FROM PLIGHTS WITH PIRATES TO CATCHING CORRUPT COPS…

A TECHNICAL RETROSPECTIVE


Ray's Jazz @ FOYLES
All Ray's jazz events are free to attend
TODAY
Saturday 6 February 6pm
IDLE WORDS
Duo
on the Ray's Jazz stage
in the Café
www.foyles.co.uk

BORN IN 1984, BUT...

WHO WATCHES THE WATCHMEN?

vigilantapps

ABOUT ME

- Work in User Behaviour / Employee Threat Detection, Born in 1984 / “Big Brother” (co-incidence..?)
- Customers include: DSTL, Mi5, Mi6, **REDACTED**, Microsoft, Verizon Business, McDonald’s, itsu, POD, NHS, C&W, Expedia, French Telecom, Bid-up TV, BidSmart and most of the UK Police Forces’ ACU teams
- Inventor of ethical P2P IM message interception
- Consulted for FCA for MiFiD II financial regulations
- Not technically qualified to do anything!!

ACTIVE PROJECTS DEPLOYED

- ~= 20,000 UK Police (ACU/PS)
  10TB+ databases
  Patent for tracking user behaviour against physical security assets & IoT
- National NHS Wales' Governance System (NIIAS)
  60-million accesses per day
  Correlating data between dozens of internal/closed systems
- All global Oil Future Traders audited WhatsApp and WeChat
  Auditing 1-million+ trades per month
  1,000 licenses active, 3,000 commitment by partners before EO 2019
  USA patent for ethical P2P encrypted IM message interception
- Essex counter-fraud system
  Detected £1m last year
- McDonald's global contract for LP Solution
  Various retail innovations awards won

vigilantapps

OH YEH AND..

- Only 10 employees...
- ...3 of which are developers
- ...and 1 awesome pipeline solution

FUTURE-PROOF DOMAIN MODEL

```
public class EventDetails
{
    public EventDetails(string eventName, DateTimeOffset? when)
    {
        this.EventName = eventName;
        this.When = when.GetValueOrDefault(DateTimeOffset.Now);
        this.Payload = new NameValueCollection();
    }

    public string EventName { get; }

    public DateTimeOffset When { get; }

    public NameValueCollection Payload { get; }
}
```

```
// GOOD: little/no architecture in concept
public interface IPlugin
{
    Task SetupAsync();

    Task TeardownAsync();
}

// GOOD: interface segregation
public interface IEmitEvents
{
    event EventHandler<EventDetailsEventArgs> EventEmitted;
}

// GOOD: plug-ins are super easy to test
public interface IReceiveEvents
{
    Task ReceiveEventAsync(EventDetails eventDetails);
}
```

EXAMPLE

```
public class CronPlugin : IPlugin, IEmitEvents
{
    private readonly ICronTaskManager cronTaskManager;
    private readonly IEnumerable<ICronTask> tasks;

    public CronPlugin(
        ICronTaskManager cronTaskManager,
        IEnumerable<ICronTask> tasks)
    {
        this.cronTaskManager = cronTaskManager;
        this.cronTaskManager.TaskTriggered += this.TaskTriggered;
        this.tasks = tasks;
    }

    public event EventHandler<EventDetailsEventArgs> EventEmitted;

    public async Task SetupAsync()
    {
        await this.cronTaskManager.StartTasksAsync(this.tasks);
    }

    public async Task TeardownAsync()
    {
        await this.cronTaskManager.ShutdownAsync();
    }

    private void TaskTriggered(object sender, CronTaskEventArgs e)
    {
        var eventDetails = new EventDetails();

        eventDetails.Payload["TaskName"] = e.TaskName;
        eventDetails.Payload.Add(e.MetaData);

        this.EventEmitted?.Invoke(this, new EventDetailsEventArgs(eventDetails));
    }
}
```

CONFIGURATION-DRIVEN IOC FOR BUSINESS LOGIC

```
plugins:
  - type: cron
    constructor:
      tasks:
        converter: cron_task_from_string
        values:
          - Name=Weekday;Cron=0 0/5 * * * MON-FRI
          - Name=Weekend;Cron=0 0/8 * * * SAT-SUN
    events:
      - Policies=(TaskName='Weekend');OnTrigger=WeekendTask
      - Policies=(TaskName='Weekday');OnTrigger=WeekdayTask

  - type: message_to_log
    events:
      - ListenFor=WeekendTask;Policies=(Always='True')
      - ListenFor=WeekdayTask;Policies=(Always='True')
```

POLICY-SET DSL

```
policy:
  BankAccountInClipboard:
    conditions:
      - (eventType='ClipboardChanged' AND clipboardData='Text' AND textContent^='[\d]{8,9}')
    reactions:
      - clearClipboard
      - screenshot: allWindows

  DLP.PrintScreenPressedInDataApp:
    conditions:
      - (eventType='KeyPressed' AND specialKeyName='PrintScreen' AND FocusedWindow=~'$(DataApps)')
```

ADVANTAGES

- Very re-usable, quickly produce production-ready code for new projects
- Uses ~= 4MiB of RAM for McDonald's
- New projects inherit 'battle testing' of existing ones
- Pipeline can be linear or star or whatever based on the routing interface implementation chosen
- Almost no architectural code, so almost everything is easy to change
- Product vs Project separation
- Developers work on Core product calibre, whilst PS/DevOps work on project BL and project implementation

