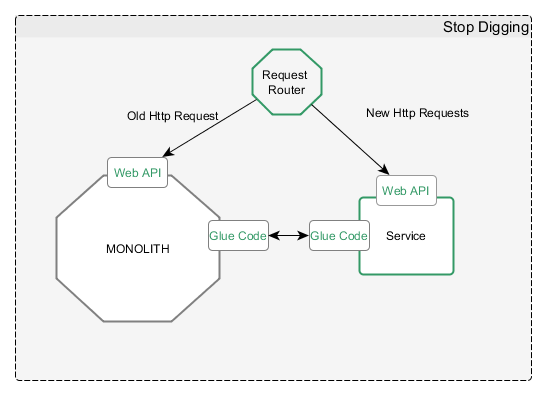
Application Modernization Plan

This document describes the process of transforming the monolithic into micro services.

Any application moving out from a big monolithic need to incrementally refactor, it means build a new application consisting of micro services, and run it in conjunction with the monolithic application. In such way the monolithic will shrink until either it disappears entirely or it becomes just another micro service.

# Strategy 1 – Stop Digging

Stop making the monolithic bigger. This means if there is a new functionality you should not add more code to the monolithic. Instead, we can try putting the new code in a standalone micro service.



* ***Request Router***: It handles incoming (HTTP) requests. The router sends requests corresponding to the new functionality to the new micro service and legacy requests to the monolith.
* ***Glue Code***: (AKA anti-corruption layer) which integrates the service with the monolithic. A service will rarely exist in isolation and will need data access to the monolith. It will read/write data owned by the monolith.

In order to access data from monolith we can (we will need to choose a way):

* *Invoke API in the monolith side.*
* *Access Monolith Database Directly.*
* *Have an own copy of data and keep sync with the monolith.*

***Note***: This will prevent the monolith to become bigger but will still have same issues than a monolith until it’s completely broken.

# Strategy 2 – Extract Services

Turn existing modules within the monolith into standalone micro services. Each time a module is extracted and turn into a service, the monolith shrinks. Once you have converted enough modules the monolith will cease to be a problem and it will tend to disappear or became small enough that it’s just another service.

Start with the easy ones which will provide us experience on extracting and refactoring.

