ADR 004: Use the Wrapper pattern

Farmacy Food interacts with many third-party partner systems. The interfaces of these external systems are bound to differ greatly. Such variety emerges both from a standards and a technology point of view.

Non-standardized contracts are a challenge. Direct linking of internal (Farmacy Food) business services to third-party services hinders portability, and creates undesirable implementation coupling. If details of the third-party APIs permeate Farmacy Food business logic services, changes on partner systems APIs can negatively affect Farmacy Food most fundamental services.

Decision

We will use the Wrapper pattern to encapsulate external services into standardized interfaces.

The standardized contracts exposed by wrapper services should adhere to Farmacy Food internal design standards and practices, while reducing or eliminating external influences as much as possible.

We will design our internal logic so that they interact with partner systems only through wrappers.

Rationale

We expect there will be many vendor-specific systems our platform will have to interact with. For example, there can be multiple suppliers of smart fridges, multiple vendor POS systems, multiple payment gateways. Adding or removing a third-party partner should be simple. Our internal systems should be affected the least possible by changes in vendors, their technology, or by the details of their specific tools.

Status

Proposed

Consequences

- Farmacy Food will have a greater degree of freedom to evolve independently of partner systems.
- Every external service should have a corresponding wrapper service.
- The wrapper services will translate the external contracts to meet our internal needs and standards.
- The additional processing may incur in some performance overhead, though generally not significant.