# Projektuppgift

*Programmering i C#.NET, dt071G*

**Titel**
Katt o skratt app

**Marie Lindström**

Mittuniversitetet
MID SWEDEN UNIVERSITY

**Campus Härnösand** Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
**Campus Östersund** Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

**MITTUNIVERSITETET**
**Avdelningen för informationssystem och -teknologi**

**Författare:** Marie Lindström, mali1910@student.miun.se
**Utbildningsprogram:** Webbutveckling, 120 hp
**Huvudområde:** Datateknik
**Termin, år:** VT, 2021

# Innehållsförteckning

# 1 Introduktion

Jag ska i detta projekt skapa en applikation i C# med ramverket .NET. Uppgiften är väldigt fri och jag får skapa en applikation som använder de teorier och tekniker vi studerat i kursen.

Som jag har uppfattat det har vi i kursen fokuserat på att bekanta oss med utvecklingsmiljön för C#. Att kunna använda en modern utvecklingsmiljö för programmering och genom detta kunna skapa enkla objektorienterade program i främst utvecklingsverktyget Visual Studio 2019. Inom objektorienterad programmering ingår att kunna använda sig av klasser och objekt för att skapa program. Kursen har även berört felhantering i skapade program.

Dessa kunskaper ska jag nu omsätta i en valfri app.

Jag har valt att titta närmare på hur man kan läsa in json data till en C# app från ett öppet API. Jag hade först en tanke om att läsa in alla aktuella platsannonser för Webbutvecklare från arbetsförmedlingens öppna api. Jag tänkte att det är en intressant app för oss som nu snart ska ut och söka dessa arbeten. Det visade sig dock vara (för mig ialla fall ) omöjligt att få det att fungera med deras api nycklar och det var omöjligt att få hjälp från dem. Jag gjorde det efter detta lätt för mig och letade reda på öppna api utan krav på nycklar. Detta gör att jag inte är helt säker på att innehållet är vad det utlovas att vara men å andra sidan tänker jag att det inte är det innehållet som är viktigt här utan hur jag löst uppgiften. Appen blev nu istället en app där man kan välja att läsa in information om en kattras utifrån en lista av raser eller läsa tio skämt.

Det som behöver lösas i uppgiften är att

Läsa in information från api:erna genom ett http request. Denna data ska deserialiseras och läsas ut i appen.

För detta krävs klasser för objekten och metoder för hur inläsning och utskrifter ska ske.

# 2 Teori

## 2.1 C#

C# sharp är ett objektorienterat programmeringsspråk som kan ses som en efterföljare till C++. Microsoft hade målet att skapa ett språk som kombinerar kraftfullhet med användarvänlighet[3]

Första versionen, C# 1.0 släpptes 2002 sen dess har flera versioner släppts och den nyaste C# 8.0 släpptes i september 2019. Utvecklingen har letts av Anders Hejlsberg.

C# är plattformsoberoende vilket innebär att man kan utveckla program till de flesta olika plattformar i det.

## 2.2 .NET Framework

.Net är ett ramverk som består av klassbibliotek med kod, Base Class Library (BCL). Koden är uppdelad i moduler utifrån vad som ska utföras och det finns tusentals färdiga klasser som man kan använda för att lösa saker på. Exempel på detta kan vara att läsa till en fil eller att ansluta till ett nätverk etc.

I .NET Framework ingår interpretatorn Common language runtime(CLR) som krävs för att köra program som utvecklats för ramverket.

.NET är utvecklat för Windows men man kan använda det på andra operativsystem med hjälp av Mono som är en öppen källkodsimpementation av .NET med både CLR och C# kompilator[1]

## 2.3 NuGet

NuGet är ”package” hanterare för .NET. Dvs det tillhandahåller paket som kan behövas vid programutveckling.

## 2.4 Newtonsoft.Json

Ett Json ramverk för .NET som kan laddas ner genom NuGet och används för att hantera Json serialisering.

## 2.5 HttppClient

En bas klass i .NET som används för att sända HTTP förfrågningar och ta emot svar från en källa som identifieras med ett URI.

# 3 Metod

Jag använder Visual Studio 2019 för att skapa applikationen.

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using static System.Console;
```

Dessa paket kommer att användas.

Nestonsoft.Json för serialiseringen av Json strängen från api:et.

System.Net.HTTP för att göra http anrop till api:et

Klassen Joke hanterar skämten

```
class Joke
{
    0 references
    public int id { get; set; }
    0 references
    public string type { get; set; }
    1 reference
    public string setup { get; set; }
    1 reference
    public string punchline { get; set; }
}
```

och klassen ”Breeds” o ”Data” kattraser och fakta om dessa.

