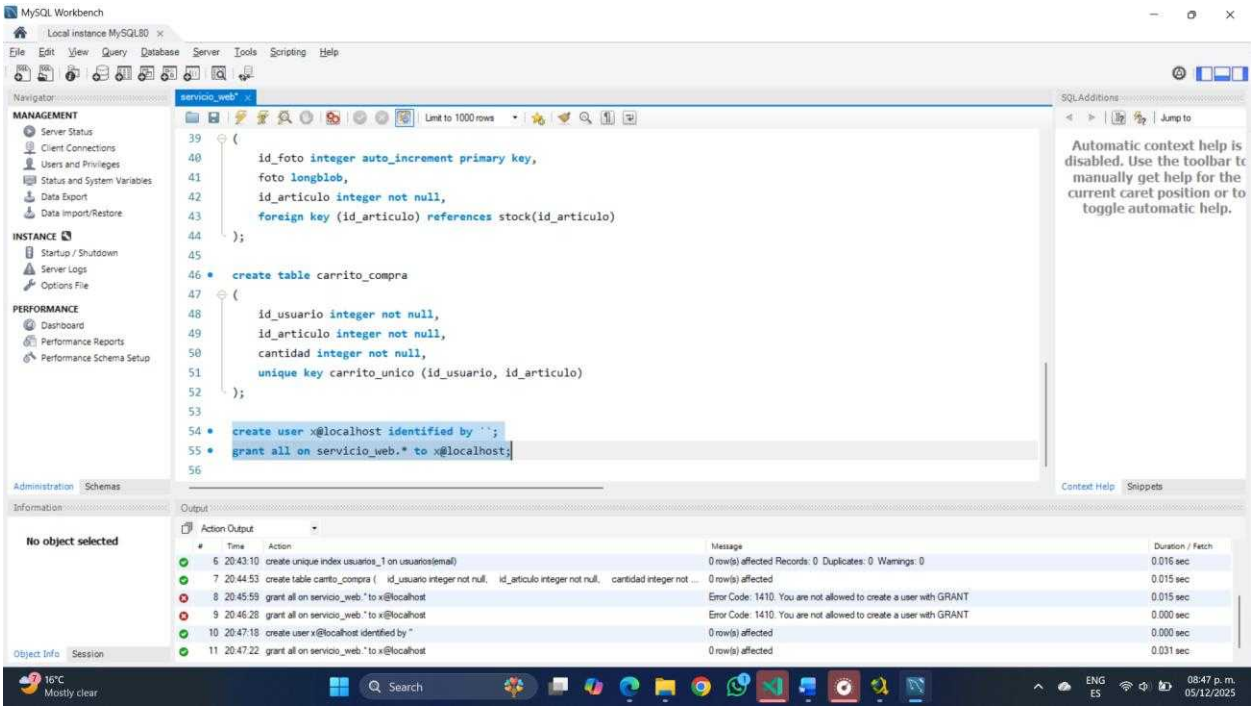


Figura 4. Confirmación de usuario local creado y concesión de privilegios en MySQL.

4.6 Instalación de Visual Studio Code

Se instaló Visual Studio Code desde el sitio oficial y se accedió al editor para instalar las extensiones necesarias: C# (Microsoft), Azure Functions y Azure Account. Se validó el inicio de sesión en Azure en caso de publicar posteriormente.

- Se descargó VS Code y se instaló con opciones por defecto.

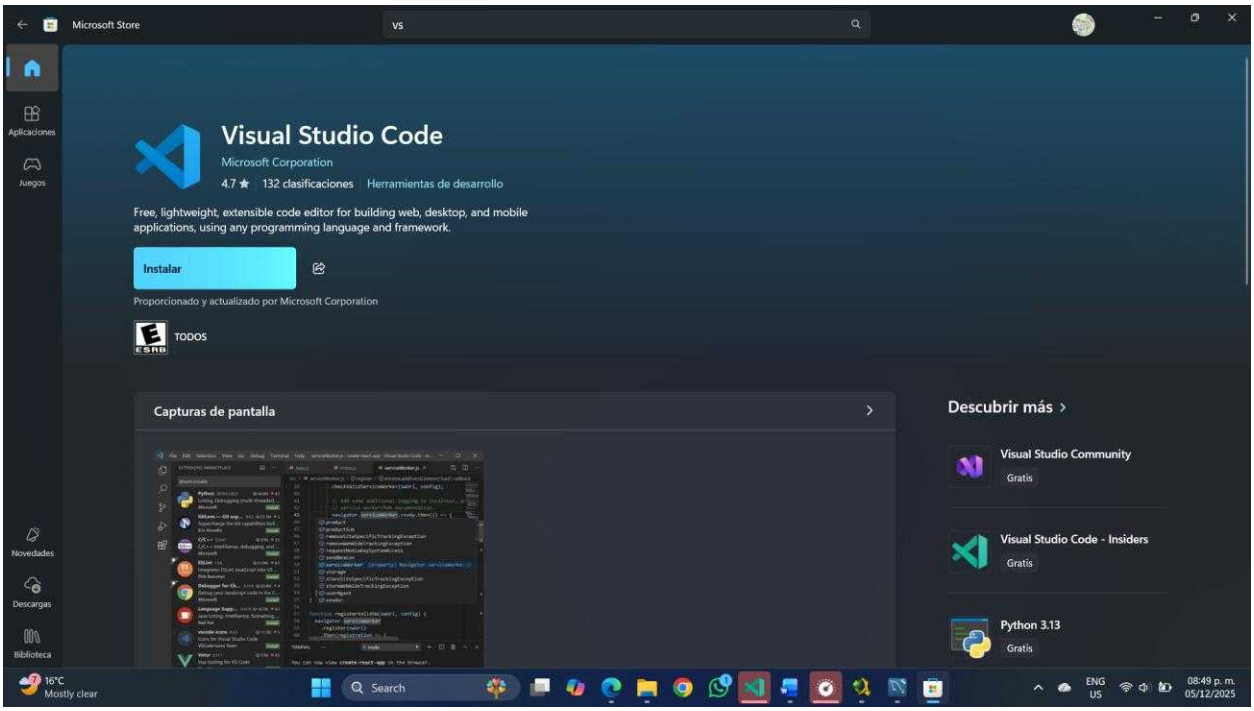
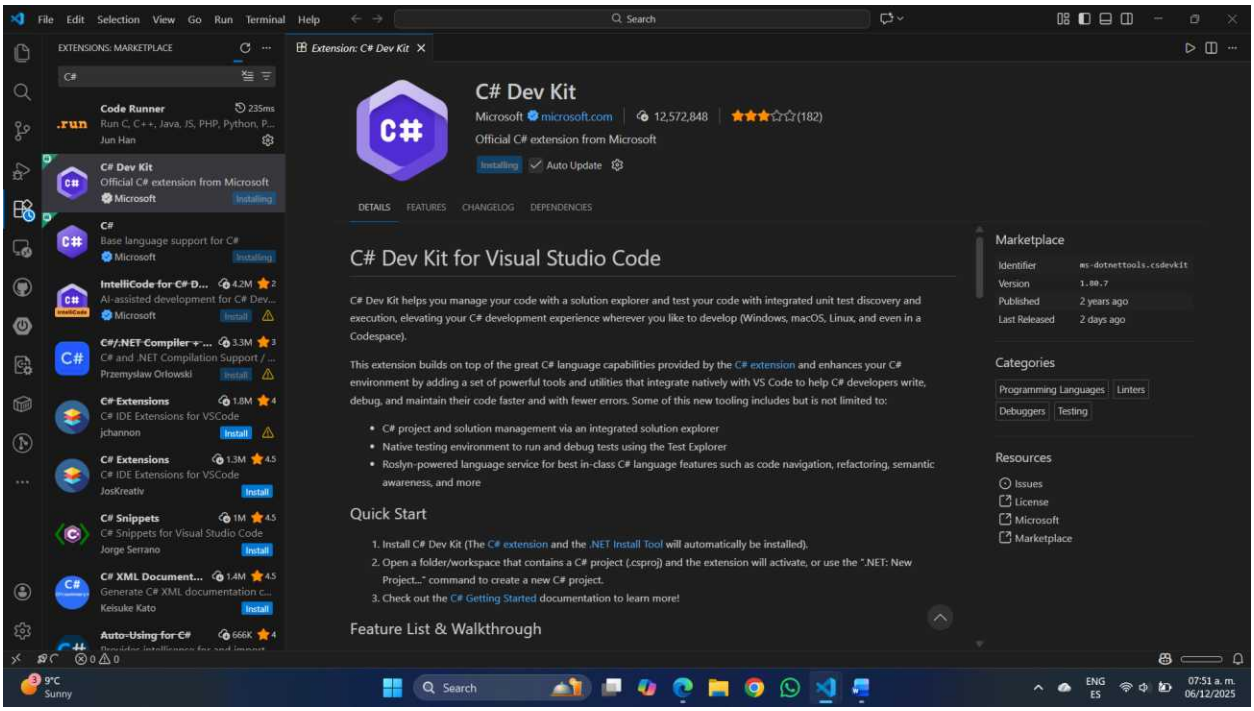


