Implementación para arquitecturas limpias

Dirección General de Sistemas de Información y Comunicaciones

Áreas de Gobierno Tecnológico de SSII y del Dato

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Versión

Guía de implementación Vigente

Versión: **1.0.1**

Estado: ACTIVO

Entrada en vigor desde: 22 may 2024

Obligado cumplimiento desde: N/A

Guía de implementación PRE-RELEASE

Versión: -

Estado: -

Entrada en vigor desde: -

Obligado cumplimiento desde: -

Tabla de Contenido

Cumplimiento normativo

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

Contacto Arquitectura: l-arquitectura.stic@juntadeandalucia.es

Histórico de cambios

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabeceras de la tablas dónde se indican las fechas de entrada en vigor y versión.

Versión	Pre-release	Adopción	Activa	Retiro	Alcance
v01r00	-	22 may 2024	5 ago 2020	-	• Versión inicial. ○ Pautas Clean Architecture

1.2. Ciclo de vida de la arquitectura de referencia

Los cambios normativos dentro de la arquitectura de referencia seguirán el siguiente ciclo de vida:

- **Pre-release:** Periodo durante el cual la normativa se hace publica aunque no es necesario adherirse inmediatamente. En este periodo aún puede sufrir pequeñas correcciones.
- **Adopción :** Periodo durante el cual la normativa inicia un periodo de tiempo flexible para su aplicación de forma obligatoria al ámbito en el cual está destinado.
- **Activa:** Periodo durante el cual es obligatorio su cumplimiento para el

1. Introducción

Esta guía pretende impulsar los diseños de las aplicaciones que están basadas en Arquitecturas Limpias y Domain Centric que permitirán un diseño orientado al negocio y un desarrollo que facilite el cambio.

Estas guías tratan de ser agnósticas al lenguaje de desarrollo

1.1. Ámbito de aplicación

Ámbito de aplicación obligatorio (*)

Esta arquitectura de referencia será de aplicación obligatoria(*) en los siguientes casos:

- Desarrollo de nuevos sistemas de información corporativos.
- Refactorización de sistemas de información existentes.
- Evolución de sistemas de información que requieran la implementación de nuevos procesos o funcionalidades y que requiera de desarrollo en funciones "core" del sistema.

Se recomienda su aplicación:

- Sistemas comerciales que permitan una arquitectura tecnológica flexible y alineada con este modelo de referencia.
- Desarrollos locales con previsión de escalar a sistemas corporativos.
- Cualquier desarrollo para implementar nuevos procesos sobre los sistemas de información existentes.

Se recomienda analizar en detalle con la unidad de Arquitectura de la STIC y particularizarla para los siguientes casos:

- Sistemas analíticos o que gestionen grandes volúmenes de datos (**).

ámbito en el cual está destinado.

- **Retiro:** Periodo durante el cual dejará de aplicarse en el ámbito al cual está destinado en sustitución de nuevas versiones o normas.

(*) Cualquier propuesta que difiera de esta arquitectura deberá ser aprobada por el Área de Gobernanza de la STIC, previa solicitud y justificación en su caso.

(**) En proceso de elaboración de una arquitectura de referencia para sistemas analíticos y big data.

2 Introducción

Clean Architecture, es una arquitectura de software que enfatiza la separación de responsabilidades y el desacoplamiento de componentes. Su objetivo principal es crear sistemas que sean fáciles de mantener, testear y evolucionar. Con estas características se busca crear sistemas modulares, altamente mantenibles y adaptables al cambio. Al definir claras separaciones de responsabilidades y establecer flujos de dependencia unidireccionales, se promueve un diseño de software robusto y flexible que puede evolucionar con los requisitos del negocio y la tecnología.

Principales Características de la Clean Architecture

1. **Independencia de la Framework**: - La lógica del negocio y las reglas de la aplicación no deben depender de ninguna tecnología específica de framework. Esto facilita cambiar de frameworks sin afectar la lógica del negocio.
2. **Testabilidad**: - Al mantener la lógica del negocio independiente de los detalles de implementación, se facilita la creación de pruebas unitarias y de integración. Los componentes pueden ser probados en aislamiento.
3. **Independencia de la Interfaz de Usuario**: - Las reglas del negocio no dependen de la interfaz de usuario, lo que permite cambiar de tecnologías de interfaz de usuario (por ejemplo, de web a móvil) sin cambiar la lógica del negocio.
4. **Independencia de la Base de Datos**: - La lógica del negocio no debe depender de los detalles de la base de datos. Esto permite cambiar de sistemas de base de datos sin afectar la lógica del negocio.
5. **Independencia de Agencias Externas**: - La arquitectura debe ser independiente de bibliotecas externas. Si una biblioteca debe cambiar, solo debe afectar una parte mínima del sistema.