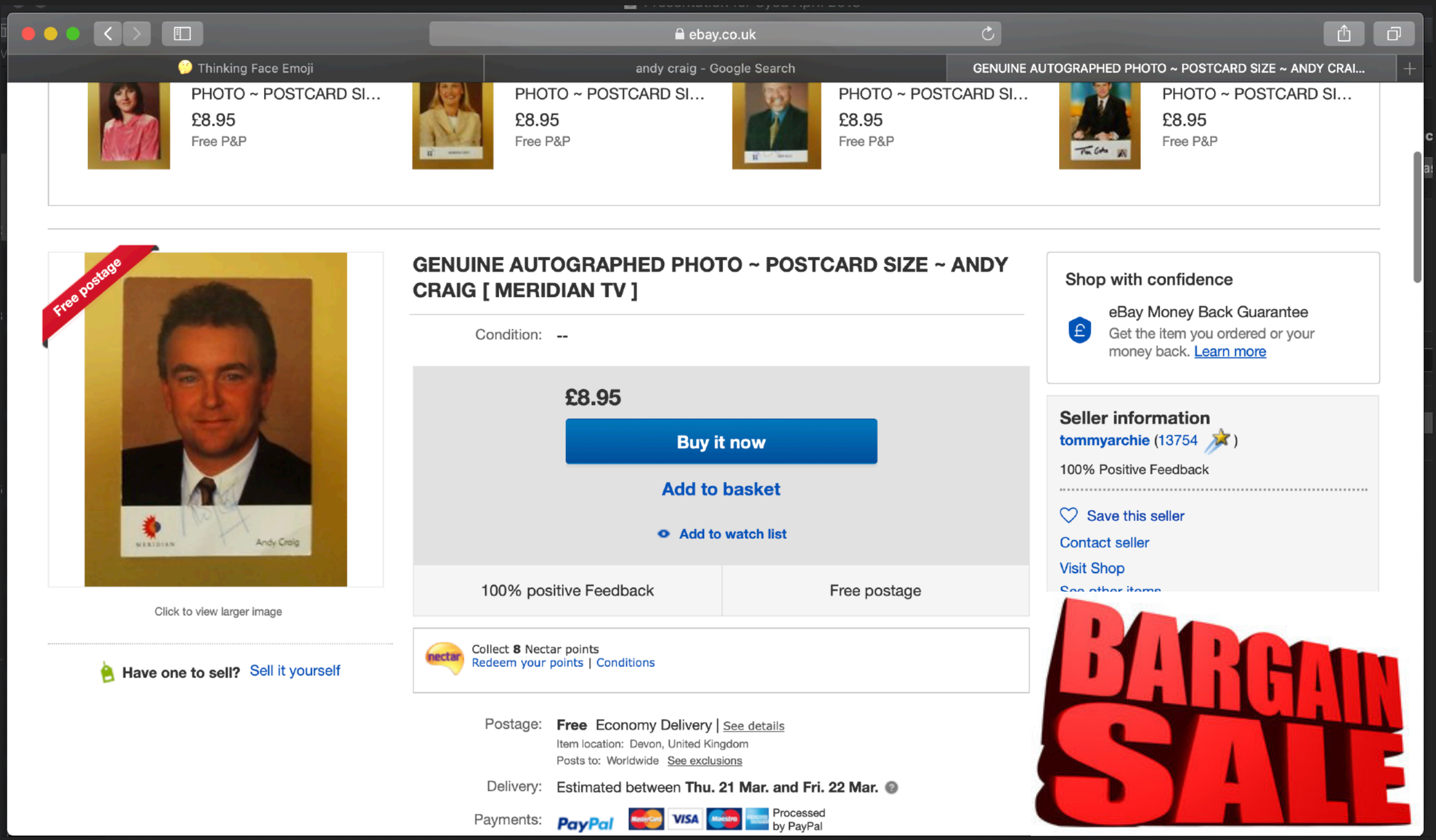
DISADVANTAGES

- Project/Sales complain cannot prove pen. testing on the product, each project implementation has to be tested individually in-situ (on purpose, this is a big positive technically!)
- Configuration files can become quite long and complex for massive projects (e.g. NHS Wales' Governance System)
- Developers can often be narrow-minded (Open/Closed)
- Takes a long time for new employees to 'get it'
- Might actually be better to adapt output and input from each plug-in and use a proper object model?

“TRUST ME…

WE’RE FROM HEAD OFFICE”

“WE’RE FROM HEAD OFFICE”



THERE ARE ONLY TWO HARD THINGS IN COMPUTER SCIENCE: CACHE INVALIDATION AND NAMING THINGS.

-- PHIL KARLTON

(YOU KNOW... OF NETSCAPE!)

55

As his only son, and colleague with him at Netscape from 95-97, I can attest that my dad did indeed throw that quote around, on more than one occasion. I'm fairly confident that he originated it (he was fond of coming up with clever quippets), though I haven't been able to figure out how it disseminated so widely over the past couple of decades. I'll keep looking around in old web archives and mails to see if I can dig something up.

share improve this answer

answered Aug 10 '17 at 19:53

David Karlton

551 ● 3 ● 2

S.O.L.I.D.

