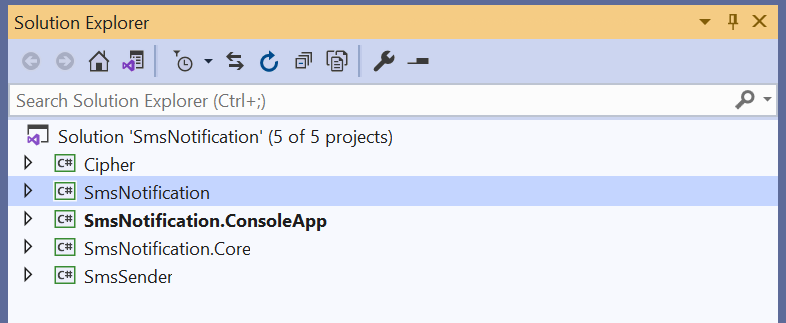
**Co będzie robiła usługa**

Usługa powinna sprawdzać błędy w tabeli **Errors** i jeżeli takie wystąpią, to powinna wysłać powiadomienia SMS.

Można użyć zewnętrznej biblioteki **ServerSMS** lub **SMSApi**.  
Skorzystaj z darmowych pakietów.  
Na ich stronach powinna być szczegółowa dokumentacja.  
Dane konfiguracyjne do tej usługi przechowuj w pliku konfiguracyjnym.

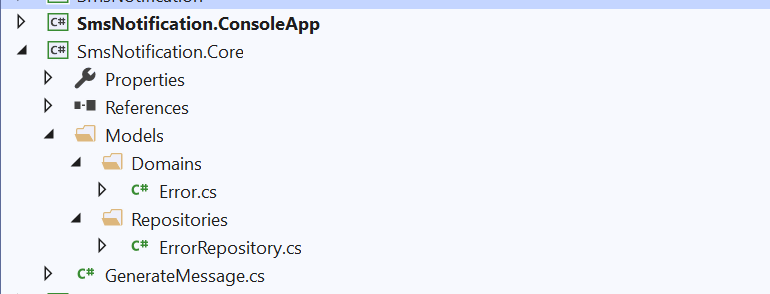
**Solucja SmsNotification**



Oprócz głównego projektu w obrębie solucji tworzymy dodatkowe projekty:

**Cipher** – class library – logika odpowiedzialna za szyfrowanie  
**SmsNotification.Core** – class library  
**SmsSender** – class library – będzie zawierał logikę odpowiedzialną za wysyłanie SMS’ów  
**SmsNotification.ConsoleApp** – aplikacja konsolowa ułatwiająca testowanie usługi

**SmsNotification.Core** – będzie zawierał definicję klasy error, klasę zawierającą repozytorium błędów oraz metodę formatującą treść SMS’a



|  |
| --- |
| namespace SmsNotification.Core.Models.Domains  {  public class Error  {  public int Id { get; set; }  public string Message { get; set; }  public DateTime Date { get; set; }  }  } |

|  |
| --- |
| namespace SmsNotification.Core.Models.Repositories  {  public class ErrorRepository  {  public List<Error> GetLastErrors(int intervalInMinutes)  {  // TODO: Pobieramy z bazy  return new List<Error>  {  new Error {Message = "błąd testowy 1", Date = DateTime.Now},  new Error {Message = "błąd testowy 2", Date = DateTime.Now},  };  }  }  } |

|  |
| --- |
| namespace SmsNotification.Core  {  public class GenerateMessage  {  public string GenerateErrors(List<Error> errors, int interval)  {  if (errors == null)  throw new ArgumentNullException(nameof(errors));  if (!errors.Any())  return String.Empty;  var message = $"Błędy z ostatnich {interval} minut.\n";    foreach (var error in errors)  {  message +=  $"{error.Message} {error.Date.ToString("dd-MM-yyyy HH:mm")}\n";    }  message += $@"Automatyczna wiadomość SMS wysłana z aplikacji";  return message;  }  }  } |

**SmsNotification** – to główny projekt

**Program.cs** – zawiera kod pozwalający na instalację i deinstalację usługi oraz kod pozwalający na uruchomienie usługi

|  |
| --- |
| namespace SmsNotification  {  static class Program  {  /// <summary>  /// The main entry point for the application.  /// </summary>  static void Main(string[] args )  {  if (Environment.UserInteractive)  {  var parameter = string.Concat(args);  switch (parameter)  {  case "--install":  ManagedInstallerClass.InstallHelper(new[]  {Assembly.GetExecutingAssembly().Location});  break;  case "--uninstall":  ManagedInstallerClass.InstallHelper(new[]  {"/u",Assembly.GetExecutingAssembly().Location});  break;  default:  break;  }  }  else  {  ServiceBase[] ServicesToRun;  ServicesToRun = new ServiceBase[]  {  new SmsNotyficationService()  };  ServiceBase.Run(ServicesToRun);  }  }  }  } |