Componentes y Capas

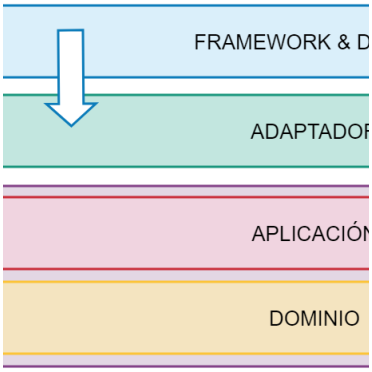
1. **Entidades de dominio**: - Contienen las reglas del negocio más generales y de más alto nivel. Estas reglas de negocio pueden ser compartidas por diferentes aplicaciones en el mismo dominio.
2. **Casos de Uso**: - Contienen la lógica de aplicación específica para un caso de uso particular. Orquestan el flujo de datos hacia y desde las entidades y dirigen a los actores externos (interfaces de usuario, dispositivos, etc.).
3. **Interfaces de Controladores y Presenters**: - Interfaces que definen cómo se debe interactuar con los casos de uso y cómo presentar los resultados. Estas interfaces ayudan a desacoplar las implementaciones concretas de los controladores y presenters de los casos de uso.
4. **Adaptadores (Interfaces de usuario, Gateways, Presenters, etc.)**: - Implementaciones concretas de los controladores, presenters y gateways. Aquí es donde los detalles específicos de la tecnología (como frameworks web, mensajería, bases de datos, etc.) se manejan.
5. **Infraestructura**: - Contiene detalles técnicos concretos como la persistencia de datos (repositorios concretos), la configuración del framework, y otros aspectos técnicos necesarios para que la aplicación funcione.

Flujo de Dependencias

- Las dependencias deben fluir desde los niveles más externos hacia los más internos.
- Las capas más externas (adaptadores e infraestructura) pueden depender de las capas más internas (casos de uso y entidades), pero no al revés.
- Se utilizan interfaces para definir contratos en las capas internas, permitiendo que las capas externas proporcionen implementaciones concretas sin que las capas internas sean conscientes de estos detalles.