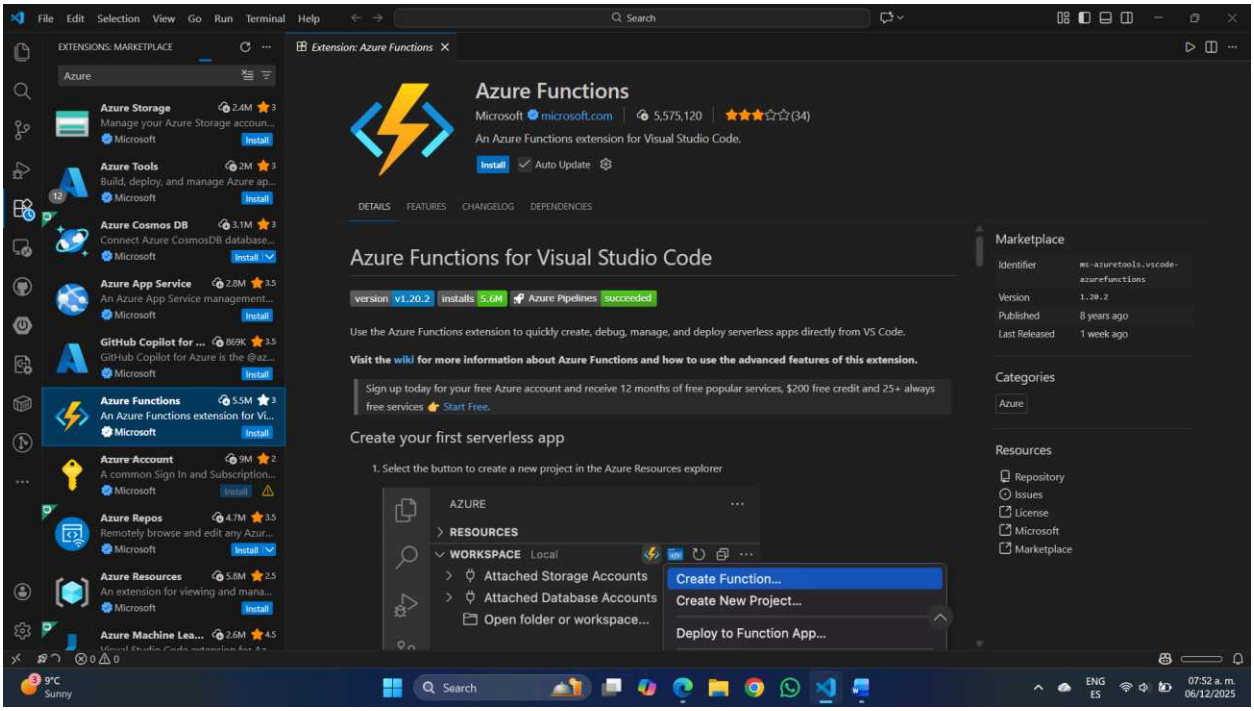
Figura 4.1 instalación de Visual Stdio Code desde Microsoft store.

- Se instaló la extensión “C#” para soporte de .NET.
- Se instaló la extensión “Azure Functions” para gestionar el proyecto serverless.
- Se instaló “Azure Account” para autenticación en Azure.

Extensión VS Code	Propósito
C# (ms-dotnettools)	Soporte lenguaje C# y .NET
Azure Functions	Creación y publicación Functions
Azure Account	Autenticación a Azure

Tabla 5 Extensiones de VS Code


File Edit Selection View Go Run Terminal Help
Search
EXTENSIONS: MARKETPLACE
C#
Code Runner
Run C, C++, Java, JS, PHP, Python, P...
Jun Han
C# Dev Kit
Official C# extension from Microsoft
Microsoft
C#
Base language support for C#
Microsoft
IntelliCode for C# D...
AI-assisted development for C# Dev...
Microsoft
C#/.NET Compiler ...
C# and .NET Compilation Support / ...
Przemysław Orłowski
C# Extensions
C# IDE Extensions for VSCode
jchannon
C# Extensions
C# IDE Extensions for VSCode
JosKreativ
C# Snippets
C# Snippets for Visual Studio Code
Jorge Serrano
C# XML Document...
Generate C# XML documentation c...
Keisuke Kato
Auto-Using for C#
Extension: C# Dev Kit
C# Dev Kit
Microsoft microsoft.com 12,572,848 (182)
Official C# extension from Microsoft
Installing Auto Update
DETAILS FEATURES CHANGELOG DEPENDENCIES
C# Dev Kit for Visual Studio Code
C# Dev Kit helps you manage your code with a solution explorer and test your code with integrated unit test discovery and execution, elevating your C# development experience wherever you like to develop (Windows, macOS, Linux, and even in a Codespace).
This extension builds on top of the great C# language capabilities provided by the C# extension and enhances your C# environment by adding a set of powerful tools and utilities that integrate natively with VS Code to help C# developers write, debug, and maintain their code faster and with fewer errors. Some of this new tooling includes but is not limited to:
C# project and solution management via an integrated solution explorer
Native testing environment to run and debug tests using the Test Explorer
Roslyn-powered language service for best in-class C# language features such as code navigation, refactoring, semantic awareness, and more
Quick Start
1. Install C# Dev Kit (The C# extension and the .NET Install Tool will automatically be installed).
2. Open a folder/workspace that contains a C# project (.csproj) and the extension will activate, or use the ".NET: New Project..." command to create a new C# project.
3. Check out the C# Getting Started documentation to learn more!
Feature List & Walkthrough
Marketplace
Identifier ms-dotnettools.csdevkit
Version 1.80.7
Published 2 years ago
Last Released 2 days ago
Categories
Programming Languages Linters
Debuggers Testing
Resources
Issues
License
Microsoft
Marketplace
9°C Sunny
Search
ENG ES
07:51 a. m. 06/12/2025


File Edit Selection View Go Run Terminal Help
Search
EXTENSIONS: MARKETPLACE
Azure
Azure Storage
Manage your Azure Storage accoun...
Microsoft
Azure Tools
Build, deploy, and manage Azure ap...
Microsoft
Azure Cosmos DB
Connect Azure CosmosDB database...
Microsoft
Azure App Service
An Azure App Service management...
Microsoft
GitHub Copilot for ...
GitHub Copilot for Azure is the @az...
Microsoft
Azure Functions
An Azure Functions extension for Vi...
Microsoft
Azure Account
A common Sign In and Subscription...
Microsoft
Azure Repos
Remotely browse and edit any Azur...
Microsoft
Azure Resources
An extension for viewing and mana...
Microsoft
Azure Machine Lea...
Extension: Azure Functions
Azure Functions
Microsoft microsoft.com 5,575,120 (34)
An Azure Functions extension for Visual Studio Code.
Install Auto Update
DETAILS FEATURES CHANGELOG DEPENDENCIES
Azure Functions for Visual Studio Code
version v1.20.2 installs 5.6M Azure Pipelines succeeded
Use the Azure Functions extension to quickly create, debug, manage, and deploy serverless apps directly from VS Code.
Visit the wiki for more information about Azure Functions and how to use the advanced features of this extension.
Sign up today for your free Azure account and receive 12 months of free popular services, $200 free credit and 25+ always free services Start Free.
Create your first serverless app
1. Select the button to create a new project in the Azure Resources explorer
AZURE
RESOURCES
WORKSPACE Local
Attached Storage Accounts
Attached Database Accounts
Open folder or workspace...
Create Function...
Create New Project...
Deploy to Function App...
Marketplace
Identifier ms-azuretools.vscode-azurefunctions
Version 1.20.2
Published 8 years ago
Last Released 1 week ago
Categories
Azure
Resources
Repository
Issues
License
Microsoft
Marketplace
9°C Sunny
Search
ENG ES
07:52 a. m. 06/12/2025

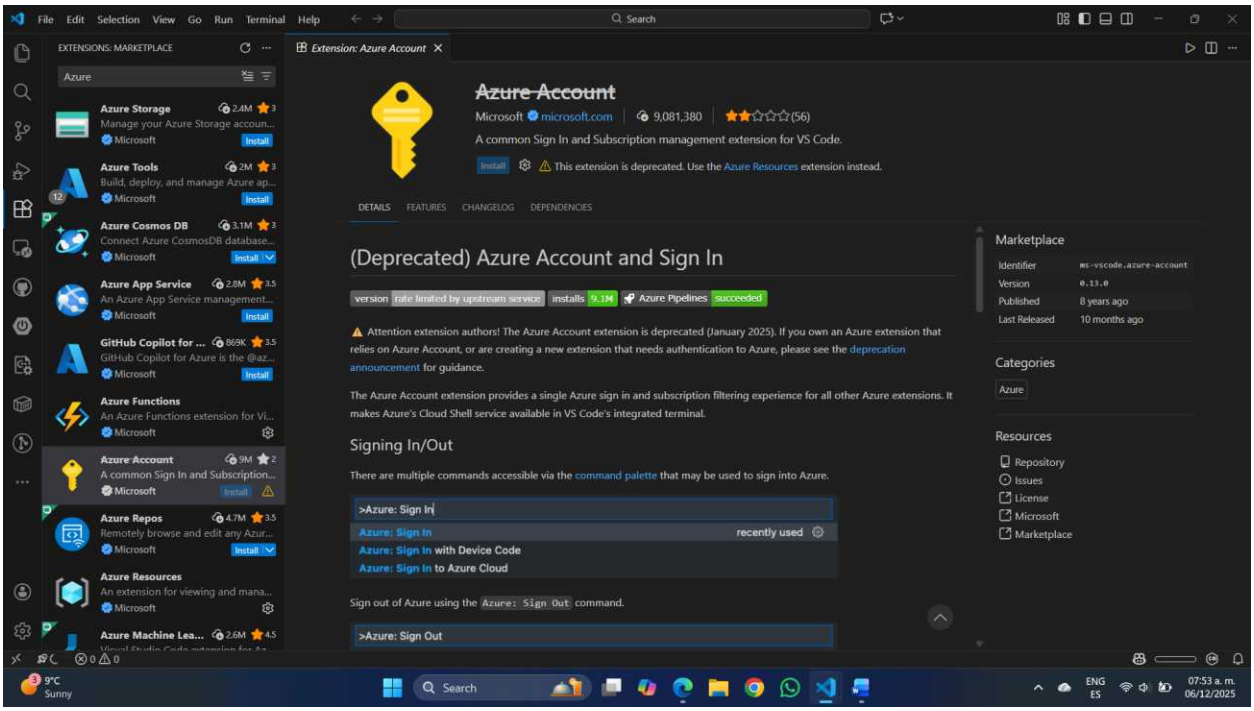


Figura 5. Visual Studio Code con extensiones C#, Azure Functions y Azure Account instaladas.

