# Software Architecture: Modular Monolith Approach

lucaspaulamonti, Monday, May 5, 2025.

## Introduction

This document describes the architecture adopted in the development of the epsychology system, justifying the choice of the modular monolithic approach. The proposal is to guarantee organization, cohesion, progressive scalability and ease of maintenance, respecting the current maturity and complexity of the project.

## Justification

The decision to use a modular monolith was motivated by technical and strategic factors:

- **Controlled initial complexity**: the project is in the construction stage, with requirements still stabilizing.
- **Development speed**: less overhead in configuring multiple projects, deployments and integrations between services.
- **Ease of debugging**: everything is in a single process, which facilitates testing and corrections.
- **Organization into domains**: despite being monolithic, the system is modularized by functional areas (e.g.: Customer, Medical, User), which facilitates future separation into microservices.
- **Gradual isolation of domains into microservices**, if the system demands horizontal scalability.
- **Reuse of modules** as libraries or independent services.

## Modules

The system is divided into modules representing business domains or independent functionalities, such as:

- **Customer** — Patient and client management.
- **Medical** — Clinical documentation and medical records.
- **User** — User control and permissions.
- **Institutional** — Patient family and address data.
- **Attribute** — Auxiliary settings and data.

# Structure

Each module follows the same internal structure:

- Source/
- Source/Module/
- Source/Module/Customer/
- Source/Module/Customer/Data/
- Source/Module/Customer/Data/PostgreSQL 17/
- Source/Module/Customer/Data/FirebirdSQL 5/
- Source/Module/Customer/Form/
- Source/Module/Customer/Form/Delphi 12/
- Source/Module/Customer/Form/Delphi 12/VCL/
- Source/Module/Customer/Test/
- Source/Module/Customer/Util/

# Tools

The following tools are being used:

- **Language**: Delphi 12.
- **Platform**: VCL for Windows Desktop applications.
- **Databases**: PostgreSQL 17, FirebirdSQL 5, MongoDB 8.
- **Code Organization**: separation between data layers (Data), interface (Form) and utilities (Util).
- **Server**: Ubuntu Server 24 LTS.

# Versioning

The versioning features that are being used are:

- **Use of a single main .dproj** to generate the current executable.
- **Clean structure with custom .gitignore** to ignore temporary IDE files (__history, .identcache, .dproj.local, etc.).
- **Each module can eventually have its own .dproj**, allowing the generation of independent executables/microservices.

# Conclusion

The modular monolith architecture offers the ideal balance between initial simplicity and growth potential. By combining modularization best practices, the system remains clean, organized, and future-proof — without incurring the early complexity of microservices.