# Adaptive api (CodingKata\_4)

***Rules***

1. Strictly practice TDD

a. Follow the: Red-Green-Refactor cycle.

b. Stop and read your code before refactoring.

2. Structure the code based on the onion architecture

***The Kata***

Below code is an example of adaptive code the result of the function is different based on the parameters it receives. This is in violation with the SOLID principles.

* Create a core layer with a domain object for image
* Create services and repositories to receive an image
* Store the image in a database in binary format
* Implement proper exception handling and write some unittests for it

**Project to refactor: Kata4 can be found in git**

private Image GetLiveImage(string imageId, bool throwExc = false)

        {

            try

            {

                Image img = \_imageService.FindImage(new Guid(imageId));

                if (img == null)

                {

                    var imgNotFoundExc = new Exception($"The image with given id couldn't be found. On page: {TemplateNamePrefix}. Id: {imageId}");

                    if (throwExc) throw imgNotFoundExc;

                    LoggingService.Log(Enums.LogLevel.Warn, imgNotFoundExc);

                }

                return img;

            }

            catch (Exception exc)

            {

                if (throwExc) throw;

                LoggingService.Log(Enums.LogLevel.Warn, exc);

                return null;

            }

        }