4.7 Instalación de .NET SDK

Se instaló .NET SDK 8.0.x (o 7.0.x si así se requiere). Se verificó la instalación ejecutando dotnet --info en una terminal. Se accedió a la terminal integrada de VS Code para futuras operaciones de paquetes.

- Se descargó el instalador .NET SDK y se ejecutó como administrador.
- Se verificó la versión instalada con dotnet --version y dotnet --info.

Comando	Resultado esperado
dotnet --version	8.0.x
dotnet --info	Información del runtime y SDK

Tabla 6 Tabla de comandos

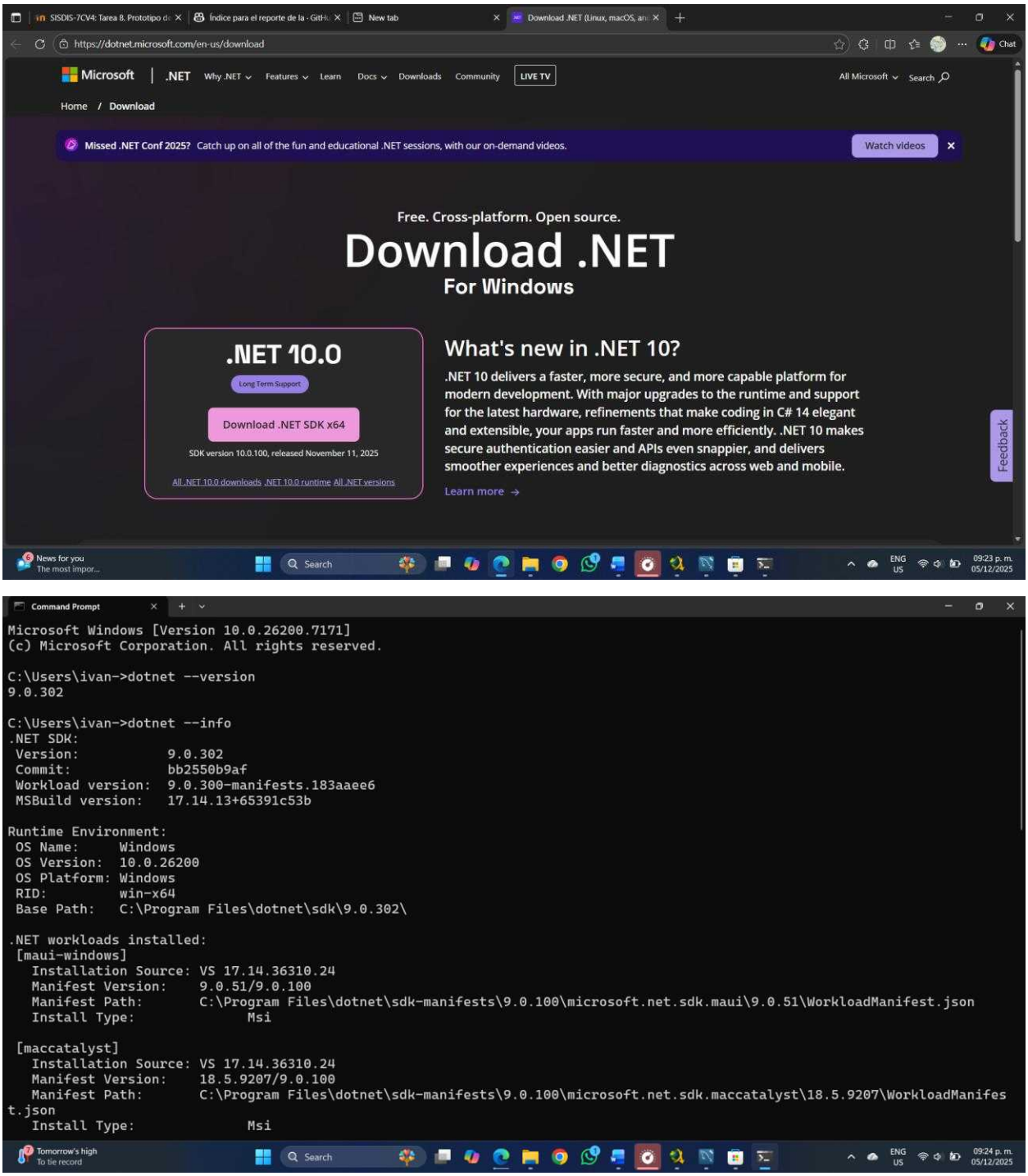


Figura 6. Verificación de .NET SDK instalado mediante la terminal.

4.8 Instalación de Azure Functions Core Tools

Se instaló Azure Functions Core Tools v4 para ejecutar Functions localmente. Se accedió a la guía oficial y se instaló usando npm o MSI (Windows). Se verificó con func --version.

- Opción MSI (Windows): descargar el instalador v4 desde la guía oficial.
- Opción npm: npm i -g azure-functions-core-tools@4 --unsafe-perm true (requiere Node.js).
- Se validó con func npm --version.

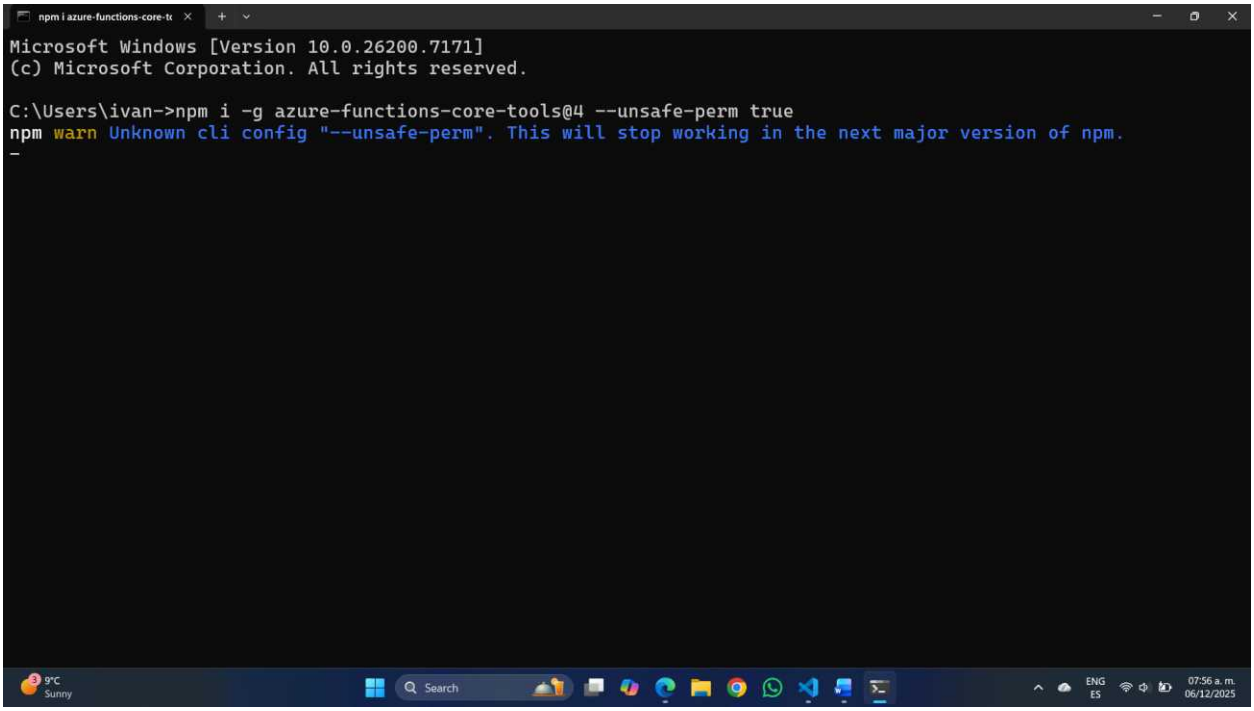
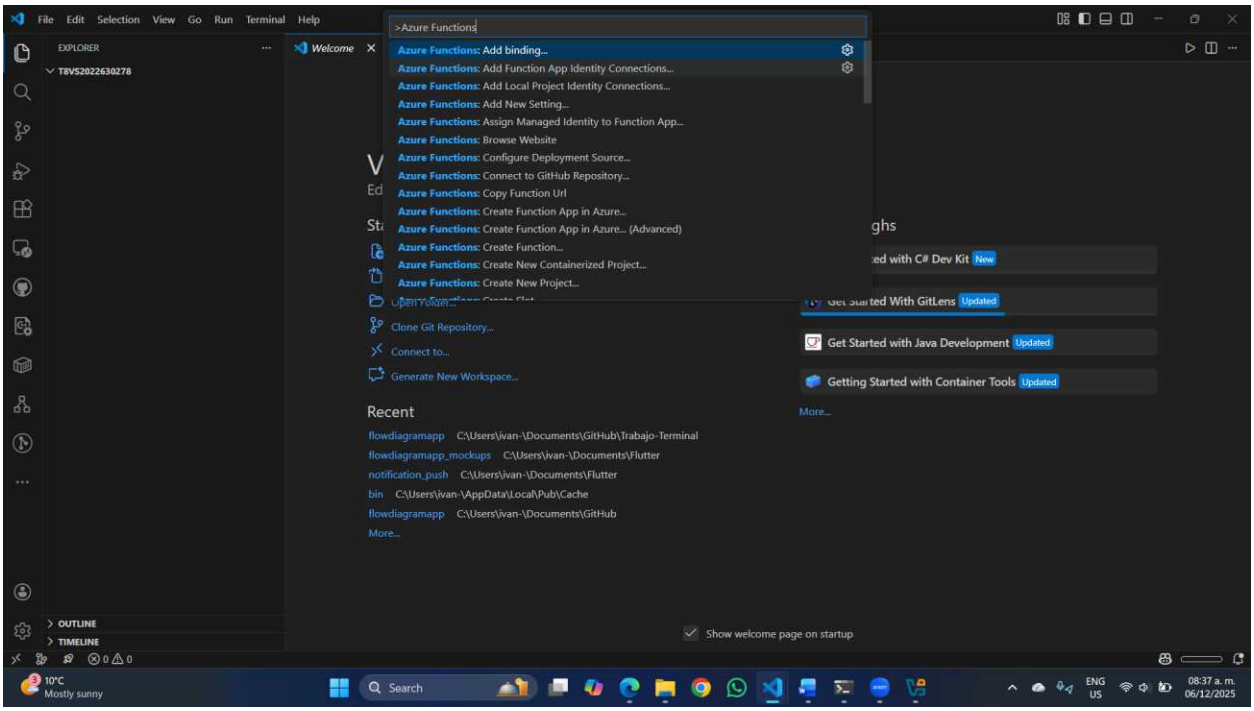