Single Responsibility

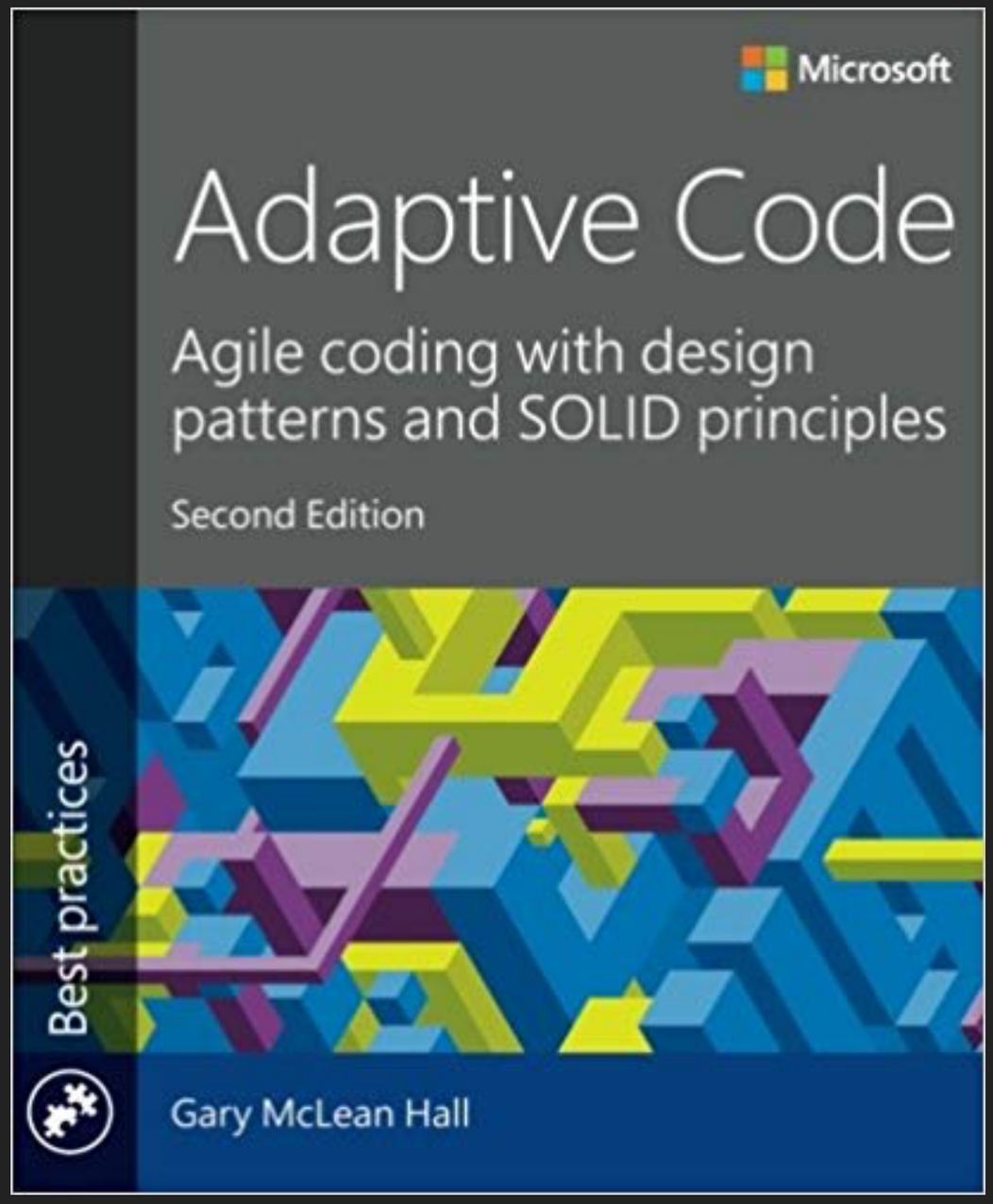
Open/Closed

Liskov Substitution

Dependency Injection

THE BEST (CODING) BOOK I'VE EVER READ

- Ignore the Agile section (you already know it!)
- Best practical demonstration of SOLID in C# I've seen
- Things will click!



INTERFACE

WHAT

ABSTRACTION

CONCEPT

CLASS

HOW

CONCRETE

IMPLEMENTATION

DO YOU CARE HOW YOUR CAR IS REPAIRED?

NAMING CLASSES & SINGLE RESPONSIBILITY

```
// BAD: narrow-minded name assumes will always write to a file
public interface IFileWriter
{
    // BAD: :path is not part of concept
    Task WriteFileAsync(string path, byte[] contents);
}

// BAD: naming of class does not show single reponsibility
public sealed class FileWriter : IFileWriter
{
    public async Task WriteFileAsync(string path, byte[] contents)
    {
        await File.WriteAllBytesAsync(path, contents);
    }
}
```

NAMING CLASSES & SINGLE RESPONSIBILITY

```
// GOOD: high-level concept is without 'architecture'
public interface IStoreDataForLater
{
    Task WriteDataAsync(string name, byte[] contents);
}

// GOOD: name clearly documents responsibility of class
public sealed class LocalFileStoreDataForLater : IStoreDataForLater
{
    private readonly string parentDirectory;

    // GOOD: implementation details in ctor
    public LocalFileStoreDataForLater(string parentDirectory)
    {
        this.parentDirectory = parentDirectory;
    }

    /// <inheritdoc/>
    public async Task WriteDataAsync(string name, byte[] contents)
    {
        // GOOD: domain-specific logic using ctor parameters
        var filePath = Path.Combine(this.parentDirectory, name);
        await File.WriteAllBytesAsync(filePath, contents);
    }
}
```

NAMING CLASSES & SINGLE RESPONSIBILITY

```
public interface IWebRequest
{
    // YAGNI: simple example doesn't include support for headers
    Task<IWebResponse> GetAsync(string url);

    Task<IWebResponse> PostAsync(string url, byte[] body);
}

// BAD: class name is utterly useless
public sealed class WebRequest : IWebRequest
{
    public Task<IWebResponse> GetAsync(string url)
    {
        // ...
    }

    public Task<IWebResponse> PostAsync(string url, byte[] body)
    {
        // ...
    }
}
```

NAMING CLASSES & SINGLE RESPONSIBILITY

```
// GOOD: naming of class represent 'how' (implementation detail)
public sealed class HttpClientWebRequest : IWebRequest
{
    public async Task<IWebResponse> GetAsync(string url)
    {
        using (var http = new HttpClient())
        using (var response = await http.GetAsync(url))
        {
            // GOOD: adapter pattern abstracts response object
            return new HttpResponseMessageAdapter(response);
        }
    }

    public Task<IWebResponse> PostAsync(string url, byte[] body)
    {
        // ...
    }
}
```

IT'S ALL ABOUT I!

