TOGAF® 10 en Práctica

De los fundamentos a la aplicación en proyectos reales

José María Flores Zazo

***A mi mujer y a mi hijo**,*

por acompañarme incluso cuando hablo más con el ordenador que con vosotros.

Gracias por estar siempre ahí... y por no esconderme el cable de corriente.

Prefacio

La arquitectura empresarial es una disciplina fundamental para las organizaciones que buscan prosperar en un entorno cada vez más complejo, competitivo y dinámico. Sin embargo, su implementación puede parecer intimidante debido a la amplitud de conceptos, metodologías y herramientas involucradas. Este libro, titulado "Descubriendo TOGAF: conceptos clave de la arquitectura empresarial", tiene como objetivo desmitificar el marco TOGAF®, presentándolo de una manera estructurada, accesible y práctica.

Basado en la versión 10 de TOGAF, este libro es un recurso diseñado para arquitectos, líderes empresariales y cualquier profesional interesado en comprender cómo las decisiones estratégicas, tecnológicas y operativas pueden ser alineadas de manera efectiva. Para facilitar esta comprensión, el contenido se divide en tres partes claramente diferenciadas, cada una abordando un aspecto crucial del marco TOGAF:

Parte I: Fundamentos de TOGAF

En esta sección se establece la base conceptual que sustenta TOGAF, explorando sus principios, objetivos y beneficios. Aquí explicamos qué significa realmente "arquitectura empresarial", por qué TOGAF es un estándar abierto, y cómo puede ser adaptado y utilizado en organizaciones de cualquier tipo y tamaño. Abordamos conceptos esenciales como:

- Los dominios arquitectónicos: negocio, datos, aplicaciones y tecnología.
- Los Building Blocks: componentes reutilizables para construir arquitecturas.
- El valor de TOGAF como marco flexible y adaptable.

Esta primera parte está diseñada para que el lector se familiarice con el por qué y qué de TOGAF, proporcionando un lenguaje común que conecta a negocio y TI. Por ejemplo, se explica cómo una empresa de tecnología puede utilizar TOGAF para integrar su ecosistema de inteligencia artificial y machine learning con sus objetivos estratégicos, facilitando la creación de productos más personalizados y eficientes.

Parte II: Architecture Content Framework (ACF)

El Architecture Content Framework (ACF) es la estructura interna que organiza todo el conocimiento generado durante la implementación de TOGAF. En esta sección, el lector aprenderá cómo documentar, modelar y gestionar el contenido arquitectónico de manera coherente y reutilizable.

Se profundiza en temas como:

- Los Deliverables y artefactos generados en cada fase del ADM.
- Los Building Blocks: desde los Architecture Building Blocks (ABB) hasta los Solution Building Blocks (SBB).
- La organización por dominios y la trazabilidad entre elementos arquitectónicos.

Por ejemplo, se explica cómo modelar la arquitectura de una plataforma de comercio electrónico, mostrando cómo las aplicaciones (carrito de compras, pasarela de pago, recomendaciones de

producto) interactúan con los flujos de datos (transacciones, perfiles de usuario) y cómo estas capas se soportan en una infraestructura tecnológica basada en servicios en la nube.

Parte III: Architecture Development Method (ADM)

La tercera parte del libro es el núcleo práctico, donde se desglosa el Architecture Development Method (ADM), el corazón operativo de TOGAF. Cada fase del ADM se analiza en detalle, desde la Fase Preliminar, donde se establece la capacidad organizativa para la arquitectura, hasta la Fase H, donde se gestionan los cambios para mantener la relevancia arquitectónica.

Cada capítulo de esta sección incluye:

- Objetivos claros de cada fase.
- Actividades clave y herramientas útiles.
- Ejemplos prácticos para ilustrar cómo aplicar cada fase en contextos reales.Un enfoque práctico y accesible

En esta parte, los ejemplos reflejan situaciones actuales y relevantes para las organizaciones modernas:

- Fase A (Architecture Vision): Se muestra cómo una empresa de servicios de streaming puede definir una arquitectura que permita la personalización de contenido basada en inteligencia artificial. Esto incluye integrar sistemas de recomendación, análisis de hábitos de consumo y optimización del ancho de banda para brindar una experiencia más fluida y atractiva al usuario.
- Fase B (Business Architecture): Se analiza cómo una startup de movilidad urbana puede redefinir su modelo de negocio para integrar servicios de micromovilidad (bicicletas y scooters eléctricos) con transporte público tradicional. Aquí se describen las capacidades de negocio que deben fortalecerse, como la gestión de usuarios, la integración de pagos y la optimización de rutas.
- Fase C (Information Systems Architecture): Se explora cómo una empresa de retail puede diseñar una arquitectura de datos y aplicaciones para implementar una estrategia omnicanal. Esto incluye construir un único repositorio centralizado para datos de clientes, inventarios y transacciones, mientras se interconectan aplicaciones móviles, portales web y sistemas de punto de venta físico.
- Fase D (Technology Architecture): Se muestra cómo una plataforma de entregas rápidas puede migrar su infraestructura tecnológica a un modelo híbrido basado en servicios en la nube y edge computing. Esto incluye la adopción de contenedores, balanceadores de carga y redes de baja latencia para mejorar la escalabilidad y el rendimiento en tiempo real.
- Fase E (Opportunities and Solutions): Un ejemplo práctico sería cómo una fintech puede priorizar y planificar soluciones para lanzar un nuevo servicio de banca digital. Esto incluye identificar brechas en su arquitectura actual, como la necesidad de autenticación biométrica, y desarrollar un roadmap claro para implementar estas capacidades en fases.
- Fase F (Migration Planning): Se analiza cómo una empresa de logística puede planificar la transición desde sus sistemas heredados hacia una nueva plataforma basada en microservicios. Aquí se detalla cómo dividir el proceso en etapas manejables, minimizando el riesgo y garantizando la continuidad del negocio.

- Fase G (Implementation Governance): Se aborda cómo una empresa de salud digital puede establecer mecanismos de supervisión para garantizar que los estándares de seguridad y privacidad se respeten durante la implementación de una plataforma de telemedicina, incluyendo la gestión de excepciones y auditorías regulares.
- Fase H (Architecture Change Management): Se ejemplifica cómo una empresa de redes sociales puede gestionar los cambios arquitectónicos necesarios para integrar nuevas funcionalidades, como salas virtuales para eventos en el metaverso. Esto incluye evaluar el impacto en los sistemas actuales, adaptar los building blocks existentes y garantizar que las actualizaciones no comprometan la experiencia del usuario.