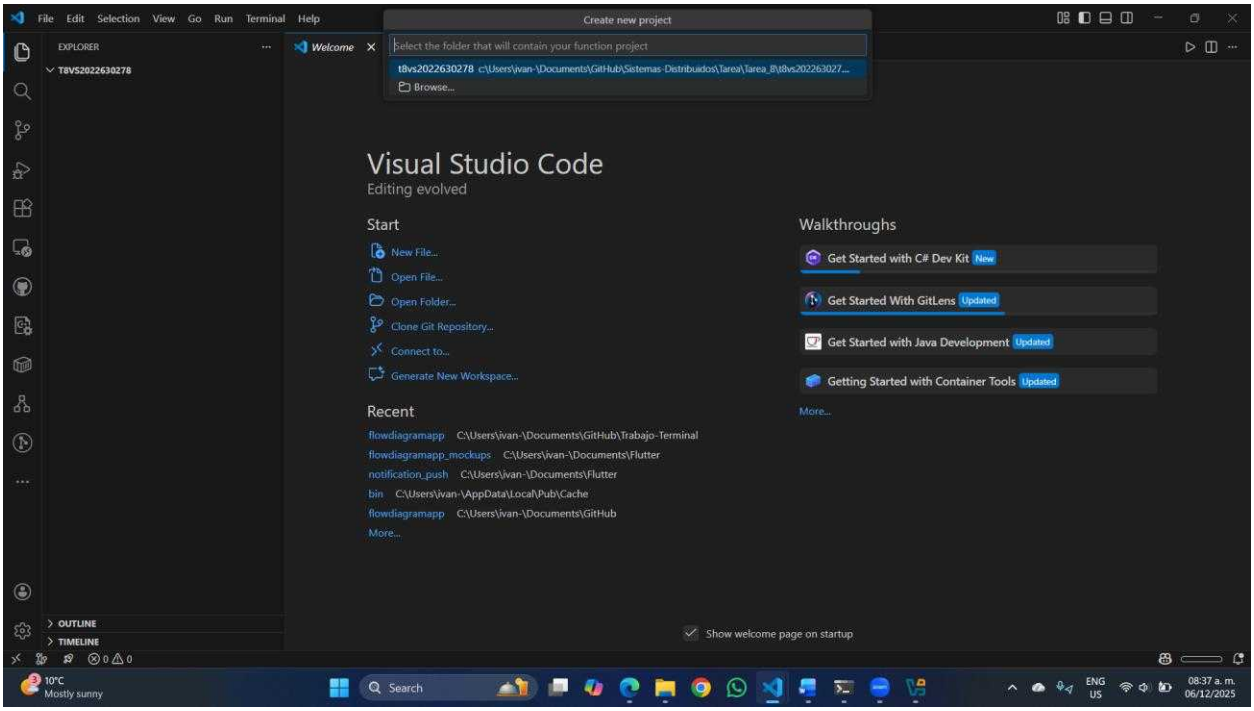
Figura 7. Instalación y verificación de Azure Functions Core Tools v4.

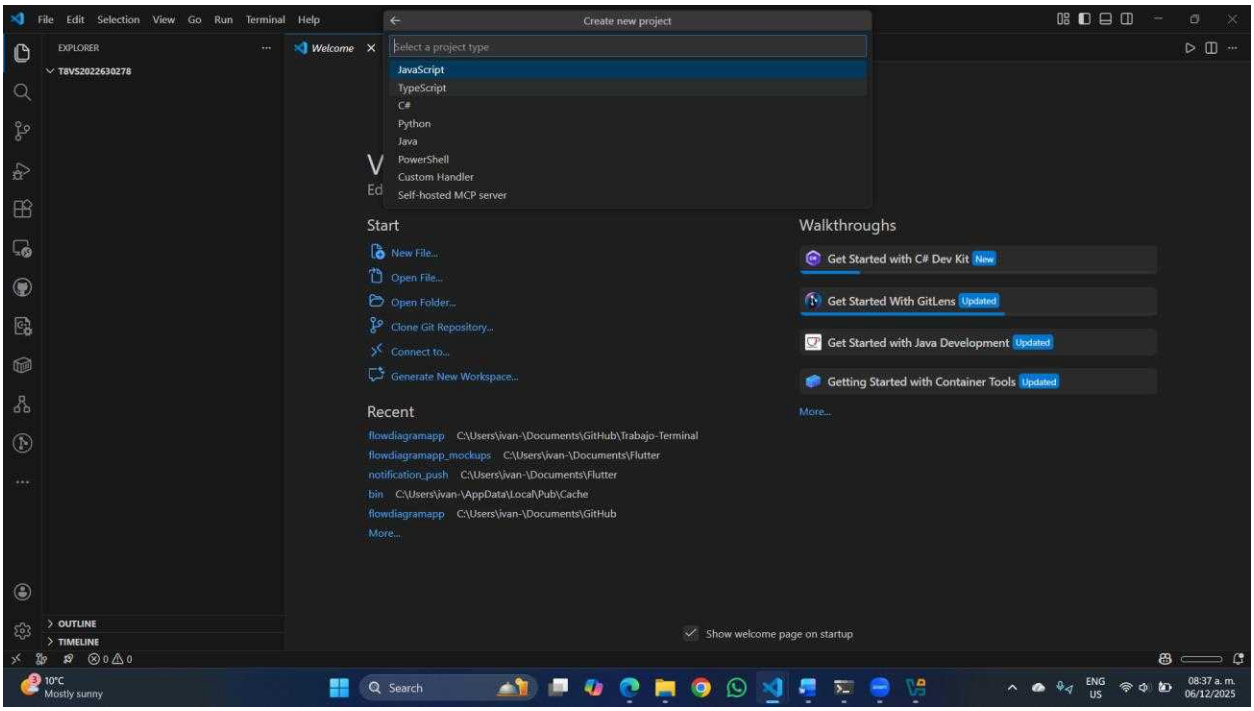
4.9 Creación del proyecto Azure Functions en Visual Studio Code

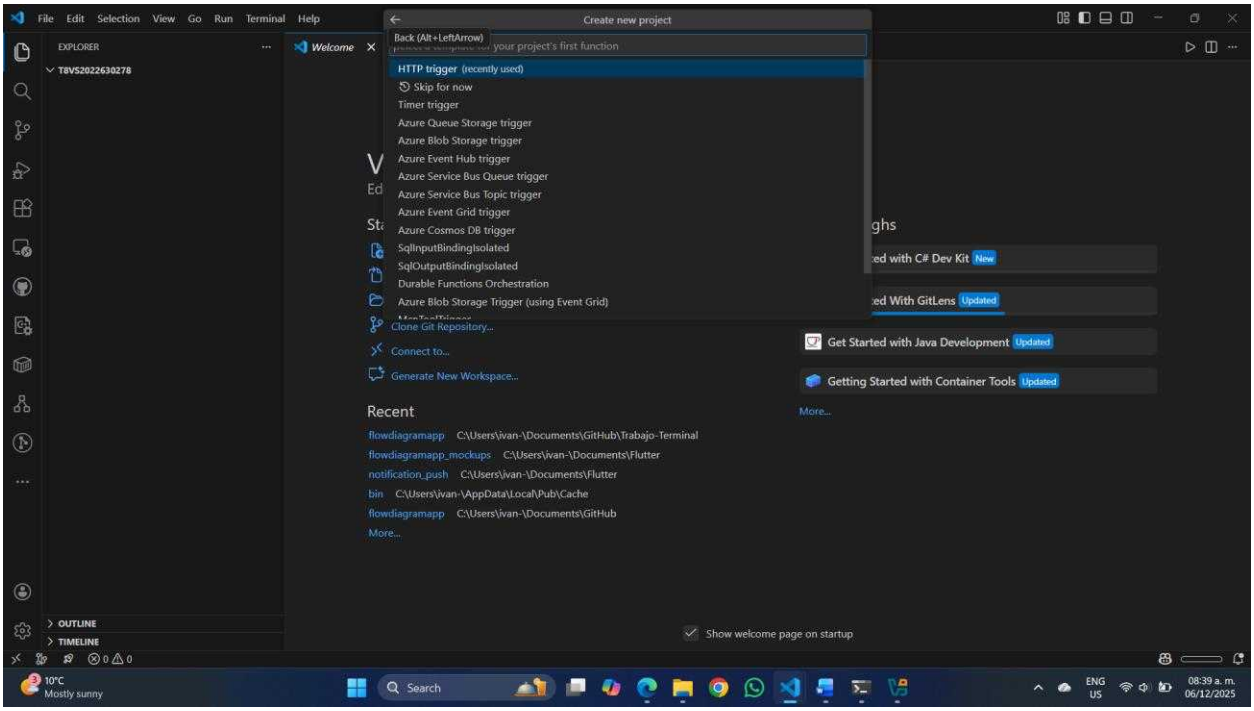
Se creó el proyecto de Azure Functions (C#, HTTP Trigger) en la carpeta t8vs2022630278. Se copió el contenido del back-end.zip al directorio del proyecto y se integraron las funciones base suministradas por el docente.