- *IMakeWebRequests*
- *ISubmitOrders*
- *IApproveOrders*
- *IKickAss*
- *IAdaptJsonStringsTo<CustomerModel>*

CONCRETE IMPLEMENTATIONS

- HttpClientWebRequests
- FirebaseFunctionsOrderSubmit
- KungFuFighting
- NewtonsoftJsonDeserialiser<?>

NAMING CLASSES & SINGLE RESPONSIBILITY

```
// GOOD: class name defines single responsibility
public sealed class FirebaseFunctionOrderSubmitter : ISubmitOrder
{
    private readonly FirebaseSettings settings;
    private readonly IWebRequest http;

    // GOOD: dependencies injected into ctor
    public FirebaseFunctionOrderSubmitter(
        // GOOD: settings work well with DI
        FirebaseSettings settings,
        IWebRequest http)
    {
        this.settings = settings;
        this.http = http;
    }

    // ...
}
```

CONSTRUCTOR ISN'T A PART OF THE INTERFACE, SO SHOULD BE USED FOR DOMAIN-SPECIFIC STUFF

∴ LEAVE THOSE METHODS CONCEPTUAL

NAMING CLASSES & SINGLE RESPONSIBILITY

```
public sealed class AwsS3BucketFileWriter : IFileWriter
{
    // BAD: path doesn't work in this implementation
    // as the concept and the implementation are all mixed up
    public async Task WriteFileAsync(string path, byte[] contents)
    {
        // 🤮 🤮 🤮 🤮 🤮
        // BAD: because path is architectural we have to hack it
        var split = path.Split('|');
        var config = new AwsConfiguration
        {
            BucketName = split[0],
            ApiKey = split[1],
            Secret = split[2],
            FileName = split[3],
            MimeType = split[4],
            // BAD: breads Liscov Principal as FormatException
            // didn't exist before, but now possible
            RetryCount = int.Parse(split[5], CultureInfo.InvariantCulture),
        };
    }
}
```

NAMING CLASSES & SINGLE RESPONSIBILITY

```
public sealed class AwsS3BucketStoreDataForLater : IStoreDataForLater
{
    private readonly AwsConfiguration awsConfiguration;

    // GOOD: works well with dependency injection
    public AwsS3BucketStoreDataForLater(AwsConfiguration awsConfiguration)