Un ejercicio de investigación y recopilación

Toda la información presentada en este libro está basada en documentos, guías y recursos oficiales de TOGAF, así como en experiencias prácticas y ejemplos recopilados en mi trayectoria profesional. TOGAF es un marco abierto, y gran parte de su contenido es público y accesible para todos. Mi intención al escribir este libro ha sido recopilar, estructurar y redactar esta información en español, con un enfoque claro y práctico que facilite su comprensión y aplicación.

Este proyecto comenzó como un ejercicio personal de estudio, en el que busqué entender a fondo los conceptos y herramientas de TOGAF para aplicarlos en mi práctica profesional. Sin embargo, pronto me di cuenta de que esta información podría ser útil para otros profesionales que, como yo, buscan dominar esta disciplina. Por ello, decidí compartir esta recopilación y análisis con los lectores, con la esperanza de que les sirva como una guía completa y accesible para descubrir y aplicar TOGAF.

Un enfoque práctico y accesible

Aunque este libro se basa en los fundamentos teóricos de TOGAF, he tratado de mantener un enfoque práctico en todo momento. Cada concepto va acompañado de ejemplos, consejos y herramientas que facilitan su aplicación en el día a día. Mi intención es que este libro no sea solo una referencia para entender TOGAF, sino también un recurso práctico que pueda ser utilizado en proyectos reales.

Los ejemplos incluidos han sido seleccionados para reflejar casos actuales y de relevancia en la era digital. Desde la adopción de tecnologías emergentes como la inteligencia artificial y los servicios en la nube, hasta los retos de integrar sistemas heredados con plataformas modernas, cada ejemplo tiene como objetivo mostrar cómo TOGAF puede ser aplicado en los desafíos tecnológicos y estratégicos más comunes de las organizaciones hoy en día.

Quiero recalcar que este libro no pretende reemplazar las guías oficiales de TOGAF ni otros recursos especializados. Más bien, busca ser un complemento que presente los conceptos clave de manera más accesible y contextualizada. Este es un ejercicio de recopilación, síntesis y estudio personal, pensado para ayudar a otros que se encuentran en el camino de aprender y aplicar TOGAF.

Una llamada a la acción

La arquitectura empresarial puede parecer un campo técnico y complejo, pero en realidad, su esencia radica en conectar estratégicamente los objetivos de negocio con las soluciones tecnológicas. Este libro es una invitación a explorar esa conexión, a aprender los principios fundamentales de TOGAF y a aplicarlos de manera concreta y estratégica en las organizaciones.

A los lectores que se embarquen en este viaje, les invito a personalizar los conceptos y adaptarlos a sus propias necesidades y contextos. TOGAF no es un marco rígido, sino una herramienta flexible que puede ajustarse a las particularidades de cada organización, sector e industria. Espero que este libro sea un punto de partida valioso en su recorrido hacia la excelencia en arquitectura empresarial.

Gracias por permitirme compartir este trabajo con ustedes. Espero que disfruten de la lectura y encuentren en estas páginas un recurso útil y práctico para su desarrollo profesional.

Sobre el autor

Soy José María Flores Zazo y tengo más de 28 años de experiencia en el desarrollo, análisis, diseño y entrega de aplicaciones en diversas tecnologías, desde aplicaciones de escritorio y web hasta móviles nativas y multiplataforma. Siempre he buscado crear soluciones funcionales, eficientes y que realmente aporten valor a las personas.

A lo largo de mi carrera, me he enfocado en tres pilares fundamentales: investigación, formación y agilidad, priorizando siempre la calidad, la reusabilidad y la legibilidad del software. He participado en todas las fases de los proyectos, desde la pre-venta y el diseño inicial hasta la implementación, además de contribuir en áreas clave como la creación de prototipos, frameworks internos y la investigación en innovación tecnológica.

Actualmente, desempeño roles como arquitecto de soluciones, desarrollador, evangelista de tecnologías y auditor de proyectos, ayudando a las empresas a implementar mejores y a llevar sus ideas desde la visión hasta la ejecución con éxito. Algunos de los proyectos que he liderado, como LaundryID y NeverAlone, han recibido reconocimiento nacional e internacional. También he sido distinguido como Microsoft MVP en Azure y Developer Technologies, un logro que refleja mi compromiso con la comunidad tecnológica.

Soy un apasionado del aprendizaje continuo y de compartir conocimiento. Participo activamente como mentor, speaker en eventos, creador de contenido y organizador de workshops gratuitos. Mi objetivo es ayudar a otros a crecer, ya sea a través de un consejo, una charla o colaborando en proyectos.

Este libro es una extensión de mi propósito de contribuir y compartir. Espero que os sea útil y que, de alguna manera, os ayude en vuestro desarrollo profesional. Podéis encontrarme en https://jmfloreszazo.com/, donde siempre estoy dispuesto a intercambiar ideas y seguir aprendiendo juntos.

CONTENIDO

> "La victoria se consigue antes de librar la batalla, cuando se ha hecho una planificación perfecta."
>
> — Sun Tzu, The Art of War

Notas:

 Punto en el que debes prestar especial atención.

 Consejos basados en la experiencia o notas del autor.

Exención de responsabilidad:

- Las marcas registradas mencionadas en este documento son propiedad de sus respectivos propietarios.
- El autor se exonera de cualquier responsabilidad por el uso de las marcas registradas mencionadas en este documento.
- El lector asume toda responsabilidad por el uso y aplicación del contenido del libro.
- El autor no asume ninguna responsabilidad por los posibles perjuicios derivados del uso del contenido del libro.

1 INTRODUCCIÓN

Hablar de arquitectura empresarial en el siglo XXI es hablar de adaptación, propósito y sostenibilidad organizativa. En un mundo donde las transformaciones digitales ya no son opcionales, sino estructurales, y donde las organizaciones se enfrentan a entornos volátiles, inciertos, complejos y ambiguos, surge la necesidad urgente de disponer de marcos que proporcionen coherencia y dirección. En este contexto, TOGAF® se consolida como un referente global.

TOGAF® —The Open Group Architecture Framework— no es simplemente una metodología; es un marco abierto y probado que permite a las organizaciones diseñar, implementar y evolucionar su arquitectura de forma estructurada y flexible. Desde su origen en la década de los noventa, derivado del TAFIM del Departamento de Defensa de EE. UU., hasta su evolución en la versión 10 actual, TOGAF ha sido construido de manera colaborativa por cientos de profesionales y organizaciones, adoptando prácticas reales, necesidades cambiantes y nuevas formas de pensar la arquitectura.