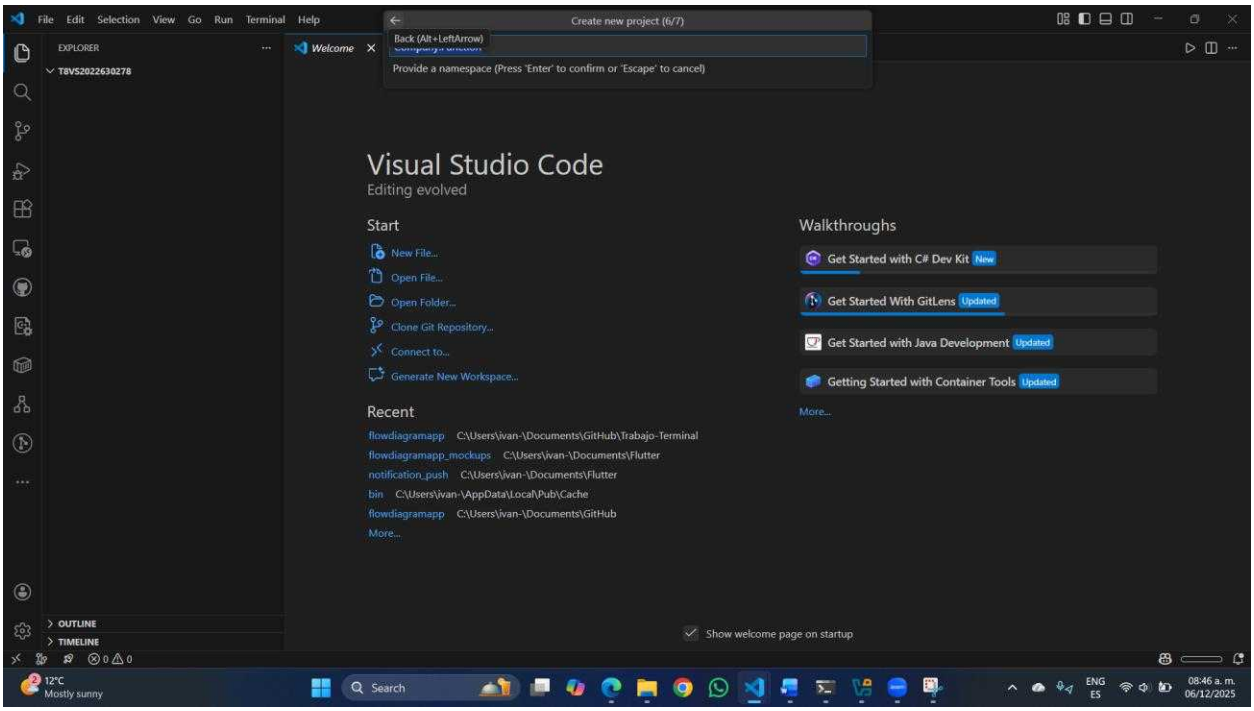
- Se accedió a VS Code y se creó el proyecto Functions (HTTP Trigger anónimo).
- Se copiaron los archivos: Get.cs, MimeMapping.cs, alta_usuario.cs, borra_usuario.cs, consulta_usuario.cs, login.cs, modifica_usuario.cs.
- Se compiló el proyecto para verificar que no existan errores.

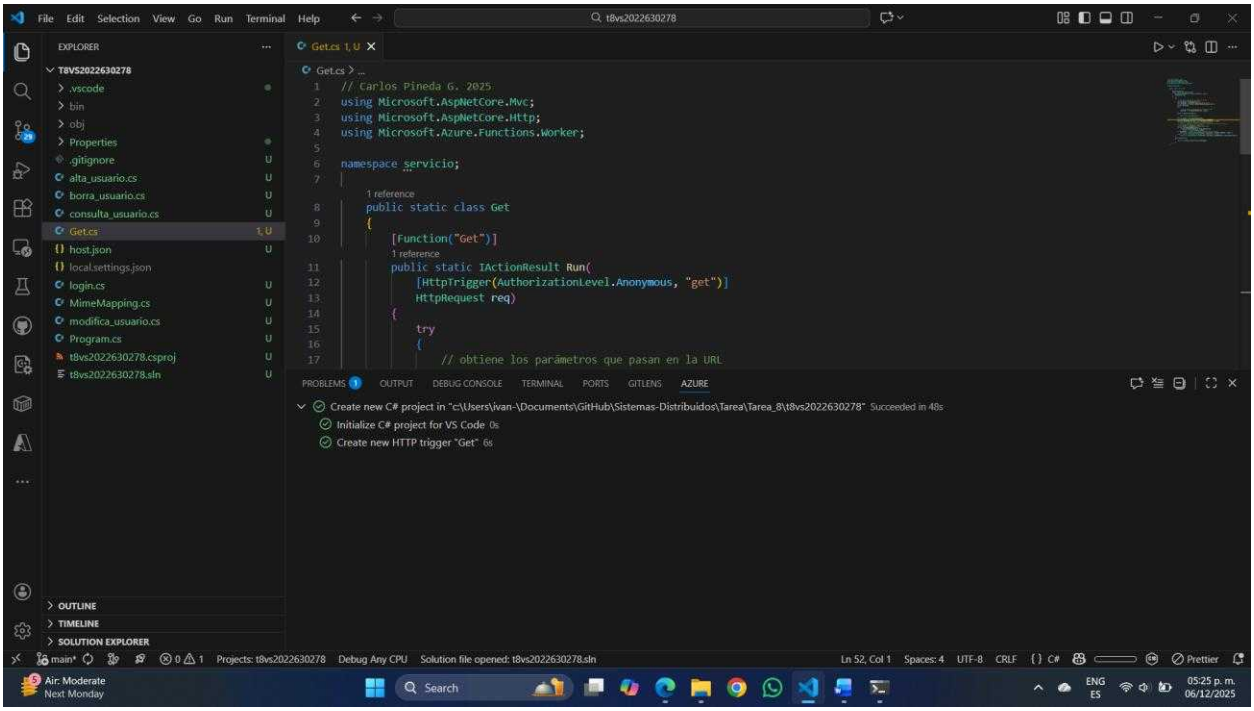

File Edit Selection View Go Run Terminal Help
>Azure Functions
Azure Functions: Add binding...
Azure Functions: Add Function App Identity Connections...
Azure Functions: Add Local Project Identity Connections...
Azure Functions: Add New Setting...
Azure Functions: Assign Managed Identity to Function App...
Azure Functions: Browse Website
Azure Functions: Configure Deployment Source...
Azure Functions: Connect to GitHub Repository...
Azure Functions: Copy Function Url
Azure Functions: Create Function App in Azure...
Azure Functions: Create Function App in Azure... (Advanced)
Azure Functions: Create Function...
Azure Functions: Create New Containerized Project...
Azure Functions: Create New Project...
EXPLORER
T8VS2022630278
Welcome
Clone Git Repository...
Connect to...
Generate New Workspace...
Recent
flowdiagramapp C:\Users\ivan-\Documents\GitHub\Trabajo-Terminal
flowdiagramapp_mockups C:\Users\ivan-\Documents\Flutter
notification_push C:\Users\ivan-\Documents\Flutter
bin C:\Users\ivan-\AppData\Local\Pub\Cache
flowdiagramapp C:\Users\ivan-\Documents\GitHub
More...
Get Started with C# Dev Kit New
Get Started With GitLens Updated
Get Started with Java Development Updated
Getting Started with Container Tools Updated
More...
Show welcome page on startup
OUTLINE
TIMELINE
10°C Mostly sunny
Search
ENG US
08:37 a. m. 06/12/2025


File Edit Selection View Go Run Terminal Help
Create new project
Select the folder that will contain your function project
t8vs2022630278 c:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_8\t8vs202263027...
Browse...
EXPLORER
T8VS2022630278
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Generate New Workspace...
Recent
flowdiagramapp C:\Users\ivan-\Documents\GitHub\Trabajo-Terminal
flowdiagramapp_mockups C:\Users\ivan-\Documents\Flutter
notification_push C:\Users\ivan-\Documents\Flutter
bin C:\Users\ivan-\AppData\Local\Pub\Cache
flowdiagramapp C:\Users\ivan-\Documents\GitHub
More...
Walkthroughs
Get Started with C# Dev Kit New
Get Started With GitLens Updated
Get Started with Java Development Updated
Getting Started with Container Tools Updated
More...
Show welcome page on startup
OUTLINE
TIMELINE
10°C Mostly sunny
Search
ENG US
08:37 a. m. 06/12/2025


File Edit Selection View Go Run Terminal Help
Create new project
Select a project type
JavaScript
TypeScript
C#
Python
Java
PowerShell
Custom Handler
Self-hosted MCP server
Welcome
EXPLORER
T8VS2022630278
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Generate New Workspace...
Walkthroughs
Get Started with C# Dev Kit New
Get Started With GitLens Updated
Get Started with Java Development Updated
Getting Started with Container Tools Updated
More...
Recent
flowdiagramapp C:\Users\ivan-\Documents\GitHub\Trabajo-Terminal
flowdiagramapp_mockups C:\Users\ivan-\Documents\Flutter
notification_push C:\Users\ivan-\Documents\Flutter
bin C:\Users\ivan-\AppData\Local\Pub\Cache
flowdiagramapp C:\Users\ivan-\Documents\GitHub
More...
Show welcome page on startup
OUTLINE
TIMELINE
10°C
Mostly sunny
Search
ENG
US
08:37 a. m.
06/12/2025


File Edit Selection View Go Run Terminal Help
Create new project
Back (Alt+LeftArrow)
HTTP trigger (recently used)
Skip for now
Timer trigger
Azure Queue Storage trigger
Azure Blob Storage trigger
Azure Event Hub trigger
Azure Service Bus Queue trigger
Azure Service Bus Topic trigger
Azure Event Grid trigger
Azure Cosmos DB trigger
SqlInputBindingIsolated
SqlOutputBindingIsolated
Durable Functions Orchestration
Azure Blob Storage Trigger (using Event Grid)
Welcome
EXPLORER
T8VS2022630278
Clone Git Repository...
Connect to...
Generate New Workspace...
Get Started with Java Development Updated
Getting Started with Container Tools Updated
More...
Recent
flowdiagramapp C:\Users\ivan-\Documents\GitHub\Trabajo-Terminal
flowdiagramapp_mockups C:\Users\ivan-\Documents\Flutter
notification_push C:\Users\ivan-\Documents\Flutter
bin C:\Users\ivan-\AppData\Local\Pub\Cache
flowdiagramapp C:\Users\ivan-\Documents\GitHub
More...
Show welcome page on startup
OUTLINE
TIMELINE
10°C
Mostly sunny
Search
ENG
US
08:39 a. m.
06/12/2025