**SmsNotyficationService.cs** (pierwotnie Service.cs) – to klasa uruchamiająca naszą usługę.

|  |
| --- |
| namespace SmsNotification  {  public partial class SmsNotyficationService : ServiceBase  {  public SmsNotyficationService()  {  InitializeComponent();  }  protected override void OnStart(string[] args)  {  }  protected override void OnStop()  {  }  }  } |

Dodajemy kilka pól:  
Pole klasy Timer w którym ustawimy co jaki czas ma się wykonywać usługa.  
Pole przechowujące interwał czasowy, oraz pola będące instancjami klas, które będziemy potrzebować

|  |
| --- |
| public partial class SmsNotyficationService : ServiceBase  {  private static readonly NLog.Logger Logger = NLog.LogManager.GetCurrentClassLogger();  private Timer \_timer; // ważne wybieramy namespace: System.Timers  private readonly int \_intervalInMinutes;    private ErrorRepository \_errorRepository = new ErrorRepository();  private GenerateMessage \_generator = new GenerateMessage();  private Sms \_sms = new Sms(); |

W konstruktorze odczytujemy parametry z ustawień aplikacji i tworzymy instancję obiektu Timer

|  |
| --- |
| public SmsNotyficationService()  {  InitializeComponent();    \_intervalInMinutes = int.Parse(ConfigurationManager.AppSettings["IntervalInMinutes"]);  int intervalInMiliseconds = \_intervalInMinutes \* 60 \* 1000;  \_timer = new Timer(intervalInMiliseconds);  } |

W metodzie **OnStart()**  
Podpinamy pod obiekt \_timer metodę, która będzie się wywoływać cyklicznie.  
Startujemy Timer i logujemy do pliku fakt wystartowania usługi

|  |
| --- |
| protected override void OnStart(string[] args)  {  \_timer.Elapsed += DoWork;  \_timer.Start();  Logger.Info("Service Started ...");  } |

W metodzie **OnStop()**  
Logujemy fakt zatrzymania usługi

|  |
| --- |
| protected override void OnStop()  {  Logger.Info("Service Stopped ...");  } |

W metodzie **DoWork()**  
W sekcji TRY wywołujemy metody, które mają się wykonywać, w przypadku wystąpienia wyjątku logujemy ten fakt do pliku i rzucamy ponownie ten  
wyjątek.

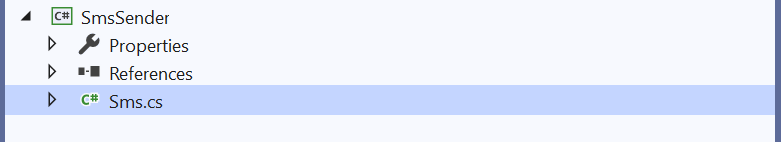
|  |
| --- |
| private async void DoWork(object sender, ElapsedEventArgs e)  {  try  {  await SendError();  }  catch (Exception ex)  {  Logger.Error(ex, ex.Message);  throw new Exception(ex.Message);  }  } |

W sekcji **TRY** metody **DoWork()** wywołujemy metodę **SendError()**

|  |
| --- |
| private async Task SendError()  {  var errors = \_errorRepository.GetLastErrors(\_intervalInMinutes);  if (errors == null || !errors.Any())  return;  // TODO:Send SMS  await \_sms.Send("Błędy", \_generator.GenerateErrors(errors, \_intervalInMinutes),   "502 010 110");  Logger.Info("Error was sent ...");  } |

|  |
| --- |
| **Uwagi dotyczące asynchronicznego wywoływania metod**  Metoda asynchroniczna musi być wywoływana ze słowem kluczowym **await**  Metoda zewnętrzna w której wywołujemy metodę asynchroniczną musi być również asynchroniczna, czyli w deklaracji musi być użyte słowo kluczowe **async**, oraz zamiast **void** powinna ona zwracać typ **Task**. |

**SmsSender** – to biblioteka klas, projekt w którym zaimplementujemy  
sam mechanizm wysyłania sms’ów



Pusty kod klasy Sms.cs

|  |
| --- |
| namespace SmsSender  {  public class Sms  {  public async Task Send(string title, string body, string phoneNumber)  { }  }  } |

