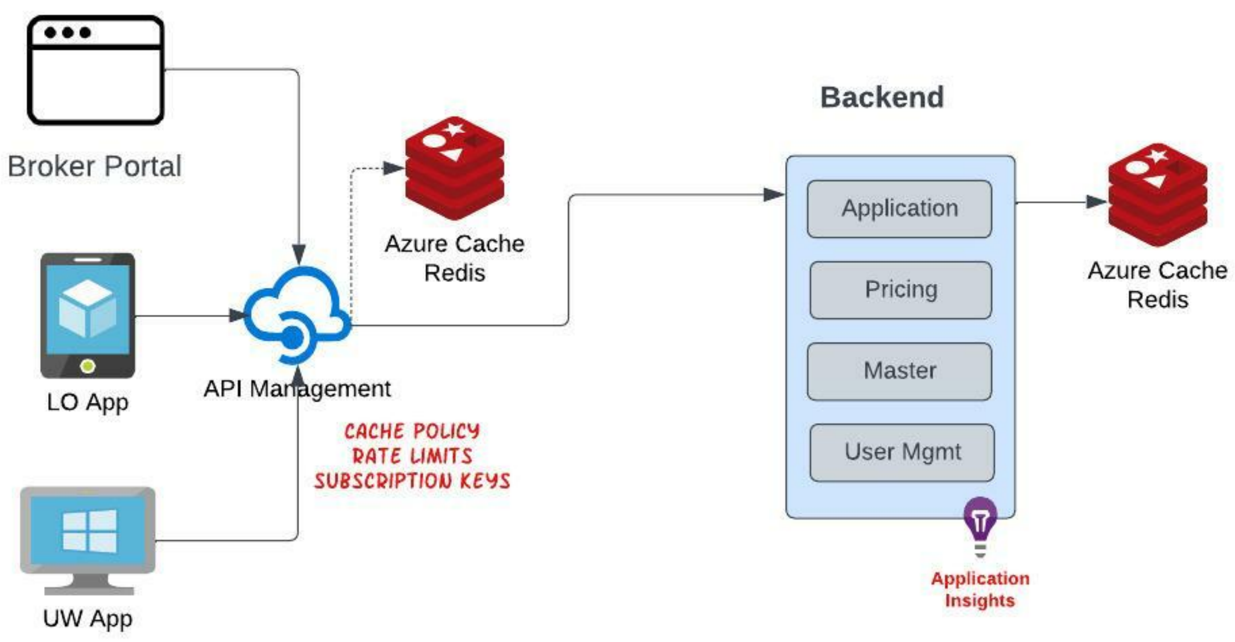
Caching Strategy for Frontends and Backends



It it recommended that any backend calls from the client applications must be routed through APIM for security, rate limiting, authentication reasons. We could use the same gateway to implement caching of responses on the APIM side of Regis cache. This is easily configured on the Azure portal. No coding is necessary.

For backend APIs we could use either Distributed Cache databases like Redis, Alachisoft or local memory. Microsoft .NET makes it very easy to implement both distributed and memory cache through standard interfaces in the framework which abstracts underlying implementation.

IDsitributedCache makes it very easy for .NET developers to implement caching in their applications.

This interface is implemented by both Memory cache and Redis cache implementation within the .NET framework. Since the developers use the interface to code against the implementation, they could rely on memory cache during local development and use Redis only for higher environments.

In the startup of your application, use the following line of code to use memory cache

```
builder.Services.AddDistributedMemoryCache();
```

for Redis, one can use following code snippet

```
builder.Services.AddStackExchangeRedisCache(options =>
 {
     options.Configuration = builder.Configuration.GetConnectionString
("MyRedisConStr");
     options.InstanceName = "SampleInstance";
 });
```

Register with the IOC and setup the caching expiry as following

```
app.Lifetime.ApplicationStarted.Register(() =>
{
    var currentTimeUTC = DateTime.UtcNow.ToString();
    byte[] encodedCurrentTimeUTC = System.Text.Encoding.UTF8.GetBytes
(currentTimeUTC);
    var options = new DistributedCacheEntryOptions()
        .SetSlidingExpiration(TimeSpan.FromSeconds(20));
    app.Services.GetService<IDistributedCache>()
                              .Set("cachedTimeUTC",
encodedCurrentTimeUTC, options);
});
```

Now all you have to do is inject `IDistributedCache` in your class constructor to start using the `GetAsync()` and `SetAsync()` methods to get and set the cache value(s).

When to use Memory cache vs Distributed cache on production

Memory Cache	**Distributed Cache (Redis, Alachisoft)**
Owned and accessed by just one process	One memory repository for multiple processes
Tightly coupled with process that owns the cache	Decoupled with the process that is accessing, can outlive the process
Simple to implement and manage (none actually!)	Easy to implement, needs DevOps support to manage
Read speed - 100 nanoseconds	Read speed - in sub micro seconds Alachisoft has a client replication model, in which the cache data is not just in the cluster node but also replicated to all the nodes in which the applications are running, this makes the read/write ops faster than accessing the cluster which adds network latency.
Simple data structure, no scaling	Complex data structure such as Hash key support, Scaling as cluster
Data is duplicated - Since data is pulled by process when it needs to cache data, number of queries will be higher compared to central cache	Central data
Good for process specific data like system client token that once obtained can be used by the process until it expires	Great for master data which is common for many processes within a domain