3 Arquitecturas Limpias dentro del SAS

quitectura l Arquitectura Limpias

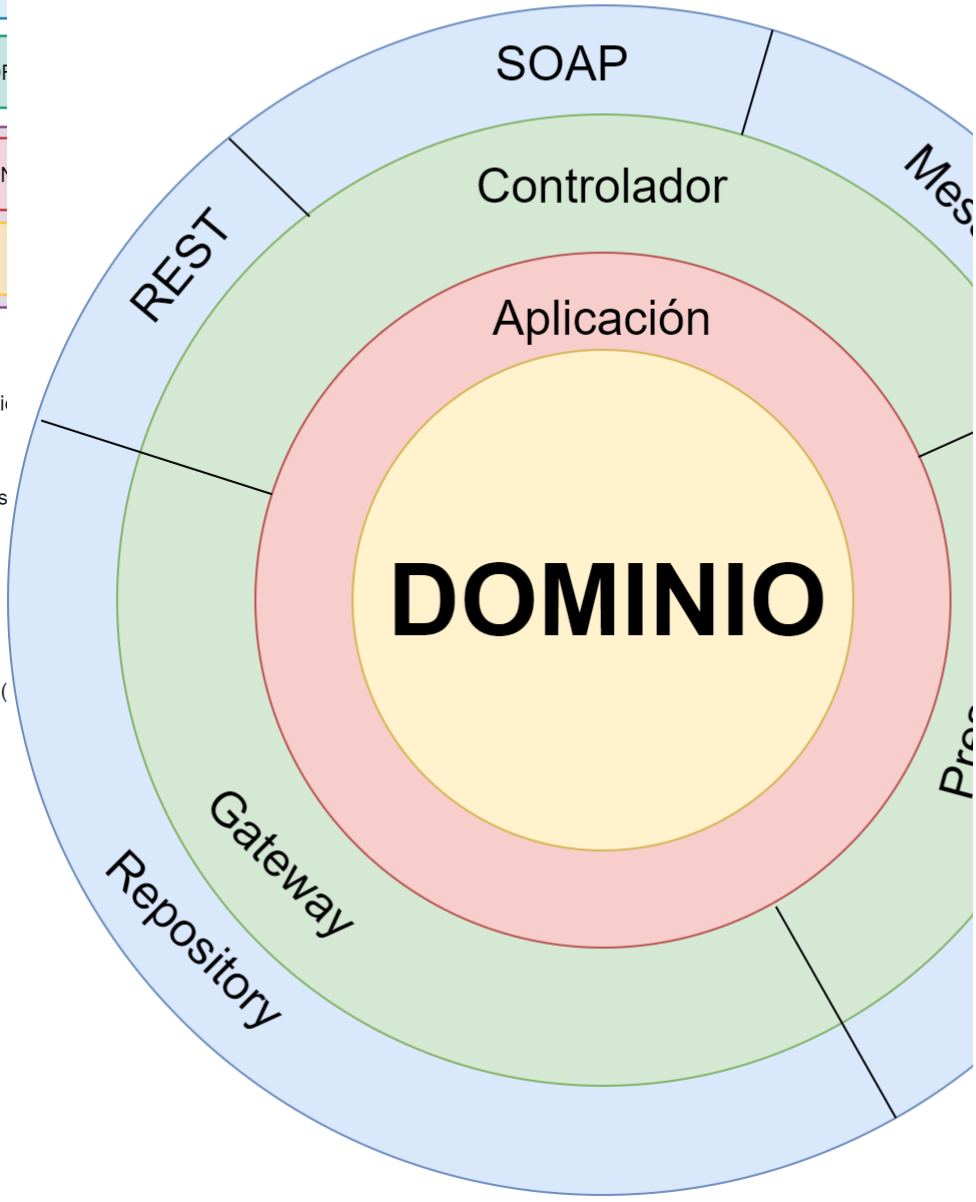


a la definición del dominio y sea agnósti

ontratos (Repositorio, Adaptadores, Cas

uso

ıdor (Rest, SOAP, Message) Presenter (



Pauta	Descripción	Regla de verificación

P1	**Dependencias Top-Down.** Como se indica en el gráfico las dependencias deben seguir un flujo top-down de tal forma que las capas superiores deben tener dependencia con la inmediatemente inferior y con el dominio.	Revisando las dependencias del gestor de paquete (Maven, Nuget, npm)
P2	**Independencía de tecnología**. Las capas del kernel (Dominio y aplicación) deben ser desarrolladas sin ningún conocimiento tecnológico que no sea inherente al dominio.	Revisando las dependencias del gestor de paquete (Maven, Nuget, npm)
P3	**Independencía de Frameworks**. Se debe evitar el uso de frameworks en capas interiores de la arquitectura, permitiendo un substitución sin efectos colaterales	Revisando las dependencias del gestor de paquete (Maven, Nuget, npm)
P4	**Testable.** las reglas de negocio son fácilmente testables sin utilizar la interfaz de usuario, base de datos, servidor web	Revisando las dependencias del gestor de paquete (Maven, Nuget, npm). Existencia de test unitarios que no necesiten configuración tecnológica
P5	**Independiente de la interfaz de usuario**. La interfaz de usuario es fácilmente sustituible o las nuevas versiones del framework de UI que usas tienen un impacto mínimo.	En thin server, la UI debe ser un proyecto a parte siguiendo la normativa front
P6	**La estructura del microservicio** debe estar basada en el arquetipo proporcionado desde la STIC	La estructura de carpetas y módulos corresponde con la misma que se define en el arquetipo

Referencias

Referencias a fuents de información externas

En esta sección se presentará un listado de bibliografia o documentos externos que sean de ayuda para la compresión de la normativa expuesta. Si no se expone información en esta sección eliminese.

Esta nota informativa debe ser eliminada en el documento normativo definitivo.