# General

* .ConfigureAwait(false) Is not actually needed in the ASP.NET core as there is no SynchronisationContext and it does not have any effect in this solution.

# HomeController.cs

* It is not ok that Privacy endpoint is synchronously calling a service and blocking the calling thread.
  + Proposed solution: await should be used when calling \_privacyDataService.GetPrivacyDataAsync()

# StatisticMiddleware.cs

* InvokeAsync: The whole idea to create a task, attach a continuation to it and then wait (block the calling thread!) a hardcoded amount of time is bad. It would even stop working if the time required to register the visit increases and exceeds the hardcoded Thread.Sleep(3000)
  + Proposed solution: use await when calling async methods of \_statisticService
* If it is not necessary to show up to date statistics in the page, then the \_statisticService.RegisterVisitAsync could be invoked without waiting for it to be completed. This would improve the page loading time drastically.
  + However, it is assumed that the requirement is to have completely up to date statistics, so the page loading performance is limited by CloudService performance.

# ManualAssistant.cs

* RequestAssistanceAsync try part: Similarly to the StatisticMiddleware.InvokeAsync, Thread.Sleep(5000) used for synchronization is a very bad idea.
  + Proposed solution: use await when calling async methods of \_supportService
* RequestAssistanceAsync catch part: Unnecessary creation of a wrapping tasks when string could already be returned directly.
  + Proposed solution: directly return the error message.