```
class Breeds
{
    2 references
    public List<Data> Data { get; set; }
}
1 reference
```

```
class Data
{
    3 references
    public string breed { get; set; }
    1 reference
    public string country { get; set; }
    1 reference
    public string origin { get; set; }
    1 reference
    public string coat { get; set; }
    1 reference
    public string pattern { get; set; }
}
```

Lösningen innehåller två metoder som löser momenten i uppgiften.

- GetJokeItems som gör ett anrop till api:et ”https://official-joke-api.appspot.com/random_ten”. Svaret deserialiseras och läggs in i en foreach loop där delarna ”setup” och ”punchline” sedan skrivs ut.

- GetCatBreeds anropar api:et ”https://catfact.ninja/breeds?” Svaret deserialiseras och i en foreach loop skrivs ”breeds” ut. Användaren uppmanas att välja en kattras och skriva in det namnet följt av enter. I en foreach loop finns en if sats som kollar om index är likamed item.-breed, dvs inmatningen och om så är fallet skrivs de olika fakta ut om rasen. Det finns här även felhantering i form av en koll så att användaren verkligen matar in något annars kommer ett felmeddelande.

# 4 Konstruktion

Jag började med att forma klassen för kattraserna ”Breeds”, som innehåller en lista ”Data” med fem strängar i. Dessa görs allihop till properties med {get; }och {set;}

HTTP client initialiseras och får namnet ”client”.

En privat asyncron funktion, ”GetCatBreeds” görs och för att få respons från Api:et används ”GetStringAsync” med anrops url i som parameter. Eftersom ”async” används måste det finnas en ”await” i initialiseringen av funktionen. ”void Main” måste även ändras till en ”async Task” .

En deserialisering behöver göras och det görs genom att skapa ett objekt av klassen som deserialiseras med ”JsonConvert.DeserializeObject”.

Listan ”Data” loopas igenom med en foreach och alla raser skrivs ut.

Användaren får instruktion om att skriva rasen den vill veta mer om och trycka enter. Det finns även instruktion om att skriva med stor bokstav där det görs i listan. Jag provade att använda ”ReadLine().ToLower();” för att slippa detta men det fungerar inte. Detta för att strängen ”index” jämförs med ”item.breed” i en if sats för att skriva ut rätt information om rätt ras. Det som den då jämförs med är det som står i api:et och det matchar inte om det blir liten input och leder då till att inget skrivs ut. Har dock inte löst hur man skulle kunna göra detta istället.

Inputen testas i en if sats och om strängen är tom får användaren ett felmeddelande om att ”Fältet får inte vara tomt”. Om det finns input i fältet jämförs det med datan i api:et och skriver ut information vid matchning.

Funktionen för ”GetJokeItem” görs i stort sett på samma sätt med deserialisering av json och en foreach loop som går igenom objekten och skriver ut de önskade delarna. Jag valde här att lägga in en ”ReadKey” efter att första delen av skämtet lästs in. Detta för att alla skämt var i två delar, med en fråga som inledning och svaret som punchlinen sen och då blev detta upplägg mest effektfullt.

En meny görs i en while loop där användaren får tre val. Läsa om katter, läsa jokes eller avsluta. 1,2 o x. Dessa val hanteras i en switch sats där case ”1” initialiserar utläsningen av kattraser. Case ”2” skriver ut skämten efter en uppmaning om att trycka enter för att läsa punchlinen och för nästa joke. Case x avslutar programmet.

# 5 Resultat

Jag har konstruerat en fungerande om än enkel applikation. Det delar som behöver fungera gör det till stor utsträckning. Jag läser in data från två olika api i json format. Detta deserialiseras och läses ut som jag hade tänkt.

Klasserna och funktionerna fungerar som de ska. Det hade varit önskvärt att kunna fånga upp även inmatning av kattraser med liten bokstav i början men detta har jag inte lyckats lösa.

# 6 Slutsatser

Under arbetet med denna applikation la jag ner mycket tid på att undersöka hur man kan använda api:n i applikationer i c#. Jag hade en tanke om att appen skulle läsa ut yrken inom webbutveckling från arbetsförmedlingens api. När inte detta gick att lösa så blev det lite panik och tidsbrist. Jag bestämde mig för att göra en liknande app med ett annat api. Det är inte helt lätt att hitta ett lämpligt api att använda och efter mycket letande fick jag helt enkelt ge mig och konstatera att jag får nöja mig med den infon som finns i dem och göra det bästa av situationen. Jag ville ändå att det var ett api där man kunde läsa in i mer än ett steg i interaktion med användaren och det var därför det blev katt apiet där jag kunde välja att hämta raserna först och att användaren väljer vilken ras den vill läsa mer om.

Delar att utveckla vidare i appen är att lägga till mer felhantering, ev vid felstavning. Att kunna skriva med småbokstäver och ändå få träff på ras. I delen med jokes skulle det kanske vara bra att kunna lämna innan man läst alla tio skämten om man vill.

Det har varit en intressant uppgift och C# är ett tilltalande och roligt programmeringsspråk. Det har dock varit svårt att veta vilken nivå man ska lägga sig på i detta projekt då det varit väldigt fritt. Friheten har givetvis även varit rolig då man kunnat fokusera på det som känts intressant.

# Källförteckning

[1] Mark J Price, ”C# 8.0 and .NET Core 3.0”
Modern Cross-plattform Development (Fourth edition)

[2] C# Tutorial (C Sharp) (w3schools.com) Hämtat 2021-01-18

[3] Introduktion till .NET – csharpskolan.se