A diferencia de enfoques cerrados o propietarios, TOGAF ofrece una base común que puede adaptarse al contexto de cada empresa. No impone una herramienta, ni una secuencia obligatoria de pasos. Proporciona una caja de herramientas estructurada, con métodos, principios, entregables y buenas prácticas que cada organización puede adoptar, adaptar o ampliar.

1.1 ¿POR QUÉ UN MARCO COMO TOGAF?

Las organizaciones modernas no solo gestionan sistemas de información, gestionan relaciones complejas entre estrategia, procesos, tecnologías emergentes, modelos operativos y regulaciones cambiantes. En ese entramado, la arquitectura empresarial se convierte en una brújula: ayuda a comprender el presente, definir el futuro deseado y construir el puente entre ambos.

TOGAF facilita este camino mediante una estructura modular que abarca:
- El ciclo de desarrollo de arquitectura (ADM)
- Los artefactos y building blocks que la componen
- Los catálogos, matrices y diagramas necesarios para comunicar y gobernar decisiones
- Un modelo de servicios arquitectónicos internos
- Guías y técnicas prácticas adaptables a distintas industrias, modelos y marcos complementarios

A diferencia de otros marcos de referencia como Zachman (más orientado a clasificar información), IT4IT (centrado en la cadena de valor de IT) o SAFe (enfocado a la agilidad a escala), TOGAF proporciona una estructura metodológica completa para diseñar y transformar la arquitectura empresarial en cualquier sector o tamaño de organización. Su modularidad actual permite incluso integrarse con estos marcos, no sustituirlos, lo que lo hace especialmente versátil.

El valor de este enfoque reside en su capacidad para orquestar múltiples visiones (negocio, TI, datos, operaciones) bajo un lenguaje común, promoviendo la colaboración, la trazabilidad y la toma de decisiones basadas en evidencias y modelos.

1.2 ¿PARA QUIÉN ES TOGAF?

TOGAF no es exclusivo de arquitectos de TI. Su adopción resulta especialmente valiosa para:
- Líderes tecnológicos y estratégicos (CIO, CTO, Enterprise Architects)
- Equipos de transformación digital
- Consultores que articulan entre negocio y tecnología
- Arquitectos de soluciones, software, datos o seguridad
- Analistas y gestores de cambio en entornos complejos
- Responsables de programas con múltiples dependencias y objetivos a largo plazo

Más allá de la certificación, TOGAF promueve un modo de pensar arquitectónico: entender las organizaciones como sistemas integrados, donde cada decisión tiene implicaciones estructurales, y donde cada componente debe alinearse con un propósito superior.

1.3 TOGAF 10: EVOLUCIÓN Y MODULARIDAD

La décima edición representa un salto cualitativo. Por primera vez, TOGAF se presenta como una colección modular de partes reutilizables. Las organizaciones pueden adoptar únicamente lo que necesitan, integrarlo con sus marcos existentes (SAFe, ITIL, COBIT, ArchiMate, DevOps…) y evolucionarlo conforme aumente su madurez.

Además, se potencia la idea de "servicios de arquitectura" como un modelo de entrega profesional, medible y escalable. El estándar ya no se limita a describir qué artefactos generar, sino cómo entregar valor continuo como práctica interna transversal.

Una guía para arquitectos en la práctica

Este libro nace con un enfoque práctico. Más allá de reproducir literalmente los capítulos del estándar, busca traducirlos a un lenguaje accesible, aplicarlos a escenarios reales y ayudar al lector a tomar decisiones informadas sobre cómo implementar y adaptar TOGAF en su organización.

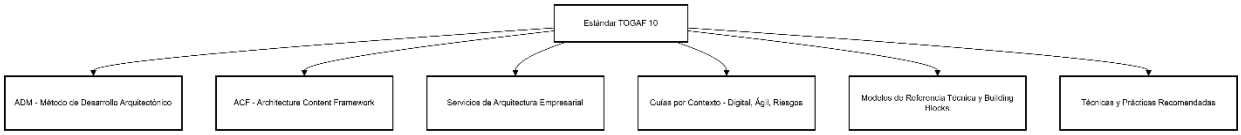
A lo largo de los capítulos, exploraremos los fundamentos, el contenido arquitectónico, el método ADM, las recomendaciones prácticas, y responderemos a las preguntas frecuentes de quienes enfrentan el reto de modernizar sus organizaciones sin perder el control en el proceso.

TOGAF no es una fórmula mágica, pero es una de las mejores brújulas disponibles para navegar en la complejidad organizativa con criterio, disciplina y foco en el valor.

Esta capacidad de integrarse con otras metodologías y marcos —sin necesidad de reemplazarlos— convierte a TOGAF 10 en una herramienta estratégica para ecosistemas complejos y entornos regulados.

1.4 TOGAF 10: UN ESTÁNDAR MODULAR Y COMPONIBLE

La estructura modular de TOGAF 10 permite a las organizaciones adoptar sólo las partes que necesitan. Su núcleo incluye el método ADM, pero se complementa con guías específicas por dominio, contenido estructurado (ACF), principios, técnicas y capacidades arquitectónicas. Este enfoque componible permite una implementación incremental según el contexto y madurez organizativa.



1.5 CONECTANDO CON EL CONTENIDO ARQUITECTÓNICO

Tras conocer la filosofía y estructura general de TOGAF 10, el siguiente paso es sumergirse en uno de sus pilares fundamentales: el Architecture Content Framework (ACF). Este marco establece el lenguaje documental común que nos permitirá definir, organizar y reutilizar los entregables arquitectónicos a lo largo del método ADM, que veremos en capítulos posteriores.

2 ARCHITECTURE CONTENT FRAMEWORK (ACF)

2.1 ¿QUÉ ES EL ARCHITECTURE CONTENT FRAMEWORK (ACF)?

El *Architecture Content Framework* (ACF) es uno de los pilares fundamentales del estándar TOGAF® 10. Su propósito es establecer un marco estructurado para describir, categorizar y relacionar los artefactos que se producen durante el desarrollo y mantenimiento de una arquitectura empresarial. En términos sencillos, el ACF define **qué se entrega** en cada fase del proceso arquitectónico, **cómo se organiza**, y **cómo se relaciona con el resto de los componentes del framework**.

