

# Raport despre folosirea unui tool de AI

## Pentru realizarea implementării

### Articolului “TestLab”

#### Modulul 1:

Modelele AI sunt integrate pretutindeni în zilele de astăzi, iar adevărul este că sunt foarte folositoare. Noi am utilizat mai mult Chat GPT [1], tratând acest model AI precum evoluția unui search engine, pentru a afla rapid informații și pentru a putea căuta în varii surse, într-un mod simplu și rapid, rezolvări pentru erori întâmpinate.

Situațiile în care am folosit Chat GPT în scopul implementării primului modul din cadrul articolului “TestLab” [2]:

1. Instalarea diferitelor tool-uri necesare:

Chat GPT este un mare ajutor când vine vorba de instalat tool-uri pe care nu le-ai mai folosit în trecut, oferind instrucțiuni ordonate și clare, ajutând și la eficientizarea procesului de căutare, salvând timpul pe care l-ai fi petrecut verificând mai multe surse web pentru a găsi comenzile specifice sistemului tău de operare.

Ex prompt:

Întrebare: “Am un proiect în asp.Net 6.0 mvc c# și vreau să încerc să implementez primul modul cu restler. ce ar trebui să fac? care sunt pașii în detaliu să pot instala toate cele necesare”

O parte din răspuns:

#### 0. Pregătiri rapide

Ai nevoie de	Detalii	
.NET 6 SDK	deja instalat pentru proiect	
Python ≥ 3.12	cerință de build pentru RESTler <a href="#">GitHub</a>	
Docker (opțional)	cea mai simplă variantă de rulare izolată	
Swagger/OpenAPI	fișierul <code>swagger.json</code> generat de aplicația ta	

## 1. Expune specificația Swagger din proiect

### 1. Adaugă Swashbuckle

```
bash
```

[Copy](#)[Edit](#)

```
dotnet add package Swashbuckle.AspNetCore
```

### 2. În Program.cs :

```
csharp
```

[Copy](#)[Edit](#)

```
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
...
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```

### 3. Rulează aplicația:

```
arduino
```

[Copy](#)[Edit](#)

```
dotnet run
```

Deschide `https://localhost:7136/swagger/v1/swagger.json` (portul poate diferi) și salvează-l ca `swagger.json` într-un folder dedicat, ex. `./restler-work`.

RESTler are doar două cerințe minime: fișier OpenAPI + (opțional) script de autentificare [GitHub](#)

## 2. Găsirea motivului pentru generarea incorectă a fișierului “swagger.json”

După instalarea Swagger, aveam probleme cu fișierul generat de acesta, deoarece părea să fie gol și era neutilizabil pentru RESTler. Chat GPT ne-a ajutat să aflăm rapid motivul pentru care fișierul nu era generat corespunzător.

### Întrebare:

“E normal sa arate asa swagger.json? { "openapi": "3.0.1", "info": { "title": "Developer-Toolbox", "version": "1.0" }, "paths": { }, "components": { } } ”

O parte din răspuns:

Bună întrebare — nu, acel `swagger.json` nu e complet. Dacă `paths` este gol ( `"paths": { }` ), înseamnă că Swagger nu a reușit să detecteze niciun endpoint din API-ul vostru.

## ? De ce se întâmplă asta?

Cele mai probabile cauze:

### 1. ✗ Nu aveți controllere decorate corect cu `[ApiController]` și `[Route(...)]`

Swagger are nevoie ca endpointurile voastre să fie definite clar în controller-ele ASP.NET pentru a le putea documenta.

#### Exemplu corect:

```
csharp
[ApiController]
[Route("api/[controller]")]
public class UsersController : ControllerBase
{
    [HttpGet]
    public IActionResult GetAllUsers()
    {
        // return lista de utilizatori
    }
}
```

### 2. ✗ Folosiți doar `ControllersWithViews` în loc de API pur

Dacă proiectul vostru este mai mult MVC (cu Razor Pages sau Views), atunci Swagger nu va "vedea" decât endpointurile cu `HttpGet`, `HttpPost`, etc., din controllere de tip `ControllerBase`.

#### ✓ Cum verifici?

1. Caută în proiect clase care:

- sunt în folderul `Controllers`
- se termină în `Controller`
- extind `ControllerBase` (nu doar `Controller`)
- au `[HttpGet]`, `[HttpPost]`, `[Route]`

2. Dacă folosești `Controller` (pentru views), Swagger nu le va documenta.

### 3. Rezolvarea diferitelor erori

Găsirea soluțiilor pentru anumite erori poate fi un proces anevoios. De obicei trebuie fie să te uiți printr-o documentație stufoasă, fie să speri că altcineva a întâmpinat înainte problema ta și a cerut ajutorul internetului iar o persoană cu mai multă experiență i-a răspuns. Însă, cu ajutorul Chat GPT, această căutare nu ia mai mult de câteva minute.

Întrebare: “C:\Users\Miru\Desktop\TSS\restler-fuzzer\restler>py -3.10 -m restler.compile --help C:\Users\Miru\AppData\Local\Programs\Python\Python310\python.exe: Error while finding module specification for 'restler.compile' (ModuleNotFoundError: \_\_path\_\_ attribute not found on 'restler' while trying to find 'restler.compile’)”

O parte din răspuns:

Ah, da! 🧠 Eroarea asta apare dintr-un motiv comun:

Când încerci să rulezi `restler.compile` cu `-m`, Python crede că `restler` e un pachet instalat global, nu un modul local din folder.

#### 🔧 Rezolvare: Rulează scriptul direct din folderul corect

##### 🔄 1. Navighează în folderul `restler-fuzzer/restler/engine`

bash

Copy

Edit

```
cd C:\Users\Miru\Desktop\TSS\restler-fuzzer\restler\engine
```

Acolo sunt scripturile reale (`restler.py`, `compile.py`, etc.)

##### ▶ 2. Rulează direct cu Python 3.10

bash

Copy

Edit

```
py -3.10 restler.py compile --api_spec ../../../../swagger.json --output_dir ../../../../output/compile
```

⚠️ Atenție la căile relative:



## Modulul 2:

În cadrul modulului 2 am avut nevoie de inteligență artificială în primul rând pentru a găsi o modalitate practică de a implementa mecanismele **VulnRISKatcher**. Întrucât nu aveam un dataset pentru fine-tuning a trebuit să găsim un model suficient de bine antrenat pentru a ne ajuta. Astfel, ne-am mai consultat cu Claude[3] pentru a ne asigura că alegem un model potrivit:

**A** deci este bun modelul microsoft/codebert-base?? o sa clasifice vulnerabilitati in cod?

Am primit ca răspuns:

Microsoft/codebert-base nu este specializat direct pentru identificarea vulnerabilităților în cod. Este un model de bază preantrenat pe cod sursă din diverse limbaje, dar nu este specific configurat sau antrenat pentru detectarea vulnerabilităților de securitate.

Am am avut nevoie de ajutor și când întâmpinam erori și probleme de care părea că nu mai scăpăm:

**A** Nu se poate încărca modelul ML: The repository for mahdin70/CodeBERT-VulnCWE contains custom code which must be executed to correctly load the model. You can inspect the repository content at <https://hf.co/mahdin70/CodeBERT-VulnCWE>.

It looks like the vulnerability detection API is trying to load a machine learning model from the Hugging Face repository ( `mahdin70/CodeBERT-VulnCWE` ), but it's encountering an error.

This error message is actually coming from Hugging Face's model loader and indicates that the model requires custom code to be executed for loading, which might be blocked for security reasons.

Într-un final, am căutat și am găsit pe cont propriu un model care să ne satisfacă nevoile.

Am mai folosit Claude pentru a mă ajuta cu noțiuni pe care nu le mai folosisem. Spre exemplu, nu știam exact cum trebuie implementat un API cu ajutorul Flask:

**A** imi poti arata cum implementez un api in python cu flask?

Desigur, te pot ajuta să implementezi un API în Python folosind Flask. Flask este un framework web ușor și simplu de utilizat, ideal pentru crearea API-urilor rapide.

Iată un ghid pas cu pas pentru implementarea unui API de bază cu Flask:

Așadar, pentru modulul 2 am folosit AI, mai specific Claude, pentru a ne lămurii în legătură cu noțiuni necunoscute, pentru a rezolva mai rapid erori și pentru a ne ajuta să alegem un model cât mai potrivit.

## Modulul 3:

În cadrul modulului 3, am apelat la ChatGPT pentru a mă documenta mai clar și eficient. Fiind la început cu testarea automată și cu analiza statică a codului, am avut multe întrebări. Nu știam exact ce unelte să folosesc sau cum funcționează fiecare, așa că am început prin a cere câteva exemple.

Primul lucru de care aveam nevoie era o unealtă de testare automată, unitară pentru proiectul nostru. Am descoperit xUnit, un framework popular folosit chiar și de Microsoft, compatibil cu .NET Core.

### ✓ Unelte pentru unit testing în .NET

#### 1. xUnit.net

- Framework modern de testare pentru .NET Core/.NET 5+
- Folosit în mod uzual de Microsoft în propriile proiecte

Am cerut apoi exemple concrete de cum se scriu testele și cum se folosesc elementele de bază din xUnit.

#### feature-uri utile xunit pentru teste unitare

xUnit este un framework popular pentru teste unitare în .NET. Oferă un set bogat de funcționalități pentru a scrie, organiza și executa teste. Iată cele mai utile **feature-uri xUnit** pentru teste unitare:

#### ✓ 1. [Fact] – Teste de bază

- Atributele `[Fact]` definesc un test simplu, fără parametri.