    {
        this.awsConfiguration = awsConfiguration;
    }

    public async Task WriteDataAsync(string name, byte[] contents)
    {
        // GOOD: easy to integrate as file concept will
        // always have name and contents,
        // everything else is an implementation detail and belongs
        // in the ctor!
        await new AwsProxy(this.awsConfiguration).WriteFileAsync(name, contents);
    }
}
```

EXPEDIA: "ALL YOUR URINE IS BELONG TO US"

IF S IS A SUBTYPE OF T, THEN OBJECTS OF TYPE T MAY BE REPLACED WITH OBJECTS OF TYPE S WITHOUT ALTERING ANY OF THE DESIRABLE PROPERTIES OF THE PROGRAM.

Liskov Substitution Principle

Q: How do you validate user input whilst writing sub-classes without breaking the Liskov Principle?

PRACTICAL LISKOV SUBSTITUTION PRINCIPAL

```
public abstract class Car
{
    public virtual bool HasEngine { get; }

    public virtual void DriveFast(Driver driver)
    {
        Guard.Pre.MustBeTrue(this.HasEngine);
        Guard.Pre.MustBeEqual(driver.Location, DriverLocation.InDriverSeat);

        // ... logic for driving the car
    }
}
```

PRACTICAL LISKOV SUBSTITUTION PRINCIPAL

```
public class SuperCar : Car
{
    public override void DriveFast(Driver driver)
    {
        // BAD: additional pre-guarding breaks Liskov
        // as it raises the bar to entry
        Guard.Pre.MustBeGreaterThan(this.EngineCapacityLtrs, 2.5F);
        Guard.Pre.MustBeEqual(
            this.EngineNoise,
            CarNoises.SexyNoiseMakesTrousersTight);

        base.DriveFast(driver);
    }
}
```

PRACTICAL LISKOV SUBSTITUTION PRINCIPAL

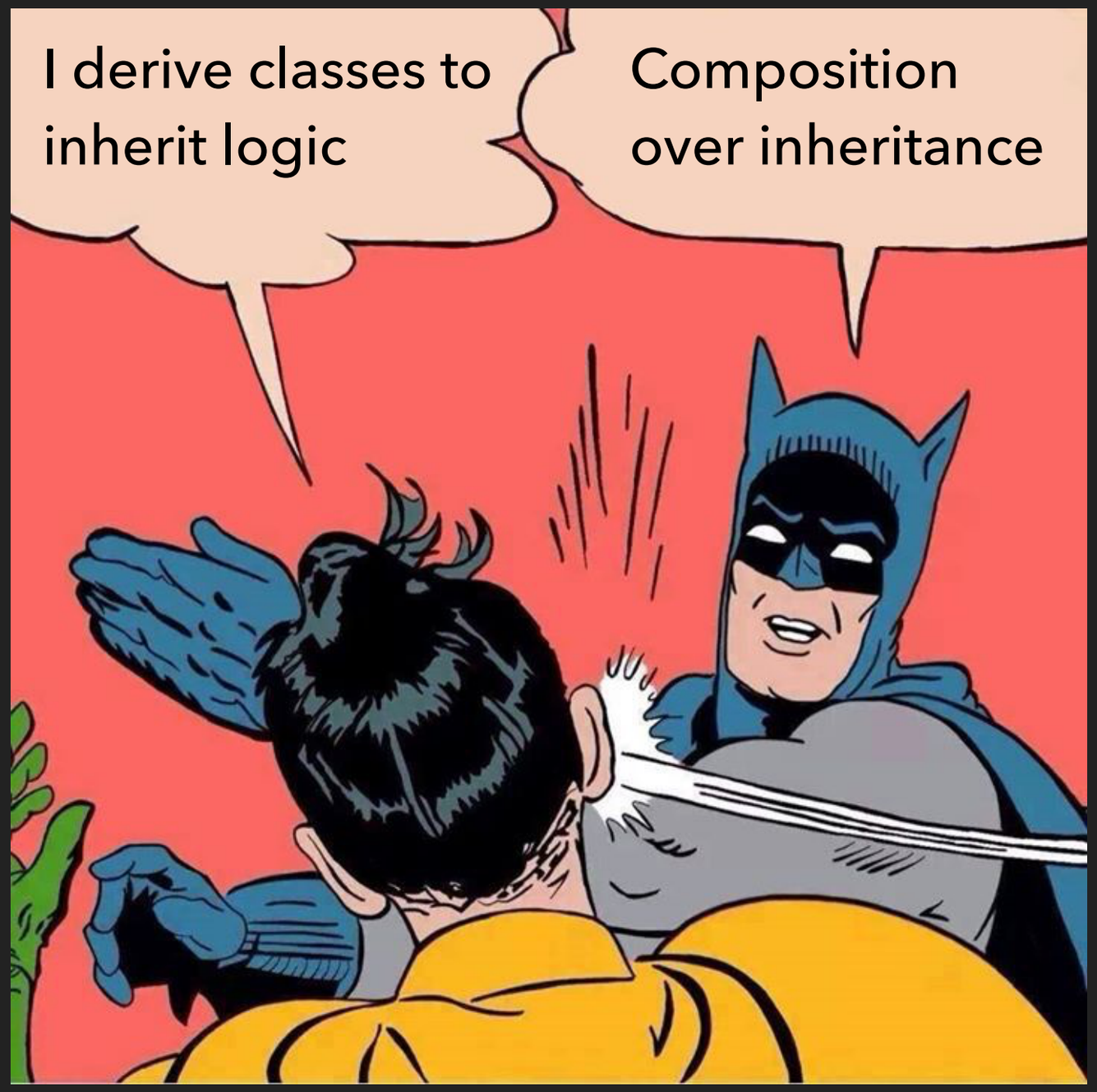
```
public class SportsCar : Car
{
    public override void DriveFast(Driver driver)
    {
        base.DriveFast(driver);

        // GOOD: ensure the state of this class is as you'd expect
        // it after applying sub-class logic
        // i.o.w. You have asserted your own logic
        Guard.Post.EnsureIsNowTrue(driver.TrousersAreTight);
    }
}
```

