

TAXA opgaven

Mål for opgaven

... er at lære om API og Blazor

Opgave tager udgangspunkt i taxaselvskabet 4x27.

- Her skal du udvikle en prisberegner for taxa-selskabet [4x27](#), som kan give en forventet pris estimat, for en given tur.
- Programmet skal bruge i deres kunde service afdeling, hvor deres medarbejderne modtager opkald fra kunder, som vil have oplyst en given pris for et given tidspunkt.
- Programmet skal højst for alle scenarier, som er beskrevet på selvskabtes hjemmeside for deres prismodel.

Her vises et eksempel på et mockup:

The mockup shows a two-panel interface. The left panel is a form with the following fields: 'km' (text input), 'Døgn tillæg' (dropdown with 'Dag / Nat' selected), 'Vogn type' (text input), and 'Tilæg' (text input). The right panel is a Google Maps view showing a route from 'Choose starting point, or click on the map' to 'Choose destination...'. Below the map, there is a 'Delays' section with the text 'Heavy traffic in this area' and a red traffic icon indicating a 'Slowdown on 02' with a '13-min delay'. The map shows Copenhagen with labels for 'NORrebro', 'Rosenborg Slot', 'INDRE BY', and 'FRIE CHRI'.

Krav til applications niveau

1. Lav et program i blazor eller Windows Forms der kan beregne priser ud fra givende priser og valgmuligheder.
2. Programmet skal have google maps indlejret og automatisk sætte indtastet start-destination.

Reference links:

[4x27 priser](#)

<https://www.google.com/maps/dir/>

Note: Med google maps kan man finde afstanden i km mellem startingpoint og destination (her vist til højre i mockup'et). Google vil oplyse din afstanden i km, du kan overføre det via API'et eller du kan indtastes i programmet til venstre.

Valgfri udvidelse til opgaven.

- Adressen <https://www.google.com/maps/dir/> skal lægges i programmet. Efter "/dir/" tilføjes der yderligere til linket med et plustegn "+" i stedet for mellemrum.
- Dette gøres for at bygge URL'en op i programmer bag ud fra kundes "ønskede krav" / "indtastet værdier" til Taxa-turen.

Krav til opgaven

1. Udarbejde et tidsplan ved hjælp af et gant-diagram. (se eksempel)
2. Du skal selv designe et mockup, som du udvikler opgaven ud fra.
3. Vælg Framework
 - Arbejder du med opgave i Blazor skal du bruge .NET 8.0
4. Du skal gøre brug af Classes, constructors, properties, methods.
5. Open/close princippet skal følges. Logikken skal være i sin egne klasse og metoderne herunder skal kun returnere beregninger til Viewet / User Interfacet - (UI).
6. Det er forbudt at tilføje static til ens kode. (Det skal i udgangspunktet kun fremkomme i klassens program main metod.)
7. Du skal gøre brug af objekter:

```
var obj = new >>class name<<();  
//Eksempel:  
Person person = new Person();
```

Note:

1. I skal hjælpe hinanden i gang med opgaven.
2. Er det helt umuligt, at komme i gang med opgaven. Så skal du starte med, at udvikle de logiske beregninger er afkoblede UI/view-delen. Dette gøres ved hjælp af open/close princippet. Her kan du starte med, at lave logikken til opgaven, som i en konsol-applikation. Logikken kan så senere overføres en til en til Winforms/Blazor applikationen.

Bilag:

- Eksempel på kode der er afkoble fra viewet
 - Efterfølgende kan vi i viewet kalde ToTitleCase uden at skrive hele funktionen.
 - Dette eksempel er vist med static metoder. Husk at du ikke må bruge keyword-static

Eksempel på afkoblet kode

```
using System;

namespace MyProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!".ToTitleCase());
        }
    }

    public static class StringManipulation
    {
        public static string ToTitleCase(this string title)
        {
            return CultureInfo.CurrentCulture.TextInfo.ToTitleCase(title.ToLower());
        }

        public static Dictionary<string, string>
        SplitFirstNameMiddelNamesAndLastName(string names)
        {
            var middlename = string.Empty;

            if (names.Split(' ').Length >= 3)
            {
                for (int i = 1; i < names.Split(' ').Length; i++)
                {
                    middlename += names.Split(' ')[i] + " ";
                }
            }

            return new Dictionary<string, string>()
            {
                ["firstname"] = names.Split(' ').First(),
                ["middlenames"] = middlename.Trim(),
                ["Lastname"] = names.Split(' ').Last()
            };
        }
    }
}
```

Eksempel på Gant diagram

```
gantt
title Project Schedule
dateFormat YYYY-MM-DD
axisFormat %m/%d

section Planning
Define Project : 2024-03-01, 7d
Research : 2024-03-08, 14d
Define Requirements : 2024-03-22, 7d

section Development
Design : 2024-03-29, 21d
Implementation : 2024-04-19, 28d

section Testing
Unit Testing : 2024-05-19, 14d
Integration Testing : 2024-06-02, 14d

section Deployment
Deploy : 2024-06-16, 7d
User Training : 2024-06-23, 14d

section Maintenance
Ongoing Support : 2024-07-07, 30d
```