Applikationsserver (3 Tier):

**F: Web API oder gRPC? Vorteile und Nachteile? Zum Beispiel: gRPC hat keinen richtigen Browser Support, was heißt das genau für uns?**

A: Pro und Contra für beide:

WebApi:

Pro:

+ Man kann es mit Postman sehr gut debuggen

+ Viel Tools: Swagger, RestSharp, etc.

+ Defacto industry standard, es wird leicht für die Kunden mit eigene Entwicklungsabteilung den Service zu benutzen

Contra:

- Langsam bei großer Datenmenge

- Kein Streaming

- Der Server kann die Daten nicht dem Client pushen, der Client muss die Daten abfragen

gRPC:

Pro:

+ Schnell bei großer Datenmenge

+ Streaming (Daten in Teilen übertragen, IAsyncEnumerable)

+ Der Server kann die Daten dem Client pushen

+ Es wird mehr und mehr verbreitet

+ Wenn richtig gemacht, man kann den Client-Code aus der Protocol-Datei generieren (wie bei WCF), auch bei Android

Contra:

* Es gibt noch kein out-of-the-box Lösung
* Es gibt kein Postman-support
* Es ist noch nur begrenzte Unterstützung von Web-Clients, es wird aber daran gearbeitet (Web-gRpc)

**F: Sicherheit, da auch public API. Verschlüsselung? Zertifikate?**

A: Beide Technologien unterstützen Verschlüsselung und verschiedene Arten von Benutzerauthentifizierung, auch mit Zertifikaten.

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/certauth?view=aspnetcore-3.1>

**F: Welcher Web Server Apache? Linux und Windows (Am besten beides, Vorzug Windows Server)?**

A: Beide können sich selbständig hosten (Kestrel server) bei jeder OS. Man kann aber auch mit einen Webserver hosten. Windows: IIS, Linux: Apache oder Nginx.

<https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/?view=aspnetcore-3.1>

**F: Performance (Auf Server Seite, aber auch Übertragung vom Client): Große Datenmengen Tests**

A: Ich habe eine Klasse mit 20\*6 = 120 Properties von verschiedenen Typen gemacht und ließ die Daten über Web und gRpc übertragen. Da der Server und der Client auf einem PC sind, die folgenden Zeiten sind nur die Dauer des Serialisierens/Deserialisierens. Bei der Netzwerkübertragung spielt auch die Datenmenge eine Rolle. Dadurch wird der Webservice noch lagsamer.

Datensätze: 10 100 1.000 100.000

/Übertragungsdauer

Web: 80ms 300ms 2.500ms 25s

gRpc: 25ms 40ms 150ms 1,3s

**F: Ausfallsicherheit? Fallback?**

A: Was ist genau damit gemeint? Wenn der Prozess von außen verwaltet wird (IIS, Apache), wird er automatisch neugestartet falls es ein Fehler passiert. Man kann auch LoadBalancer verwenden falls nötig.

**F: Ist es mögliche nur bestimmte Server Methoden für bestimmte Client/User sichtbar zu machen(zu callen)?**

A:

**Option A:** Falls es um die Edition/Tier/Lizenzbegrenz geht, es wird vielleicht Sinn machen, die Optionen während build ausschließen oder in zusätzliche Assemblies packen und nur den richtigen Kunden installieren.

**Option B:** Falls es um die Rechteverwaltung für Benutzer geht, dann kann man eigene Service/Middleware für Authorization auf der Serverseite bauen.

F: Einbindung und Verwenden SoBase.dll zum Beispiel:

Logging (NLog Abstraktion, verwenden bereits MS.Extensions.Logging)

IOC (DI) (Autofac Abstraktion): Hier prüfen wie man Autofac mit MS.Extensions.DipendencyInjection zu verbinden kann, damit wir auch weiterhin unsere Abstraktion verwenden können

A: Darüber kann ich später recherchieren, sobald ich die dll sehe. Es sieht aber so aus, als ob es überhaupt keine Sorgen machen wird. In meinem Beispiel benutze ich reine DependencyInjection von Microsoft, kann aber auch auf Autofac umstellen.

**F: Aktuelle Umsetzung (Idee):**

**Beispiel:**

**WPF Client (MVVM):**

XXX: Ab dort gleiche Assemblies auf Server (3-Tier) und Client (2-Tier).

2 Tier:

(UI) (Domain.Core) (Bl.Core) (Domain.Core) (Dal.Core)

ViewModel => XXX IAuftragsService => IOC Container => AuftragsService => IAuftragsContext => IOC Container => AuftragsContext => DB (MS Sql Server)

3 Tier:

(UI) (Domain.Core) (Wrapper auf Client?) (Wrapper auf Server?) (Domain.Core) (Bl.Core) (Domain.Core) (Dal.Core)

ViewModel => IAuftragService => IOC Container => AuftrgsServiceServerApi => JSON/Binary => Wep API Controller/ gRPC Service => XXX IAuftragsService => IOC Container => AuftragsService => IAuftragsContext => IOC Container => AuftragsContext => DB (MS Sql Server)

A: In meinem Beispiel habe ich 3-Tier und 2-Tier realisiert. Bei 2-Tier können wir alle Server-Logik direkt aufrufen ohne Netzwerkdatenübertragung. Alles hängt nur von der Client-Assembly und der Konfig-Datei ab.

**F: Die Wrapper auf Client UND Server Seite autogenerieren? Wäre das möglich bei**

1) WepApi?

2) gRPC?

Zum Beispiel:

interface IAuftragService

{

bool SaveAuftrag(AuftragsModel)

List<AuftragsModel> GetAuftragById(long id)

}

A: Soweit ist die Implementierung manuell. Wenn man den Boilerplate-Code in einer Base-Klasse versteckt, dann sieht es auf dem Client so aus:

public Task<AuftraegeAntwort> GetAuftragById(AuftragAnfrage anfrage)

=> InvokeAsync(service => service.GetAuftragById(anfrage));

So eine Wrapper-Klasse kann man auch mit .Net5 generieren lassen oder sogar jetzt schon mit T4.

**F: Kann aus diesem Interface der Client Wrapper "AuftrgsServiceServerApi" erzeugt werden?**

A: Ja, wie oben.

**F: Kann aus diesem Interface der Servert Wrapper "Wep API Controller/ gRPC Service" erzeugt werden?**

A: Theoretisch ja, wenn der BL schon woanders implementiert ist. Es hängt von der Implementation der BL ab.