A: You cannot raise the ‘bar of entry’. The calling code will be unaware of this and the substitute will break existing caller’s logic under conditions the super-class would have succeeded.

...You can however add additional checking ('post-guarding') to ensure your sub-class has left its instance in the correct state before leaving the method.

Bonus points:



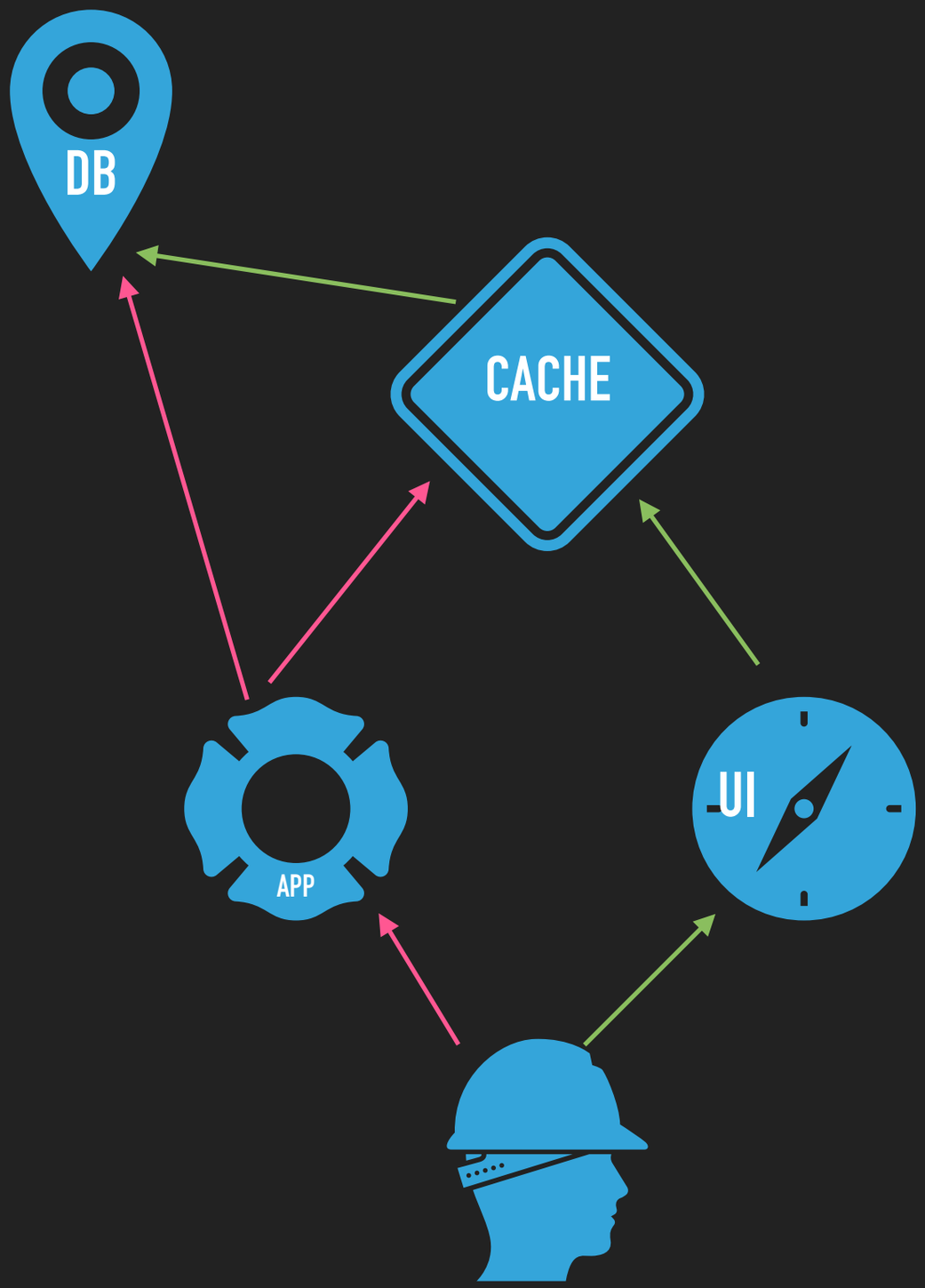

MR MESSY
Quality Street

SOMALIS IN BERKSHIRE

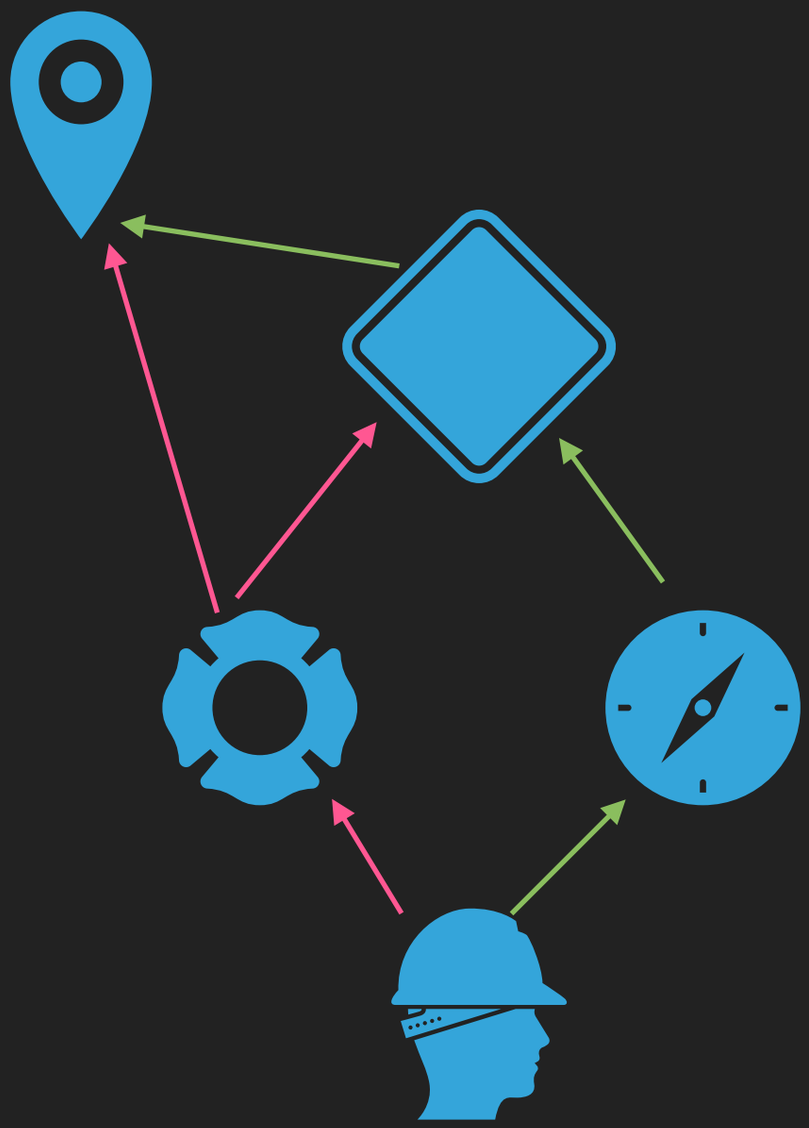
FIGHTING PIRATES WITH

MI6, MI5, FBI & CIA


TRADITIONAL CACHING INVALIDATION



TRADITIONAL CACHING INVALIDATION

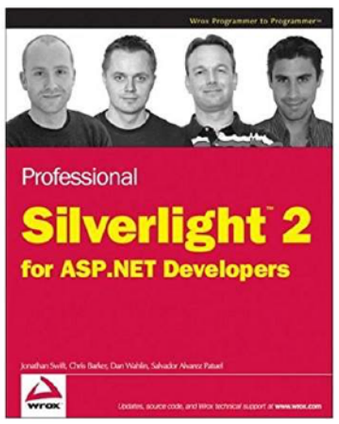


- What if DevOps/Support/Developer updates database?
- What if Cache Service is offline when a change occurs?
- Is ordering of updates important?
- No Atomicity or Consistency between DB and Caching service (Congrats, you just broken your ACID database!)