Create new project (6/7)
Back (Alt+LeftArrow)
Provide a namespace (Press 'Enter' to confirm or 'Escape' to cancel)
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Generate New Workspace...
Walkthroughs
Get Started with C# Dev Kit
Get Started With GitLens
Get Started with Java Development
Getting Started with Container Tools
More...
Recent
flowdiagramapp C:\Users\ivan-\Documents\GitHub\Trabajo-Terminal
flowdiagramapp_mockups C:\Users\ivan-\Documents\Flutter
notification_push C:\Users\ivan-\Documents\Flutter
bin C:\Users\ivan-\AppData\Local\Pub\Cache
flowdiagramapp C:\Users\ivan-\Documents\GitHub
More...
Show welcome page on startup


// Carlos Pineda G. 2025
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using Microsoft.Azure.Functions.Worker;
namespace servicio;
public static class Get
[Function("Get")]
public static IActionResult Run(
[HttpTrigger(AuthorizationLevel.Anonymous, "get")]
HttpRequest req)
try
// obtiene los parámetros que pasan en la URL
Create new C# project in "c:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_8\t8vs2022630278" Succeeded in 48s
Initialize C# project for VS Code 0s
Create new HTTP trigger "Get" 6s

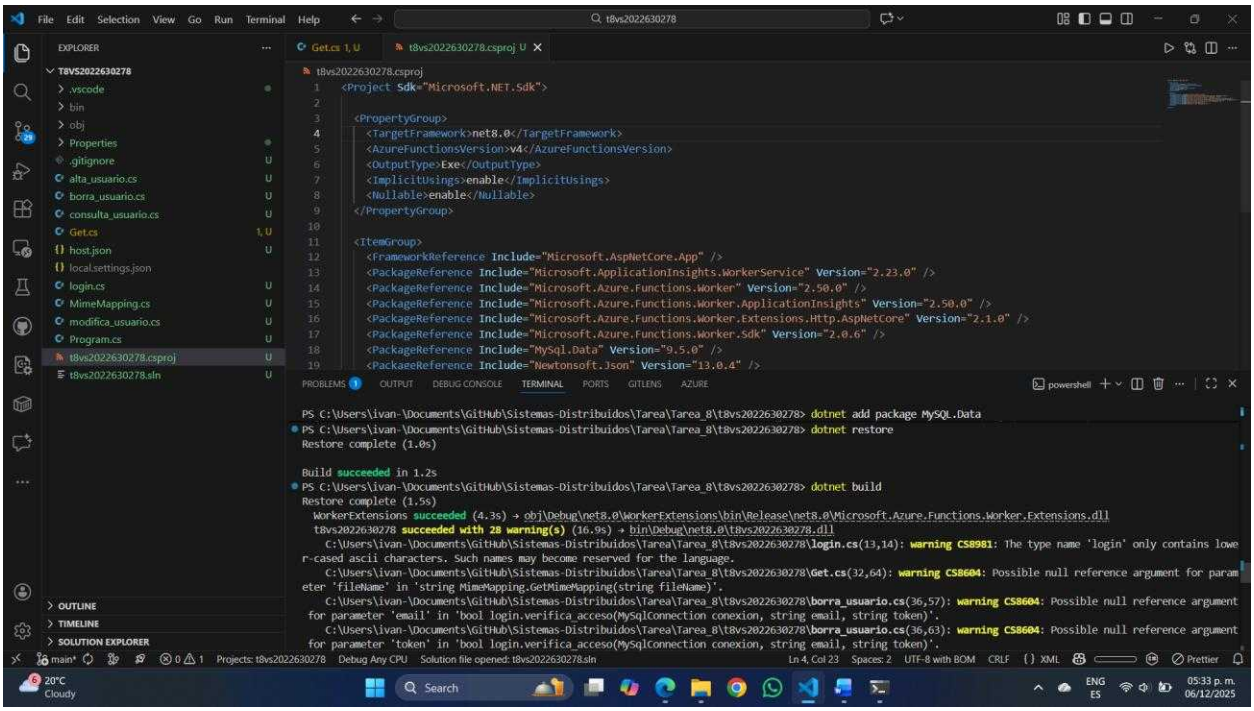


Figura 8. Estructura del proyecto t8vs2022630278 con funciones del back-end integradas.

4.10 Instalación de paquetes .NET requeridos

Se instalaron los paquetes Newtonsoft.Json y MySQL.Data desde la terminal de VS Code para habilitar la serialización JSON y la conectividad a MySQL. Se accedió al proyecto y se ejecutaron los comandos:

dotnet add package Newtonsoft.Json

dotnet add package MySQL.Data

dotnet restore

dotnet build

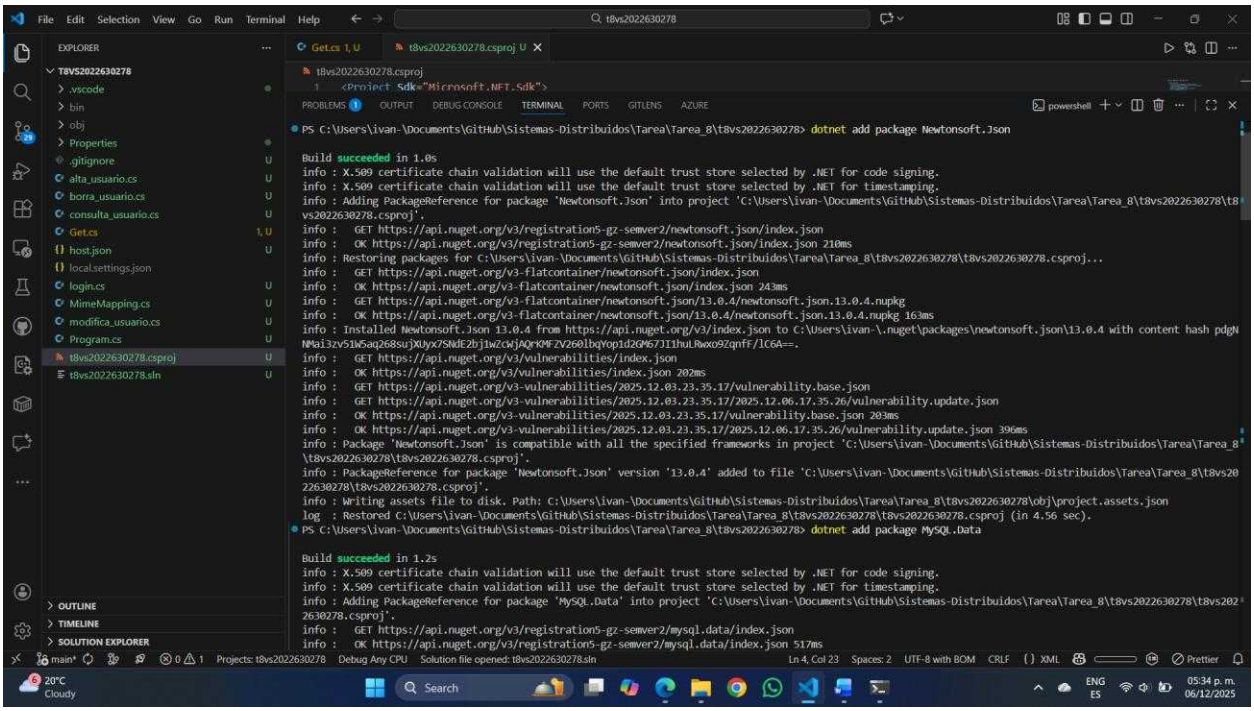


Figura 9. Instalación y restauración de paquetes .NET en el proyecto Functions.

4.11 Configuración de variables locales y front-end

Se creó y configuró local.settings.json con las variables de entorno para pruebas locales: Server, UserID, Password, Database y ROOT. Se colocó el front-end en una carpeta local y se verificó que la función Get sirva los archivos correctamente.

Ejemplo de local.settings.json (valores de ejemplo locales):

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated",
    "Server": "localhost",
    "UserID": "hugo",
```