A medida que las organizaciones crecen, los equipos multidisciplinares generan múltiples modelos, diagramas, listas y documentos estratégicos o técnicos. Sin una taxonomía clara, estos entregables pueden volverse inconsistentes, duplicados o incomprensibles fuera de su contexto inmediato. Aquí es donde el ACF aporta orden, coherencia y trazabilidad.

2.1.1 Objetivos del ACF

- Establecer una **estructura común** para todos los artefactos arquitectónicos.
- Proporcionar una **nomenclatura estándar y reutilizable** que facilite la colaboración entre equipos.
- Asegurar la **trazabilidad de las decisiones** arquitectónicas mediante una organización clara de entregables.
- Favorecer la **reutilización de componentes** ya modelados (por ejemplo, building blocks comunes).
- Mejorar la **gobernanza y comunicación** de la arquitectura a través de catálogos, matrices y diagramas alineados.

2.1.2 ¿Por qué es importante el ACF?

A menudo, los equipos arquitectónicos trabajan en silos: algunos modelan procesos, otros sistemas, otros capacidades tecnológicas o requisitos de seguridad. El ACF proporciona una **estructura transversal**, un lenguaje común que permite integrar todas esas visiones parciales en una **visión arquitectónica completa y coherente**. Es especialmente valioso en proyectos con múltiples partes interesadas, donde las decisiones deben ser trazables y auditables.

2.1.3 Relación con el método ADM

El ACF no existe de forma aislada: está íntimamente ligado al *Architecture Development Method* (ADM), el método central de TOGAF para desarrollar arquitecturas.

Exploraremos el ADM en profundidad más adelante. Por ahora, basta con entender que es el proceso metodológico que guía la creación de arquitecturas a lo largo de diferentes fases. Aquí nos centraremos solo en los productos que genera y cómo se organizan dentro del ACF.

Cada fase del ADM produce ciertos entregables: catálogos de aplicaciones, matrices de relaciones, diagramas de flujos o building blocks reutilizables. El ACF sirve como el contenedor que **clasifica, organiza y estandariza** todos estos productos.

Por ejemplo:

- En la Fase B (Arquitectura de Negocio), se pueden generar un catálogo de capacidades, una matriz de roles y procesos, y un diagrama de actividades clave.
- En la Fase C (Arquitectura de Sistemas de Información), se incluirán catálogos de aplicaciones y datos, matrices de interfaces y diagramas lógicos.

Estos artefactos **se documentan y almacenan** en un repositorio estructurado según el ACF, lo que facilita su gestión, versionado y posterior reutilización.

2.1.4 Estructura básica del ACF

El Architecture Content Framework define tres tipos principales de elementos:

1. **Entregables (*Deliverables*)**
   Son paquetes documentales formales, que se presentan a partes interesadas clave y que suelen marcar hitos del proyecto. Por ejemplo: una Evaluación de Arquitectura, un Documento de Requisitos, o un Plan de Transición.
2. **Artefactos (*Artifacts*)**
   Son modelos, gráficos o documentación individual que describe un aspecto concreto de la arquitectura. Pueden ser incluidos en uno o varios entregables. Por ejemplo: un Diagrama de Infraestructura, una Matriz de Servicios, un Catálogo de Objetos de Datos.
3. **Bloques de Construcción (*Building Blocks*)**
   Representan unidades lógicas (de negocio, aplicación, datos o tecnología) que pueden ser reutilizadas en múltiples arquitecturas. Ejemplos: un Servicio de Autenticación, un Proceso de Atención al Cliente, un Motor de Reglas de Negocio.

Este modelo tripartito proporciona una **visión modular y escalable**, compatible tanto con arquitecturas corporativas de gran escala como con proyectos ágiles o departamentales.

2.2 Componentes del ACF

El *Architecture Content Framework* (ACF) de TOGAF se articula en torno a tres componentes fundamentales: **entregables**, **artefactos** y **building blocks**. Estos elementos permiten estructurar de forma lógica y trazable todo el contenido arquitectónico que se genera a lo largo del ciclo de vida de una arquitectura empresarial.

Su correcta distinción y uso asegura una práctica de arquitectura eficaz, repetible y orientada al valor, facilitando la comunicación entre arquitectos, stakeholders y equipos técnicos.

2.2.1 Entregables (Deliverables)

Los **entregables** son el producto final de una fase del proceso arquitectónico o de una actividad concreta. Se trata de **documentos formales**, empaquetados, que han sido validados, revisados y aceptados por los actores relevantes (por ejemplo, un comité de arquitectura, dirección de IT, responsables de negocio, etc.).

Un entregable puede contener múltiples artefactos que lo sustentan o detallan. No obstante, lo importante es que representa un **hito documental**, muchas veces asociado a decisiones estratégicas, aprobaciones formales o avances clave del proyecto.

Ejemplos comunes de entregables:

- Evaluación del estado actual (baseline architecture)
- Documento de visión arquitectónica
- Documento de requisitos de negocio
- Modelo de referencia tecnológico
- Plan de migración y hoja de ruta
- Plan de transición y proyectos

Un entregable puede tener naturaleza ejecutiva o técnica, pero en ambos casos representa un acuerdo compartido entre arquitectura y el resto de la organización.

2.2.2 Artefactos (Artifacts)

Los **artefactos** son las piezas individuales de contenido arquitectónico que componen los entregables. Pueden ser modelos, documentos, listas, diagramas o cualquier representación que describa un aspecto concreto de la arquitectura empresarial.

Son el corazón documental del trabajo arquitectónico: permiten representar procesos, capacidades, relaciones, flujos de datos, componentes tecnológicos, dependencias lógicas y mucho más.

TOGAF agrupa los artefactos en tres categorías principales:

- **Catálogos**: Listas estructuradas de elementos (por ejemplo, capacidades de negocio, aplicaciones, objetos de datos).
- **Matrices**: Representaciones bidimensionales que relacionan elementos entre sí (por ejemplo, aplicaciones por proceso, roles por función).
- **Diagramas**: Modelos visuales que explican de forma gráfica relaciones, flujos o estructuras (por ejemplo, un diagrama de capas o un mapa de integración).

Ejemplos típicos de artefactos:

- Catálogo de funciones de negocio
- Matriz de asignación de actores

- Diagrama de arquitectura lógica
- Modelo de interfaces entre aplicaciones
- Matriz CRUD de datos vs. procesos

Los artefactos tienen valor tanto por sí solos como en conjunto. Son las piezas que dan vida al rompecabezas arquitectónico.