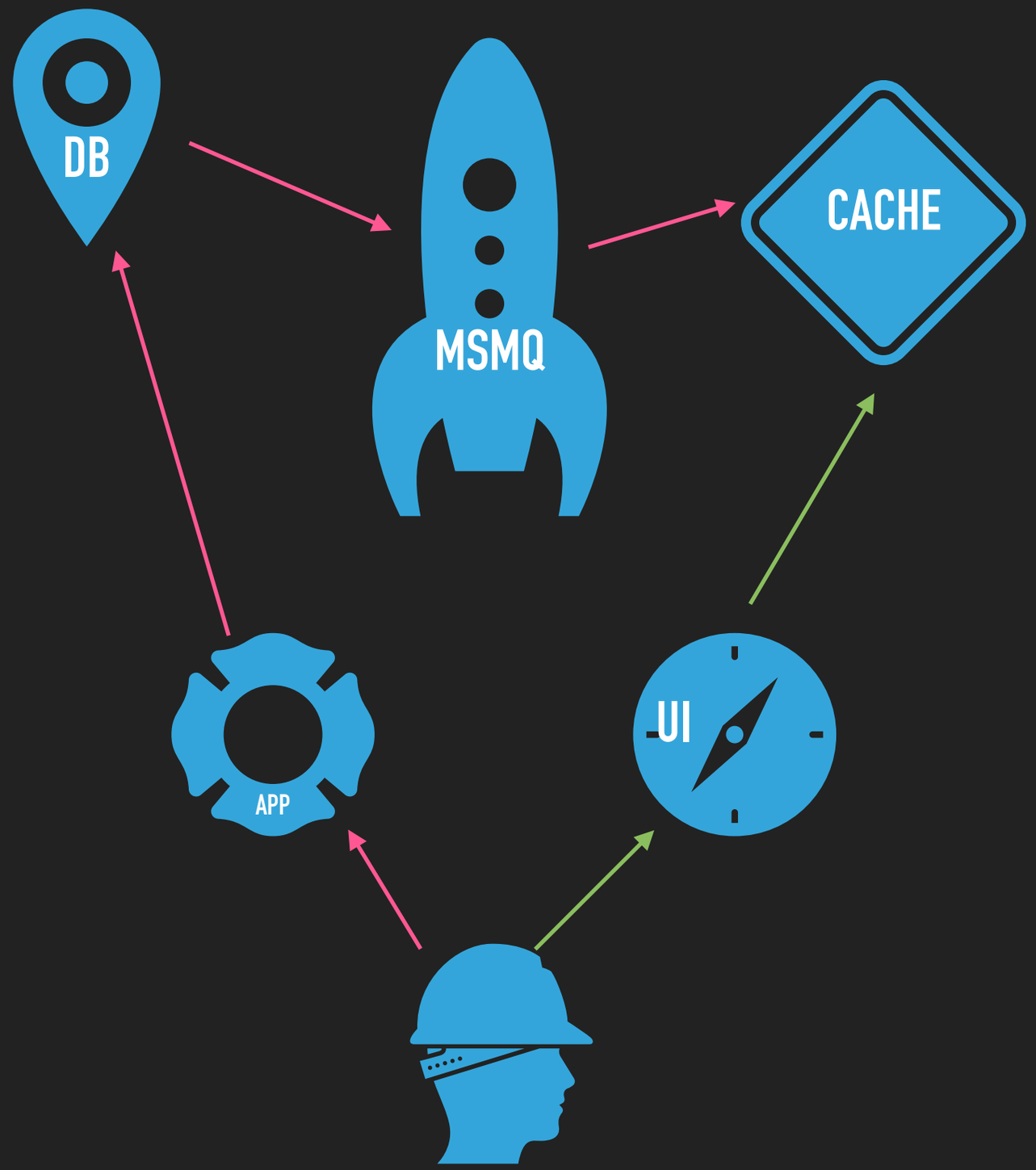
Salvador Alvarez Patuel • 1st

Partner - Head of Cognitive Transformation at IBM Spanish South America

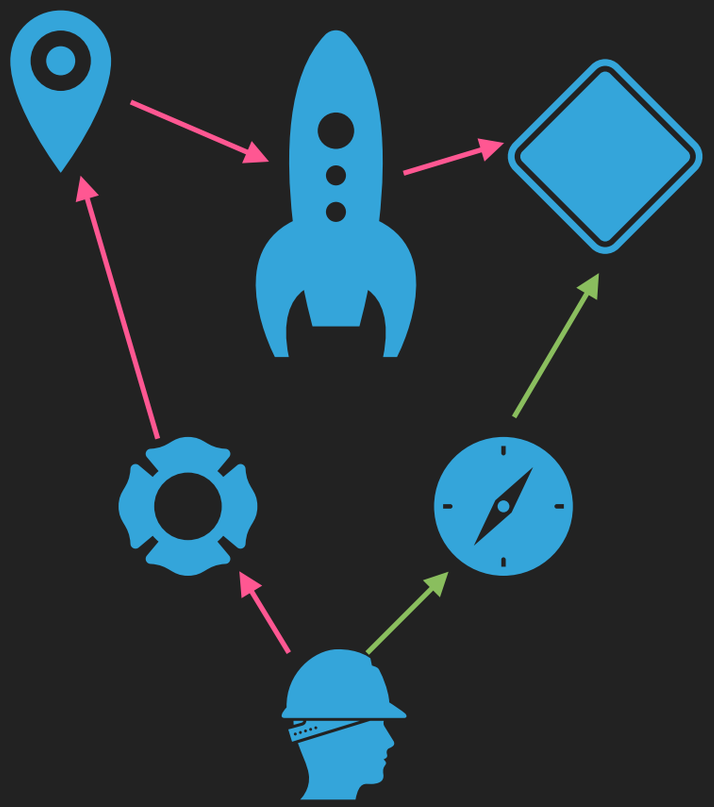
Argentina



SALVADOR'S CACHING INVALIDATOR DESIGN



SALVADOR'S CACHING INVALIDATOR DESIGN



- Database trigger fires COM Extension, sends MSMQ message, Caching Service subscribed to queue
- All database updates make it to the cache
- Cache Service goes offline, reads queue on start-up
- Queue is ordered as database changes were written
- Atomicity solved and Eventual Consistency is acceptable (Still not ACID compliant)
- Better separation of responsibilities (i.e. app service only ever updates DB), massively simplifying error handling / retry logic

"GET READY, YOU'RE ON IN 5…"

