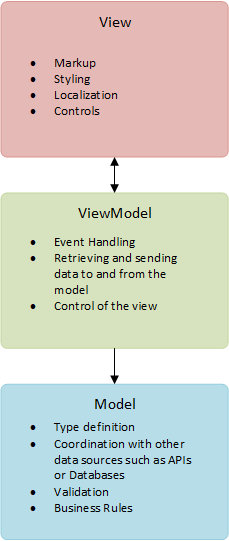
**Model-View-ViewModel (MVVM)**

MVVM adalah software design pattern untuk memisahkan program logic dan user interface controls. MVVM membantu mengatur kode dan memecah program menjadi modul untuk development, updating dan reuse of code yang simple dan cepat. Design ini untuk mempermudah jika program akan digunakan pada UI Framework yang lain.

MVVM dibagi menjadi View, ViewModel dan Model :

* View, adalah layer UI yang menerima user input.
  + - * ViewModel, adalah Layer yang berada diantara View dan Model untuk mengatur controls pada View dan mengambil data dari model.
* Model, adalah layer yang berisi program logic, validasi dan pengambilan data dari API.

**Example**

1. Create New Blazor WebAssembly App Project dan create Folder DTOs, Models dan ViewModels

Graphical user interface, text

Description automatically generated

1. Create Model

Create FetchDataModel.cs dengan code berikut:

**using** BlazorMVVM**.**DTOs**;**

**using** System**.**Net**.**Http**.**Json**;**

**namespace** BlazorMVVM**.**Models

**{**

**public** class FetchDataModel

**{**

**private** **readonly** HttpClient \_httpClient**;**

**public** FetchDataModel**(**HttpClient httpClient**)**

**{**

\_httpClient **=** httpClient**;**

**}**

**public** **async** Task**<**WeatherForecast**[]>** RetrieveForecast**()**

**{**

**return** **await** \_httpClient**.**GetFromJsonAsync**<**WeatherForecast**[]>(**"sample-data/weather.json"**);**

**}**

**}**

**}**

* Inject HTTPClient di constructor.
* Get data dari Json yang terdapat pada folder sample-data/weather.json

1. Create ViewModel

Create FetchDataViewModel.cs dengan code berikut:

**using** BlazorMVVM**.**DTOs**;**

**using** BlazorMVVM**.**Models**;**

**namespace** BlazorMVVM**.**ViewModels

**{**

**public** class FetchDataViewModel

**{**

**private** **readonly** FetchDataModel \_fetchDataModel**;**

**public** List**<**WeatherForecast**>** forecasts**;**

**public** FetchDataViewModel**(**FetchDataModel fetchDataModel**)**

**{**

\_fetchDataModel **=** fetchDataModel**;**

**}**

**public** **async** Task RetrieveForecastsAsync**()**

**{**

var loResult **=** **await** \_fetchDataModel**.**RetrieveForecast**();**

forecasts **=** loResult**.**ToList**();**

**}**

**}**

**}**

* Inject FetchDataModel di constructor.
* Panggil method RetrieveForecast pada FetchDataModel dan convert menjadi list. Tamping hasil ke property forecasts.

1. Ubah ViewModel

Ubah FetchData.razor menjadi:

@page "/fetchdata"

**<**PageTitle**>**Weather forecast**</**PageTitle**>**

**<**h1**>**Weather forecast**</**h1**>**

**<**p**>**This component demonstrates fetching data **from** the server**.</**p**>**

@if **(**FetchDataViewModel**.**forecasts **==** **null)**

**{**

**<**p**><**em**>**Loading**...</**em**></**p**>**

**}**

**else**

**{**

**<**table class**=**"table"**>**

**<**thead**>**

**<**tr**>**

**<**th**>**Date**</**th**>**

**<**th**>**Temp**.** **(**C**)</**th**>**

**<**th**>**Temp**.** **(**F**)</**th**>**

**<**th**>**Summary**</**th**>**

**</**tr**>**

**</**thead**>**

**<**tbody**>**

@foreach **(**var forecast **in** FetchDataViewModel**.**forecasts**)**

**{**

**<**tr**>**

**<**td**>**@forecast**.**Date**.**ToShortDateString**()</**td**>**

**<**td**>**@forecast**.**TemperatureC**</**td**>**

**<**td**>**@forecast**.**TemperatureF**</**td**>**

**<**td**>**@forecast**.**Summary**</**td**>**

**</**tr**>**

**}**

**</**tbody**>**

**</**table**>**

**}**

@code **{**

**[**Inject**]** **public** FetchDataViewModel FetchDataViewModel **{** **get;** **set;** **}**

**protected** **override** **async** Task OnInitializedAsync**()**

**{**

**await** FetchDataViewModel**.**RetrieveForecastsAsync**();**

**}**

**}**

* Panggil method RetrieveForecastsAsync pada FetchDataViewModel

1. Add Model dan ViewModel ke Depedency Container di Program.cs

builder**.**Services**.**AddTransient**<**FetchDataModel**>();**

builder**.**Services**.**AddTransient**<**FetchDataViewModel**>();**