```
csharp
[Fact]
public void TestMethod1()
{
    Assert.Equal(4, 2 + 2);
}
```

După ce m-am mai lămurit cu partea de testare, am trecut la a găsi o unealtă pentru analiza codului. Întreband ChatGPT-ul, am descoperit Roslyn, dezvoltată chiar de Microsoft.

ce unelte putem folosi pentru analiza codului în .net și unit testing

Pentru analiza codului și testare unitară în .NET, ai la dispoziție o gamă largă de unelte care acoperă diverse nevoi: de la verificarea calității codului, la detectarea bugurilor, la scrierea și rularea testelor unitare. Iată o listă împărțită pe categorii:

### Unelte pentru analiză de cod în .NET

- 1. Roslyn Analyzers (Microsoft.CodeAnalysis)**
  - Parte din compilatorul .NET
  - Permite analiză statică la compilare
  - Suport pentru reguli personalizate

L-am mai întrebat apoi să-mi arate câteva funcții utile. Mi-a arătat cum pot folosi Roslyn pentru a extrage proprietăți și a analiza atributele din codul sursă.

functii utile roslyn

Roslyn, cunoscut oficial ca .NET Compiler Platform, este o platformă puternică dezvoltată de Microsoft pentru lucrul cu codul sursă C# și VB.NET. Ea oferă API-uri pentru analiză, generare și transformare de cod sursă. Mai jos sunt câteva **funcții utile** (sau API-uri, clase și metode) din Roslyn care sunt frecvent utilizate:

### 1. Parsare cod sursă

Folosită pentru a converti text brut în arbore de sintaxă (Syntax Tree).

```
csharp
var tree = CSharpSyntaxTree.ParseText(@"
class MyClass {
    void MyMethod() {
        Console.WriteLine("""Hello""");
    }
}
");
var root = tree.GetRoot(); // SyntaxNode
```

Deci, pentru modulul 3, folosind ChatGPT, am reușit rapid să aleg un framework de testare, să înțeleg cum se scriu testele și cum să analizez codul pentru a le genera. Fără aceasta unealtă, mi-ar fi luat mult mai mult timp să înțeleg totul.

**Referințe:**

- [1] OpenAI, Chat GPT, <https://chatgpt.com/>, Data generării: 11 aprilie 2025
- [2] TestLab: An Intelligent Automated Software Testing Framework, Tiago Dias, Arthur Batista, Eva Maia, Isabel Praça , <https://arxiv.org/abs/2306.03602>, Ultima accesare: 16 mai 2025
- [3] Anthropic, Claude 3.7 Sonnet, <https://claude.ai/>, Data generării: 25 aprilie 2025