2.2.3 Building Blocks

Los **building blocks**, o bloques de construcción, representan unidades modulares y reutilizables que describen capacidades o elementos dentro de una arquitectura. Se consideran **componentes fundamentales**, que pueden combinarse, ensamblarse y reaprovecharse a lo largo de múltiples proyectos o dominios arquitectónicos.

TOGAF distingue dos tipos principales:

- **Architecture Building Blocks (ABB)**: Representan capacidades conceptuales o lógicas, aún sin definir su implementación exacta. Son ideales en fases iniciales de definición.
- **Solution Building Blocks (SBB)**: Representan soluciones concretas y reales que implementan los ABB. Incluyen ya tecnología, producto, proveedor, etc.

Los building blocks permiten estructurar la arquitectura como una **configuración de piezas intercambiables**, ayudando a mantener la coherencia, evitar duplicidades y fomentar la interoperabilidad entre áreas.

Ejemplos de ABB:

- Servicio de autenticación de usuario
- Proceso de onboarding de cliente
- Almacenamiento de datos geoespaciales

Ejemplos de SBB:

- Servicio OAuth2 en Azure AD
- Aplicación de onboarding desarrollada en Angular + .NET
- Base de datos PostgreSQL con extensión PostGIS

Un buen repositorio de building blocks permite a las organizaciones acelerar su capacidad de innovación mediante la reutilización sistemática de componentes existentes.

Componente	Descripción	Naturaleza	Ejemplos típicos	Relación con el ADM
Entregables	Documentos formales que recogen resultados completos y revisados	Conjunto estructurado	Documento de visión, Evaluación de arquitectura, Plan de transición	Se generan al final de fases clave (e.g. Fase A, E, F)

Artefactos	Elementos individuales que representan información arquitectónica específica	Documento o modelo	Catálogo de aplicaciones, Matriz CRUD, Diagrama de capas	Se elaboran dentro de cada fase del ADM
Building Blocks	Unidades modulares reutilizables de capacidades o componentes	Conceptual o técnico	Servicio de identidad, Proceso de pagos, API REST estándar	Se modelan en fases B–D y se refinan en E–F

Tabla 1 - Comparativa de componentes del ACF

2.3 Repositorio de Arquitectura

En el contexto del *Architecture Content Framework* (ACF), el **repositorio de arquitectura** es una pieza clave que permite gestionar de forma eficiente y controlada todos los elementos que conforman la arquitectura empresarial. No es simplemente un espacio para guardar documentos: es una **plataforma estructurada** que habilita la trazabilidad, la colaboración, la gobernanza y, sobre todo, la **reutilización sistemática del conocimiento arquitectónico**.

2.3.1 ¿Qué es un repositorio de arquitectura?

Un repositorio de arquitectura es un sistema de almacenamiento —ya sea una herramienta especializada, una plataforma colaborativa o un conjunto de carpetas estructuradas— que contiene **los entregables, artefactos y building blocks** generados durante el desarrollo y mantenimiento de la arquitectura empresarial.

Su función va más allá del archivado: debe permitir **versionado, control de cambios, consulta por parte de stakeholders, navegación por relaciones entre elementos, y recuperación de componentes reutilizables**. En términos prácticos, un buen repositorio actúa como **la memoria organizativa de la arquitectura**.

Una arquitectura sin repositorio es como una orquesta sin partitura: cada instrumento puede sonar bien por separado, pero el conjunto carecerá de armonía y dirección.

2.3.1.1 Beneficios clave del repositorio

- **Reutilización de componentes arquitectónicos** (building blocks ya definidos, patrones, catálogos)
- **Trazabilidad entre fases del ADM**, desde los requisitos iniciales hasta los componentes implementados
- **Referencia común** para todos los arquitectos, analistas y responsables de negocio
- **Auditoría y gobernanza** de las decisiones arquitectónicas tomadas
- **Gestión del cambio**, facilitando el impacto analysis ante modificaciones de requisitos o tecnología

2.3.1.2 ¿Qué contiene un repositorio TOGAF?

Aunque puede adaptarse según las necesidades de cada organización, un repositorio alineado con TOGAF 10 suele estructurarse en torno a las siguientes áreas:

Área	Contenido típico
Catálogos	Capacidades de negocio, procesos clave, actores, aplicaciones, objetos de datos, componentes tecnológicos
Matrices	Relaciones entre elementos: procesos vs. aplicaciones, roles vs. funciones, datos vs. sistemas, CRUD, cobertura normativa
Diagramas	Modelos visuales de arquitectura (por capas, integración, despliegue, información, etc.)
Artefactos adicionales	Roadmaps, análisis de brechas (*gap analysis*), requisitos, vistas por stakeholder
Building Blocks reutilizables	Servicios, procesos, componentes tecnológicos modulares, APIs, librerías comunes
Decisiones arquitectónicas (ADR)	Registro de decisiones clave tomadas, con motivación, alternativas consideradas, impactos esperados
Normativas y principios	Reglas de diseño, estándares técnicos, principios de arquitectura definidos por el Architecture Board