```
    "Password": "AQUI-VA-LA-CONTRASEÑA-DEL-USUARIO-HUGO",
    "Database": "servicio_web",
    "ROOT": "C:\\t8vs2022630278\\front-end"
  }
}
```


Variable	Valor (ejemplo local)	Descripción
Server	localhost	Host local de MySQL
UserID	hugo	Usuario creado para pruebas
Password	AQUI-VA-LA-CONTRASEÑA-DEL-USUARIO-HUGO	Contraseña del usuario local
Database	servicio_web	Nombre de la base de datos
ROOT	C:\t8vs2022630278\front-end	Ruta del front-end para servir con la función Get

Tabla 7 Tabla de variables

Se colocó en C:\t8vs2022630278\front-end:

- prueba.html
- WSClient.js
- usuario_sin_foto.png

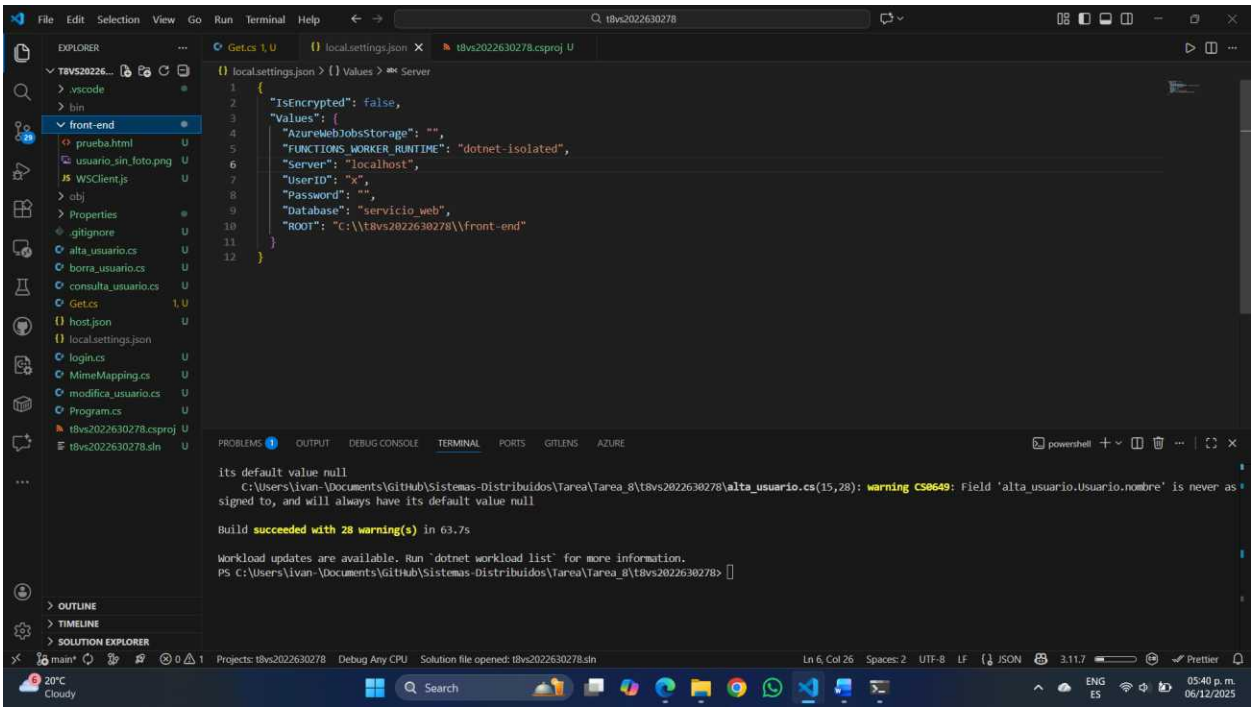
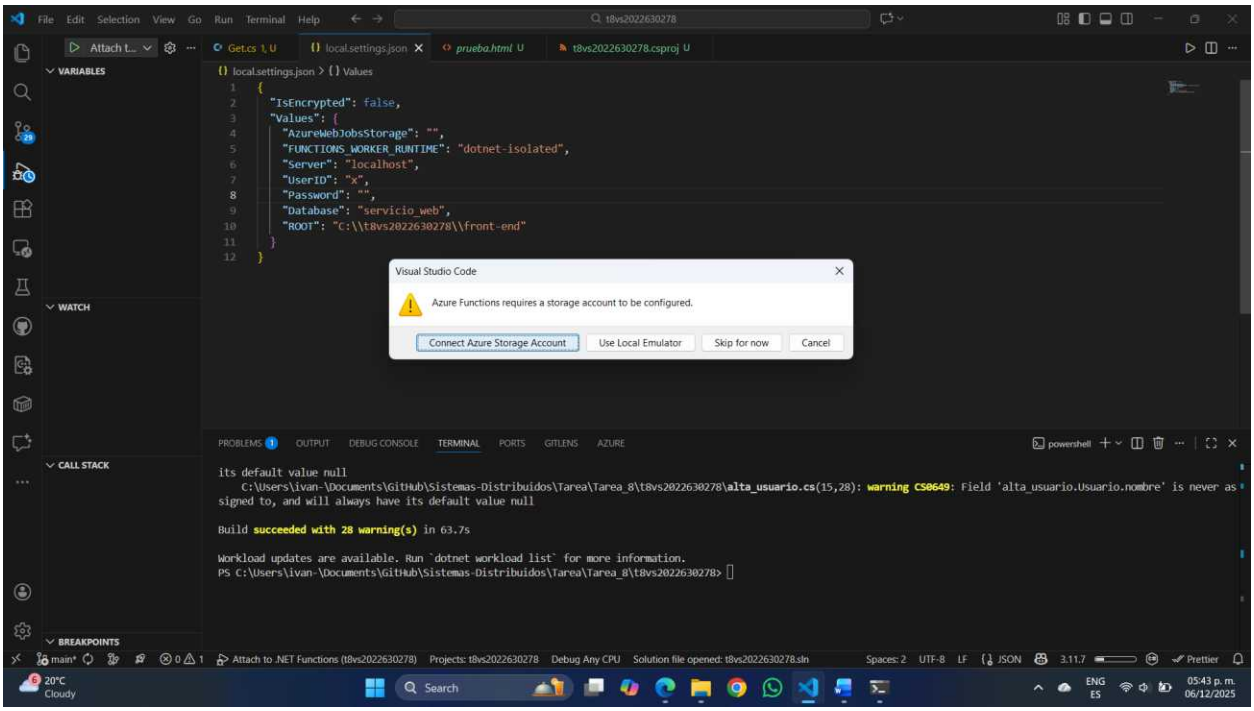


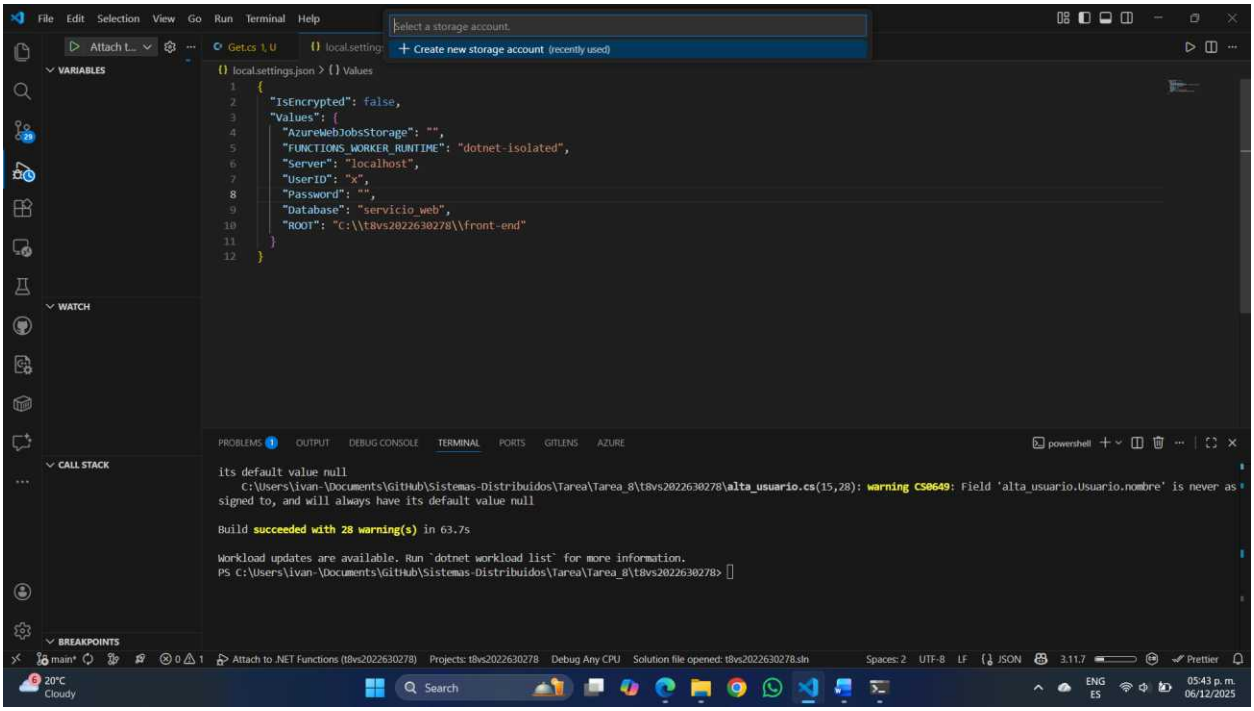
Figura 10. Configuración de local.settings.json y preparación de la carpeta del front-end.

4.12 Pruebas locales iniciales del front-end y back-end

Se accedió al proyecto Functions y se inició localmente con func start o F5 en VS Code. Se abrió el navegador en la URL:

http://localhost:7071/api/Get?nombre=/prueba.html


Visual Studio Code
Azure Functions requires a storage account to be configured.
Connect Azure Storage Account
Use Local Emulator
Skip for now
Cancel


Select a storage account.
Create new storage account (recently used)

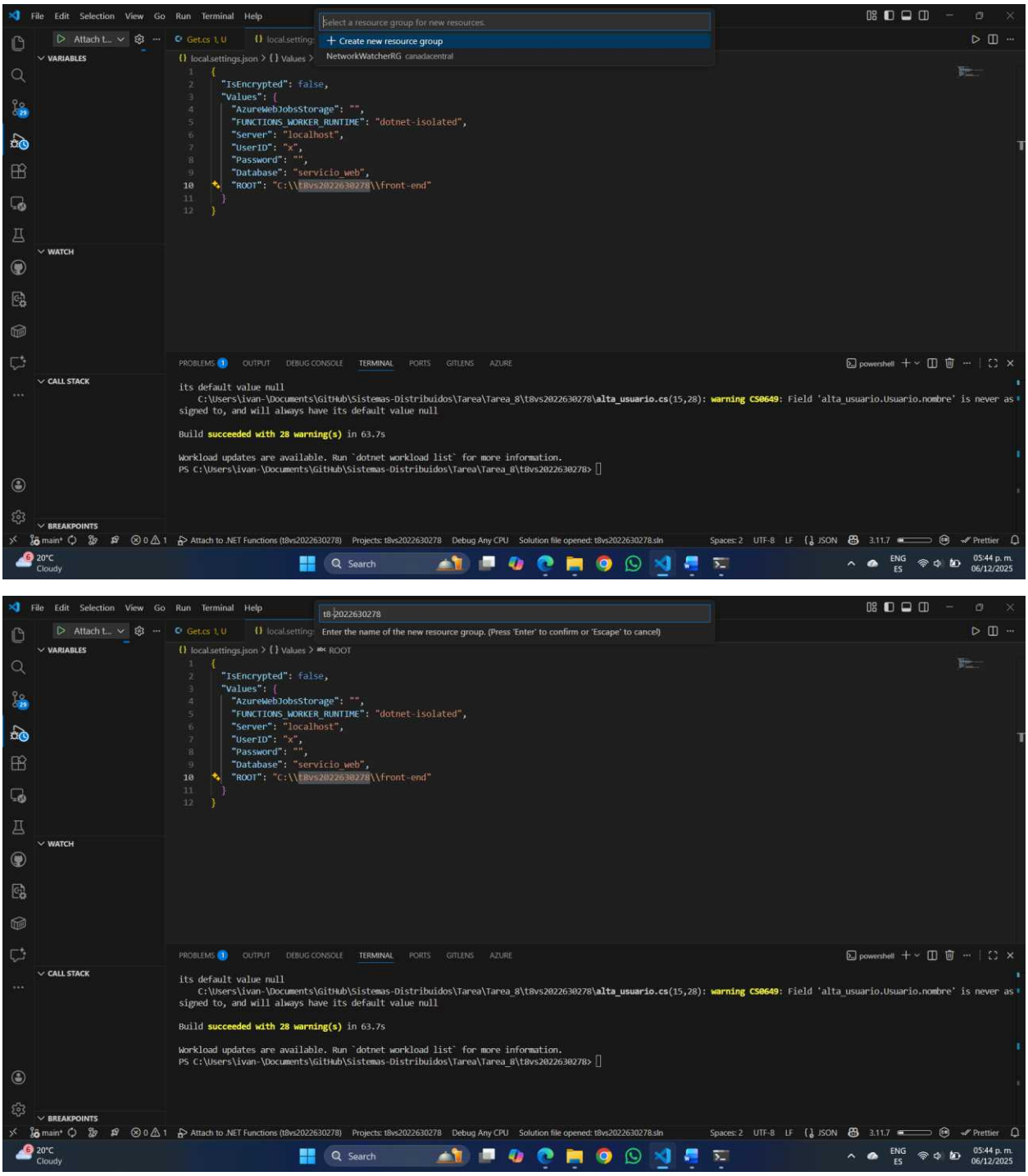


Figura 10. Compilación del proyecto.

Se realizó el flujo de registro y login de usuario y se validaron las respuestas JSON. Se prepararon los comandos curl para cada función a fin de incluirlos más adelante en el reporte de evidencias.

Prueba	URL / Operación	Resultado esperado
Cargar front-end	/api/Get?nombre=/prueba.html	Render de la app en navegador
Registro usuario	POST /api/alta_usuario	200 y mensaje de confirmación
Login	GET /api/login	200 y token
Consulta perfil	GET /api/consulta_usuario	JSON con datos del usuario
Modificar perfil	PUT /api/modifica_usuario	200 y mensaje de modificación
Borrar usuario	DELETE /api/borra_usuario	200 y mensaje de borrado

Tabla 8 Tabla de pruebas

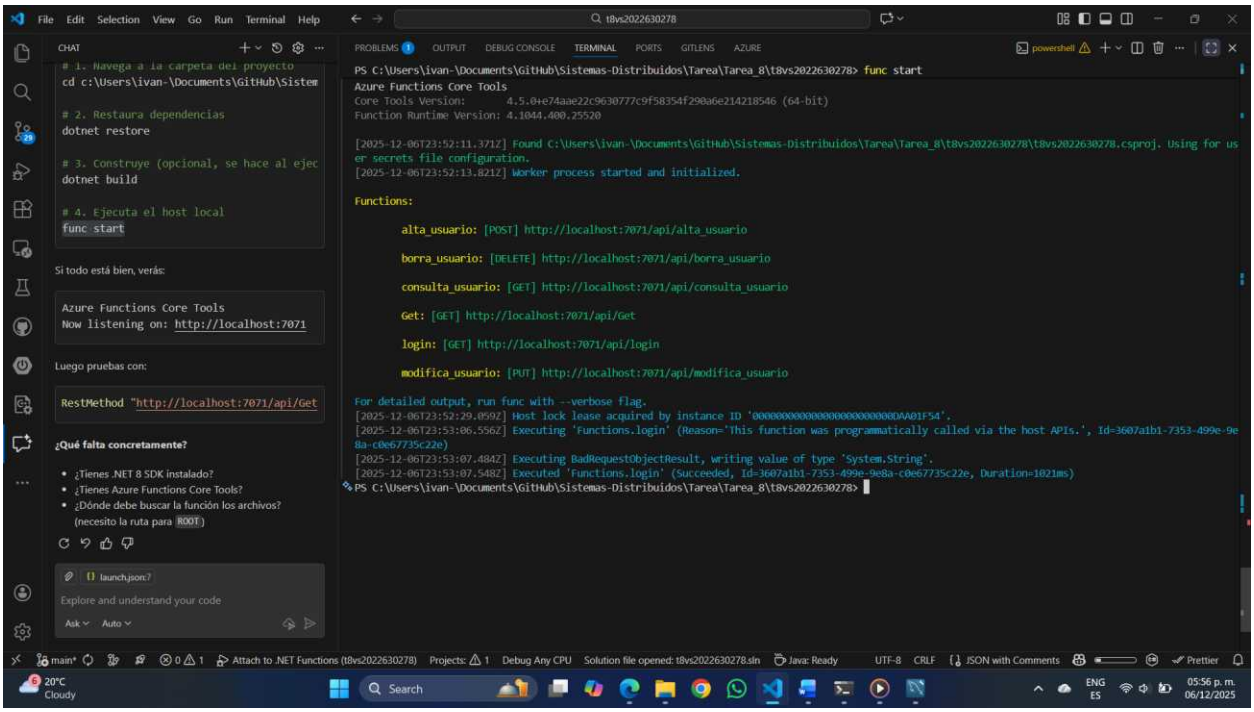


Figura 11. Navegador mostrando la app y ejecución del flujo de usuario contra el back-end local.

5 Desarrollo

6 Back-end: Funciones de Azure implementadas

Se realizó la implementación del back-end utilizando Azure Functions en C#, conectando con MySQL PaaS mediante el paquete MySQL.Data y manejando la serialización con Newtonsoft.Json. Se accedió a la base de datos “servicio_web” con variables de entorno y se respetó el patrón de respuestas JSON y códigos HTTP. En operaciones críticas del carrito y del stock se utilizaron transacciones para garantizar consistencia. A continuación se describen las funciones implementadas y su comportamiento.

6.1 Resumen de funciones y operaciones

