On the AspergillosisEPR site several scheduled background tasks are run periodically at various times.

Registered with DI container in Startup.cs

services.AddScoped<IViewRenderService, ViewRenderService>();

services.AddSingleton<IHostedService,PatientVoriconazoleLevelBackgruondTask>();//monday

services.AddSingleton<IHostedService, ImmunoglobulinUpdateBackgroundTask>(); //runs tuesday

services.AddSingleton<IHostedService, EmptyPostCodesUpdateScheduledTask>(); //runs wednesday

services.AddSingleton<IHostedService, PatientTestResultBackgroundUpdateTask>();//thursday

services.AddSingleton<IHostedService, PatientRadiologyUpdateBackgroundTask>();//friday

services.AddSingleton<IHostedService, PatientICD10DiagnosesBackgroundTask>();//saturday //services.AddSingleton<IHostedService, PatientSGRQImporterBackgroundTask>();//saturday at 12:15

services.AddSingleton<IHostedService, PatientAdministrationSystemStatusTask>(); //sunday

services.AddHostedService<QueuedHostedService>();

services.AddSingleton<IBackgroundTaskQueue, BackgroundTaskQueue>();

IHostedService interface is used, and most code adapted from:

<https://thinkrethink.net/2018/02/21/asp-net-core-background-processing/>

When implementing the IHostedService interface, you have two methods to implement:

StartAsync()

StopAsync()

The classes associated with the above tasks are all defined in:

In AspergillosisEPR/BackgroundTasks

Each task class contains a string called Schedule defined like: (this one from ImmunoglobulinUpdateBackgroundTask)

protected override string Schedule => "59 23 \* \* 2";

it defines a Cron expression for scheduling the task.

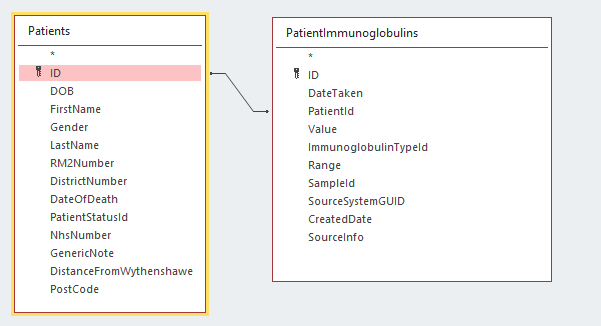
|  |  |
| --- | --- |
| Cron expression | |
| 1  2  3  4  5  6  7  8  9 | ┌───────────── minute (0 - 59)  │ ┌───────────── hour (0 - 23)  │ │ ┌───────────── day of month (1 - 31)  │ │ │ ┌───────────── month (1 - 12)  │ │ │ │ ┌───────────── day of week (0 - 6) (Sunday to Saturday;  │ │ │ │ │                                       7 is also Sunday on some systems)  │ │ │ │ │  │ │ │ │ │  \* \* \* \* \* |

FluentScheduler for Asp.Net Core is employed to do the scheduling

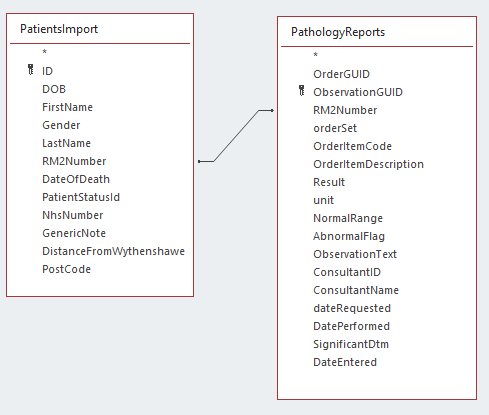
**Tasks:**

PatientVoriconazoleLevelBackgruondTask -Updates table PatientImmunoglobulins from

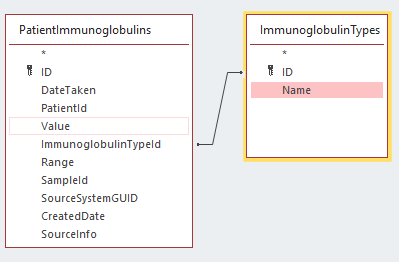
**AspEPR3 Database Links for Immunoglobulins**



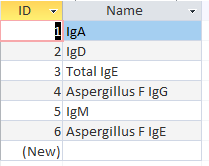
**AspEPRImport Database showing Pathology table**



**Showing Immunoglobulin Types in Database**



**Table ImmunoglobulinTypes**



**IgD** doesn’t seem to be used.

**Model class for PatientImmunoglobulin**

using AspergillosisEPR.Lib.Exporters;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

namespace AspergillosisEPR.Models.Patients

{

public class PatientImmunoglobulin : Exportable

{

public int ID { get; set; }

public int PatientId { get; set; }

public int ImmunoglobulinTypeId { get; set; }

public string SampleId { get; set; }

public string Range { get; set; }

[DataType(DataType.Date)]

[Display(Name = "Date Taken")]

[DisplayFormat(DataFormatString = "{dd-MM-yyyy}")]

public DateTime DateTaken { get; set; }

public decimal Value { get; set; }

public ImmunoglobulinType ImmunoglobulinType { get; set; }

public decimal? SourceSystemGUID { get; set; } = 0;

public DateTime? CreatedDate { get; set; }

override public List<string> ExcludedProperties()

{

return new List<string>()

{

"PatientId", "Patient", "ImmunoglobulinType", "SampleId", "Range"

};

}

public static string IgFromCode(string code)

{

return Codes()[code];

}

public static Dictionary<string, string> Codes()

{

var codes = new Dictionary<string, string>();

codes.Add("ASPIGG", "Aspergillus F IgG");

codes.Add("ASPIGE", "Aspergillus F IgE");

codes.Add("IGEX2", "Total IgE");

codes.Add("WIGM", "IgM");

codes.Add("WIGA", "IgA");

return codes;

}

}

}

The red hi-lited code shows lookup table between **PatientImmunology** table and **PathologyReports** Table