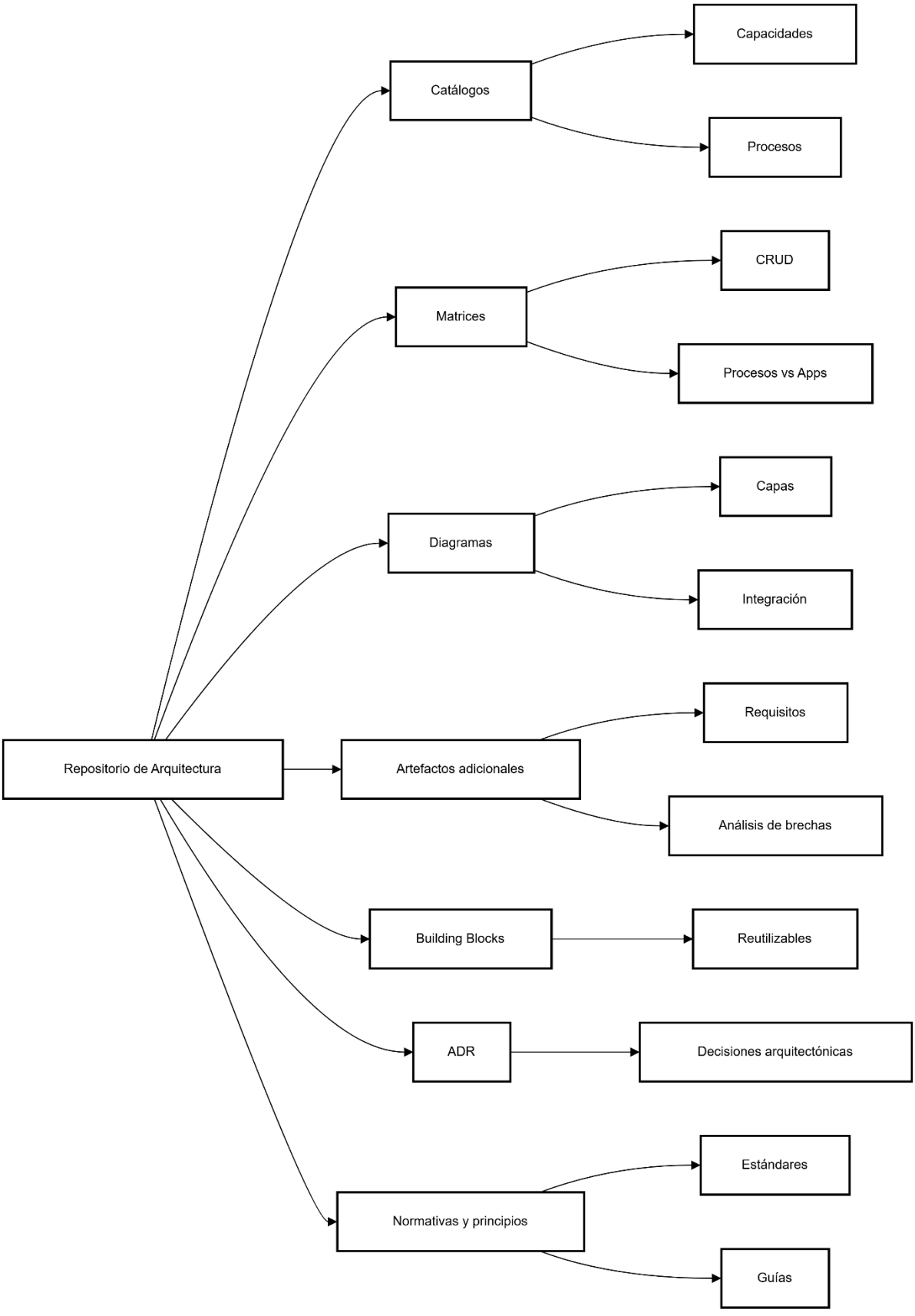
Un buen repositorio debe alinearse con la estructura de fases del método ADM —que exploraremos más adelante—, facilitando que cada fase produzca y consuma artefactos coherentes y trazables.

2.3.2 Herramientas para el repositorio

TOGAF no impone el uso de una herramienta específica, pero recomienda que el repositorio sea accesible, mantenible y colaborativo. Algunas opciones comunes:

- **Herramientas especializadas**: Sparx EA, Avolution ABACUS, BiZZdesign, HOPEX (caras, pero potentes).
- **Repositorios abiertos y colaborativos**: Archi + Git, Wiki corporativa + Draw.io, Notion + PlantUML (alternativas accesibles y personalizables).
- **Entornos corporativos seguros**: Azure DevOps Wiki + Repos, SharePoint versionado, Confluence + Jira Integration.

Lo importante no es la herramienta en sí, sino que el repositorio permita una arquitectura **viva, auditable, reutilizable y comprensible por todos los stakeholders**.


Repositorio de Arquitectura
Catálogos
Capacidades
Procesos
Matrices
CRUD
Procesos vs Apps
Diagramas
Capas
Integración
Artefactos adicionales
Requisitos
Análisis de brechas
Building Blocks
Reutilizables
ADR
Decisiones arquitectónicas
Normativas y principios
Estándares
Guías

TOGAF no impone el uso de una herramienta específica, pero recomienda que el repositorio sea accesible, mantenible y colaborativo. Algunas opciones comunes:

2.3.3 Mi planteamiento de un repositorio

Como parte de mi enfoque práctico de arquitectura empresarial, propongo una forma estandarizada de estructurar la documentación basada en TOGAF 10, reforzada por una metodología propia denominada **ArcDoc+**, y aplicada a casos reales como de arquitectura de microservicios .NET desplegada en Kubernetes.

Esta estructura está pensada para ser clara, modular y navegable. Puedes formalizarla en una wiki técnica o repositorio Git— permitiendo una trazabilidad completa de todos los entregables generados a lo largo del ciclo ADM.

FALTA

2.4 Tipos de Artefactos Arquitectónicos

En toda iniciativa de arquitectura empresarial, la producción de artefactos es una actividad esencial. Estos artefactos constituyen la evidencia tangible del pensamiento arquitectónico y sirven como medio de comunicación entre los distintos actores involucrados: desde equipos técnicos hasta directivos, pasando por usuarios, proveedores o responsables de cumplimiento normativo.

El **Architecture Content Framework (ACF)** de TOGAF propone una clasificación práctica de los artefactos, basada en su función y nivel de abstracción. Esta categorización no solo mejora la trazabilidad y consistencia de los modelos generados, sino que también favorece su reutilización, validación y mantenimiento a lo largo del tiempo.

A continuación, se describen los tres tipos principales de artefactos arquitectónicos:

2.4.1 Catálogos

Los catálogos son **listas estructuradas** de elementos arquitectónicos que existen o se planifican dentro de la organización. Su función es registrar de manera ordenada los distintos objetos que forman parte de la arquitectura, como:

- Capacidades de negocio
- Aplicaciones en uso
- Entidades de datos
- Tecnologías disponibles
- Servicios o APIs expuestos

Un buen catálogo permite responder preguntas clave como: *¿Qué tenemos hoy?, ¿qué capacidades están duplicadas?, ¿qué sistemas se solapan?, ¿qué datos son críticos?*

Ejemplo para EcoShop:

ID	Nombre de Aplicación	Dominio Funcional	Tecnología	Estado
APP-001	Gestor de Productos	Catálogo / Ecommerce	.NET Core	En producción
APP-002	Motor de Recomendaciones	Marketing / IA	Python	En desarrollo

2.4.2 Matrices

Las matrices permiten **relacionar elementos** entre sí y descubrir dependencias, impactos y alineamientos. Son especialmente útiles para entender la cohesión o dispersión de activos dentro de la arquitectura.