La siguiente tabla resume cada función: método HTTP, endpoint, parámetros clave, operaciones en la base de datos, uso de transacción y códigos de respuesta.

Función	Método	Endpoint	Parámetros clave	Operaciones DB	Respuestas
alta_usuario	POST	/api/alta_usuario	email, password, nombre, apellidos, fecha, teléfono, genero, foto	INSERT en usuarios; INSERT opcional en fotos_usuarios	200 OK mensaje éxito; 400 Error con {"mensaje": "..."}
consulta_usuario	GET	/api/consulta_usuario	email, token	SELECT usuarios LEFT JOIN fotos_usuarios	200 OK con JSON usuario; 400 error
modifica_usuario	PUT	/api/modifica_usuario	email, token; body con datos y foto	UPDATE usuarios; opcional UPDATE password; DELETE/INSERT	200 OK mensaje; 400 error

				fotos_usuarios	
borra_usuario	DELETE	/api/borra_usuario	email, token	DELETE fotos_usuarios; DELETE usuarios	200 OK mensaje; 400 error
login	GET	/api/login	email, password (sha256)	SELECT credenciales; UPDATE token	200 OK {"token": "..."}; 400 "Acceso denegado"
alta_articulo	POST	/api/alta_articulo	nombre, descripcion, precio, cantidad, foto, id_usuario, token	INSERT stock; INSERT opcional fotos_articulos	200 OK mensaje; 400 error acceso/validación
consulta_articulos	GET	/api/consulta_articulos	palabra_clave, id_usuario, token	SELECT con LIKE en nombre/descripcion; LEFT JOIN foto	200 OK arreglo de artículos; 400 error acceso
compra_articulo	POST	/api/compra_articulo	id_articulo, cantidad, id_usuario, token	SELECT stock; INSERT/UPDATE carrito_compra; UPDATE stock	200 OK; 400 "No hay suficientes artículos en stock"
elimina_articulo_carrito_compra	DELETE	/api/elimina_articulo_carrito_compra	id_usuario, id_articulo, token	SELECT cantidad del carrito; UPDATE stock (+); DELETE carrito	200 OK; 400 error acceso/consistencia
elimina_carrito_compra	DELETE	/api/elimina_carrito_compra	id_usuario, token	SELECT todos del carrito; UPDATE stock (+) por cada; DELETE	200 OK; 400 error

				carrito del usuario	
modifica_carrito_compra	PUT	/api/modifica_carrito_compra	id_articulo, incremento(+1/-1), id_usuario, token	SELECT stock y carrito; UPDATE carrito y stock	200 OK; 400 "No hay suficientes artículos en stock" / "No hay más artículos en el carrito"

Figura 9. Resumen de endpoints y operaciones del back-end.

6.2 alta_usuario (registro de usuarios)

Se realizó la función alta_usuario para registrar usuarios nuevos. Se validaron campos obligatorios, se insertó el registro en usuarios y, si el front-end envió foto en base64, se insertó en fotos_usuarios. Se utilizó una transacción para agrupar ambas operaciones y asegurar atomicidad.

- Se accedió a variables de entorno: Server, UserID, Password, Database.
- Se instaló MySQL.Data para la conexión y se empleó LastInsertedId para recuperar id_usuario.
- Se manejó OkObjectResult en éxito y BadRequestObjectResult con {"mensaje": "..."} en error.