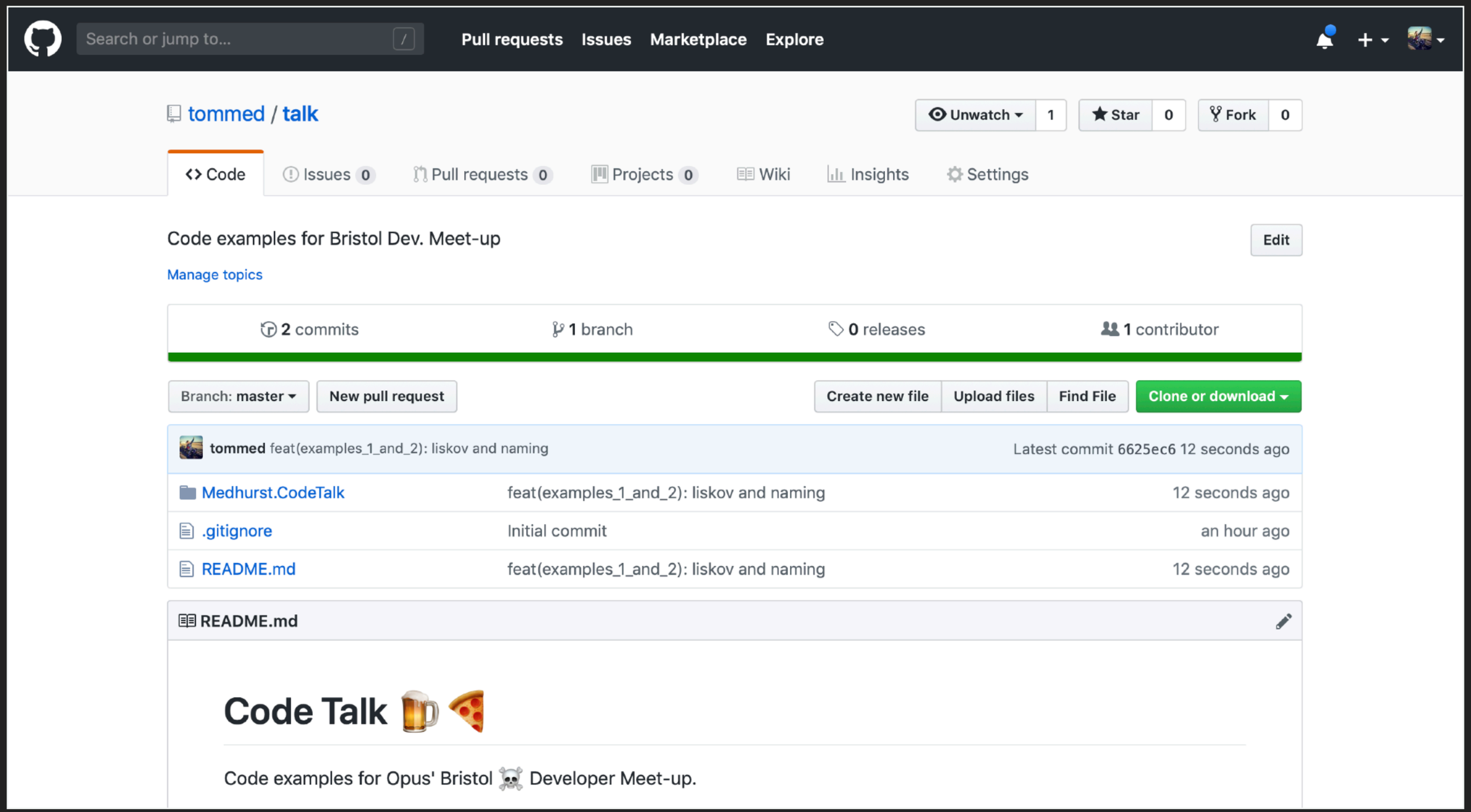
LIVE TV IN QATAR

BNP PARIBAS

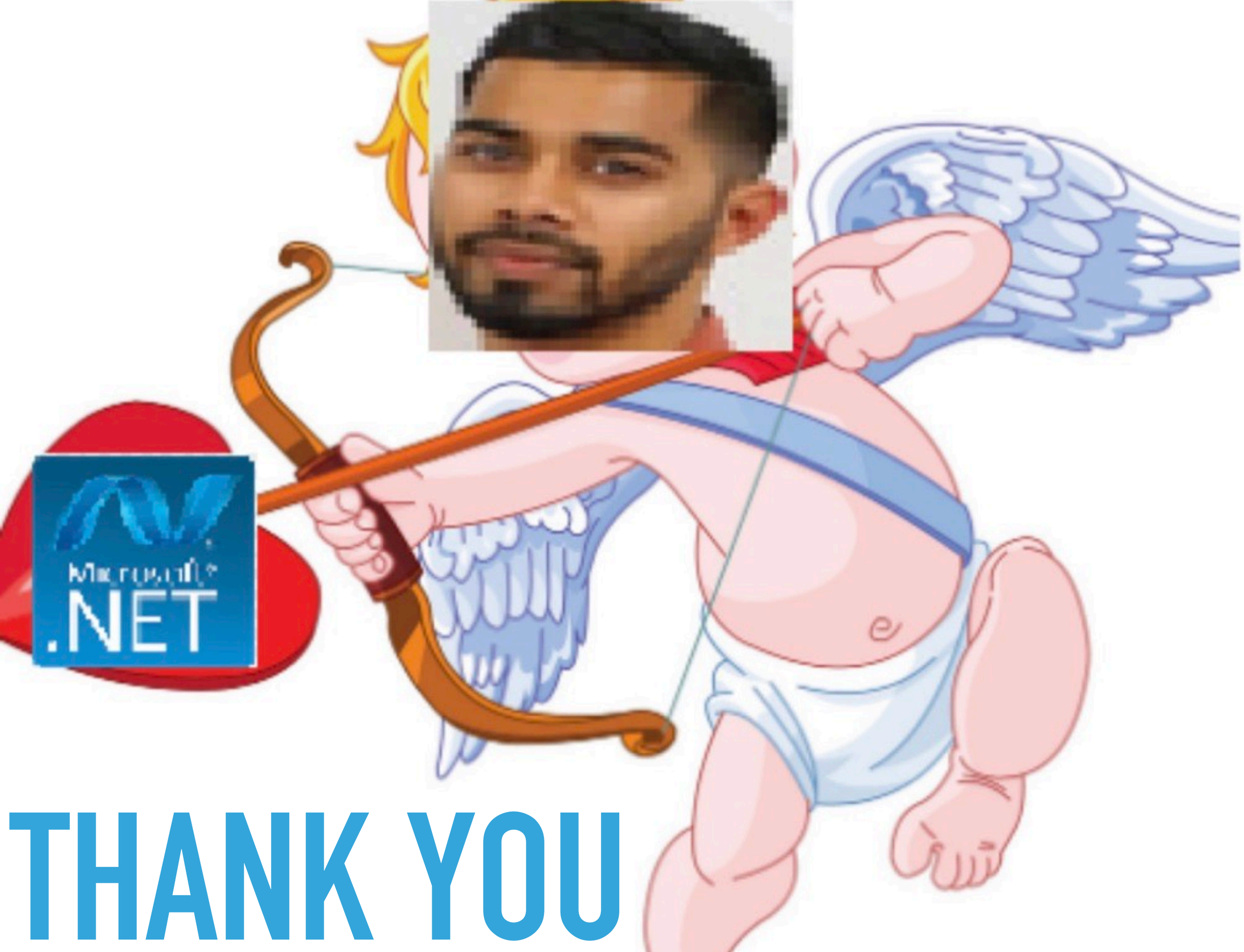
المشرق

BADOIT

github.com/**tommed/talk**



Q&A



THANK YOU