Ejemplos de relaciones comunes que se documentan en matrices:

- Funciones de negocio ↔ Aplicaciones que las soportan
- Aplicaciones ↔ Bases de datos que utilizan
- Capas tecnológicas ↔ Requerimientos de seguridad
- Procesos ↔ Riesgos identificados

Ejemplo para EcoShop:

Función de Negocio	Aplicación Soporte
Gestión de pedidos	Sistema de Pedidos (APP-004)
Personalización de ofertas	Motor de Recomendaciones (APP-002)
Atención al cliente	CRM Online (APP-006)

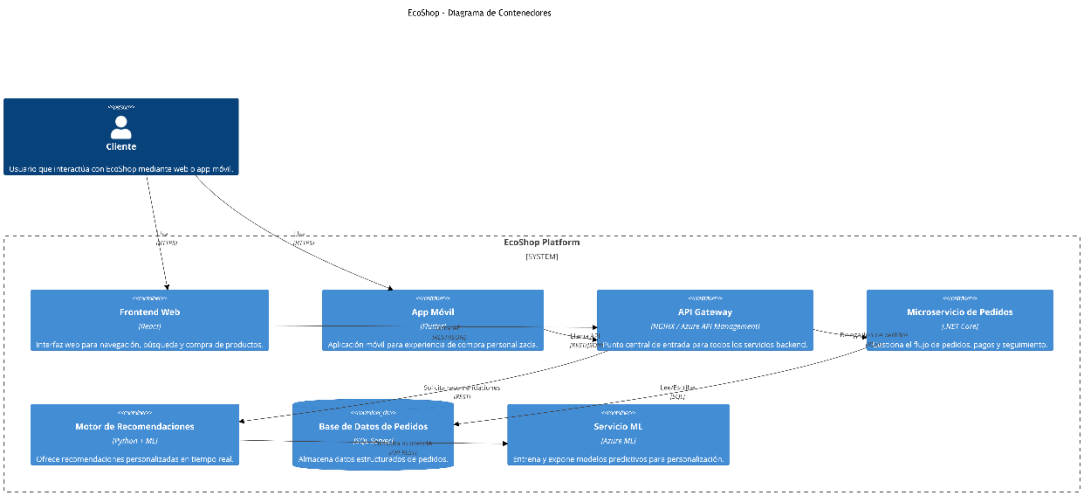
2.4.3 Diagramas

Los diagramas representan **visualmente** los componentes arquitectónicos y sus relaciones. Son esenciales para facilitar la comprensión, promover discusiones estratégicas y documentar decisiones clave.

Pueden adoptar diferentes notaciones y niveles de detalle, como:

- Diagramas de flujo de procesos (BPMN)
- Diagramas de arquitectura lógica
- Diagramas de despliegue físico
- Diagramas de componentes (como C4 Model)
- Diagramas de secuencia o casos de uso en UML

Ejemplo conceptual en EcoShop (en notación C4 Model):



Este tipo de representaciones no solo ilustra cómo se organiza la solución, sino que también permite detectar cuellos de botella, puntos únicos de fallo o áreas de mejora.

2.5 Aplicación práctica del ACF

El **Architecture Content Framework (ACF)** no es simplemente una taxonomía teórica; su verdadero valor se manifiesta en cómo se aplica en el día a día de una iniciativa de arquitectura empresarial. A través de su estructura clara y reutilizable, el ACF actúa como el hilo conductor que une los entregables arquitectónicos con el método de desarrollo (ADM) y garantiza que todos los esfuerzos estén alineados, documentados y trazables.

A continuación, se detallan los principales beneficios prácticos de aplicar el ACF en un proyecto real como **EcoShop**:

2.5.1 Mantener la coherencia documental

El ACF establece convenciones comunes sobre **cómo se nombra, organiza y presenta** cada tipo de artefacto. Esto evita que distintos equipos documenten lo mismo de forma redundante o incoherente. En el caso de EcoShop, al usar el ACF se definieron claramente catálogos como el de **Building Blocks** y **funciones de negocio**, así como matrices como la de **trazabilidad de requisitos**, siguiendo un mismo patrón documental que puede ser comprendido y mantenido por todo el equipo, sin importar la fase en la que estén.

2.5.2 Evitar ambigüedades al describir la arquitectura

Muchos errores arquitectónicos no son de diseño, sino de **interpretación**. Un ACF bien implementado define claramente **qué representa un artefacto, para qué se usa y con qué nivel de detalle**. Por ejemplo, cuando en EcoShop se modela el microservicio de pedidos, el ACF obliga a documentar tanto su descripción funcional como su diagrama C4, su API en OpenAPI y su dependencia con la base de datos de pedidos, evitando suposiciones o lagunas.

2.5.3 Promover la reutilización de componentes arquitectónicos

Gracias a su orientación modular, el ACF permite identificar patrones o componentes (building blocks) que pueden ser reutilizados en distintas iniciativas. En EcoShop, por ejemplo, el componente de autenticación se definió como **building block independiente**, reutilizable en futuros proyectos de canales digitales (como apps de fidelización o área privada de clientes). Además, los modelos de datos pueden replicarse para nuevos productos sin partir de cero.

2.5.4 Mejorar la calidad y trazabilidad de las decisiones arquitectónicas

El ACF promueve prácticas como el uso de **ADRs (Architectural Decision Records)** y matrices de impacto, lo que permite saber no solo **qué** decisiones se tomaron, sino **por qué**, **cuándo**, y con qué riesgos asociados. En el caso de EcoShop, el ADR que documenta la elección de arquitectura de microservicios sobre monolito describe las alternativas consideradas, su impacto en mantenimiento, escalabilidad y su trazabilidad hasta el objetivo estratégico de ofrecer más agilidad.

2.5.5 Alinear los entregables con las fases del ADM

Cada artefacto definido en el ACF está **alineado con una fase del ADM**, lo cual facilita su adopción progresiva en función del avance del proyecto. Por ejemplo:

Fase ADM	Artefactos típicos ACF
Fase A (Visión)	Documento de visión, Declaración de trabajo, Caso de negocio
Fase B	Catálogo de capacidades, BPMN de procesos clave
Fase C	Modelo de datos, Servicios OpenAPI, Diagrama C4
Fase D	Catálogo tecnológico, Infra as Code, Stack de referencia
Fase E	Evaluación de soluciones, Building Blocks, Repositorio de componentes
Fase F	Plan de releases, pipelines DevOps, matriz de migración

Fases G-H	Informes de calidad, control de versiones, gestión del cambio

En EcoShop, esto se traduce en un repositorio estructurado (`ArcDoc-SORA`) donde los artefactos están organizados según las fases del ADM, lo cual facilita auditorías, revisiones y evolución continua.

