**Árbol de navegación – Arquitectura**

**01. Auth (Hexagonal)**

**API (Presentation)**

- Minimal API's

- Swagger

- Middlewares

- Global Exceptions

- Fluent Validation

- Authentication

- Logging - Tracing (NLog)

- Metrics - Prometheus - Grafana\*

- RateLimiting

**Application**

- CQRS (Commands)

- Validation Behavior

**Domain**

- Entities

- Interfaces

- Resources

- Services

- Value Objects

- Aggregate Roots \*\*\*

**Infrastructure**

- Configurations

- Repositories

**02. VirtualCareCore (Hexagonal)**

**API (Presentation)**

- Minimal API's

- Swagger

- Middlewares

- Health Checks

- Global Exceptions

- Interceptors After Request (SignalR)

- Authorization

- Localization (Resources\*)

**Application**

- Modules

- Commands (CQRS)

- Queries (CQRS)

- Responses

- Validators

- Validation Behavior

- Mappings

**Domain**

- Entities

- Enums

- Helpers

- Interfaces

- Primitives

- Resources

- Services

**Shared**

**Notifications.Library**

- SignalR

**StateMachine.Library**

- Patron Saga

**Queues.Library**

- RabbitMQ

- Kafka

- AzureServiceBus

**SharedClasses**

- RequestResult

-DTOs

-**HttpFunctions \*\***

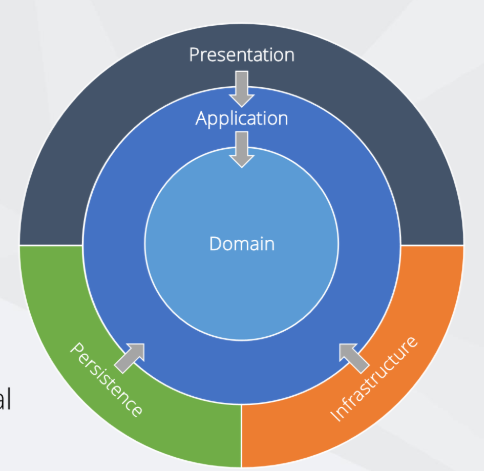
**-**HttpFunctions

1. **Logging**
2. **Tracing (Middleware – Catch Logs)**
3. **Authentication y Authorization**
4. **ApplicationInsights \*\***
5. **HealthChecks**
6. **Metrics**
7. **Warnings**
8. **CQRS**
9. **XUnit \*\***

**SignalR**

MessageOrchestator (RabbitMQ)

**Diagrama Arquitectura**



**Resumen**

* + **Auth (DDD - Hexagonal)**: Contiene la API con sus middlewares, logging, métricas, rate limiting, la capa de aplicación con CQRS y validación, la capa de dominio con entidades, interfaces, recursos, servicios, objetos de valor y raíces agregadas, y la infraestructura con configuraciones y repositorios.
  + **Notifications.Library**: Contiene SignalR.
  + **StateMachine.Library**: Contiene el patrón Saga.
  + **Queues.Library**: Contiene RabbitMQ, Kafka y AzureServiceBus.
  + **VirtualCareCore (DDD - Hexagonal)**: Contiene la API con sus middlewares, la capa de aplicación con módulos, CQRS, respuestas, validadores, comportamiento de validación y mapeos, la capa de dominio con entidades, enums, helpers, interfaces, primitivas, recursos y servicios.

Esta estructura sigue los principios de DDD (Domain-Driven Design) y la arquitectura hexagonal, organizando el proyecto en capas bien definidas y separando las preocupaciones de cada componente.