2.6 Ejemplo de Repositorio de Arquitectura

Imaginemos una organización —pongamos que se trata de **EcoShop**, en su vertiente B2B— que decide **modernizar su sistema de gestión de clientes (CRM)** para adaptarlo a un enfoque omnicanal, más automatizado y conectado con el resto del ecosistema digital.

Este esfuerzo implica rediseñar procesos, migrar tecnologías, y alinear múltiples departamentos. El éxito de la iniciativa dependerá, entre otras cosas, de una **arquitectura empresarial sólida y bien documentada**. Aquí es donde el **Architecture Content Framework (ACF)** cobra valor: actúa como el esqueleto documental que permite organizar, relacionar y comunicar los artefactos arquitectónicos generados a lo largo del proyecto.

A continuación, se muestra cómo se utilizaría el ACF en esta iniciativa:

2.6.1 Catálogo de aplicaciones existentes

El punto de partida es entender el ecosistema actual. Para ello, se crea un catálogo de aplicaciones centrado en aquellas que gestionan o interactúan con datos de clientes.

ID	Nombre de Aplicación	Función principal	Tecnología	Estado
APP-CRM01	CRM On-Premise	Gestión central de clientes	Oracle	Activa
APP-WEB01	Portal Web de Clientes	Visualización y edición de datos	React + API	En uso
APP-CAMP01	Plataforma de Campañas	Envío de email marketing	SaaS	Integrada

Este catálogo permite identificar **redundancias, riesgos de obsolescencia y puntos de integración** clave.

2.6.2 Matriz de dependencias entre procesos y sistemas

Se documenta una matriz que relaciona los procesos críticos con las aplicaciones que los soportan. Esto revela, por ejemplo, **procesos manuales mal automatizados** o **sistemas que tienen múltiples responsabilidades**.

Proceso	Aplicación Principal	Notas
Alta de cliente	CRM On-Premise	Flujo parcialmente manual
Actualización de datos	Portal Web de Clientes	Sin validación cruzada

Segmentación para campañas	Plataforma de Campañas	Depende de exportaciones manuales

Gracias a esta matriz, se visualiza que la falta de integración genera **fricción operativa** y pérdida de eficiencia.

2.6.3 Diagramas de interacción y flujos de usuario

Los diagramas representan visualmente **cómo los sistemas interactúan** y cómo los usuarios recorren los distintos canales.

2.6.3.1 Ejemplo de diagrama (nivel C4 – contenedores):

Para complementar los catálogos y matrices anteriores, resulta fundamental representar la arquitectura de forma visual. El modelo C4 proporciona una notación sencilla y eficaz para comunicar arquitecturas a distintos niveles de detalle, siendo el nivel de contenedores especialmente útil en fases tempranas del ADM.

En el caso de EcoShop, el diagrama de contenedores permite visualizar cómo se organiza la plataforma objetivo y cómo interactúan sus principales componentes.

Descripción del diagrama (C4 – Contenedores):

La arquitectura target de EcoShop se estructura en los siguientes contenedores principales:

- Frontend Web Omnicanal
  - Aplicación web utilizada por clientes y agentes comerciales para la gestión de clientes, pedidos y campañas.
  - Tecnología: SPA (React / Angular).
- API de Gestión de Clientes
  - Servicio backend responsable de la gestión centralizada de datos de cliente, validaciones y reglas de negocio.
  - Tecnología: API REST (.NET / Java).
- Servicio de Campañas y Segmentación
  - Componente encargado de la lógica de segmentación y activación de campañas, integrado con herramientas SaaS de marketing.
- Plataforma de Datos de Clientes
  - Repositorio central de datos maestros de cliente, histórico de interacciones y preferencias.
  - Tecnología: Base de datos relacional y/o plataforma analítica.
- Sistemas Externos
  - Sistemas heredados y plataformas externas (ERP, sistemas de facturación, proveedores de mensajería).

Relaciones clave:

- El frontend consume exclusivamente APIs expuestas por la capa de servicios.
- Los servicios backend acceden a la plataforma de datos de forma controlada.
- Las integraciones con sistemas externos se realizan mediante interfaces bien definidas.

Este diagrama permite:

- Entender rápidamente la estructura de la solución.
- Identificar responsabilidades claras por componente.
- Servir como base para fases posteriores de diseño técnico (Fase D).

2.7 Relación entre el ACF y el ADM

El Architecture Content Framework (ACF) no es un fin en sí mismo, sino un soporte estructurado para aplicar el método central de TOGAF: el Architecture Development Method (ADM).

Aunque todavía no hemos abordado el ADM en profundidad —lo haremos más adelante—, es importante entender desde ahora que el ADM es un ciclo iterativo de fases que guía el desarrollo de una arquitectura empresarial desde su concepción estratégica hasta su evolución continua. Cada fase del ADM da lugar a una serie de entregables concretos, y es precisamente el ACF el que establece cómo deben documentarse, clasificarse y conservarse esos entregables para garantizar trazabilidad, coherencia y reutilización.

2.7.1 Relación práctica: Fases ADM y Artefactos ACF

Cada fase del ADM tiene una serie de artefactos esperados, que deben documentarse y almacenarse en el repositorio ACF de forma ordenada. Por ejemplo:

Fase B – Arquitectura de Negocio (vista preliminar)

Aunque el detalle de esta fase se cubrirá más adelante, podemos adelantar que se encarga de modelar el funcionamiento actual y futuro del negocio. En esta fase, el ACF organiza los siguientes artefactos típicos:

Catálogo de Capacidades de Negocio: Lista estructurada de lo que la organización puede hacer (ej. gestión de campañas, procesamiento de pedidos).

Diagrama de Procesos Clave: Representación visual en BPMN u otro lenguaje estándar que muestre los procesos relevantes que deben mantenerse o rediseñarse.

Matriz de Actores y Funciones: Tabla que cruza roles organizativos (actores) con las funciones o actividades que ejecutan en distintos procesos.

2.7.2 Objetivo del ACF en relación con el ADM

El Architecture Content Framework establece una correspondencia clara entre cada fase del ADM y los artefactos que deben producirse. Esta relación asegura que el esfuerzo arquitectónico no se traduzca en documentación dispersa, sino en conocimiento estructurado y reutilizable.

La siguiente tabla resume esta relación de forma práctica:

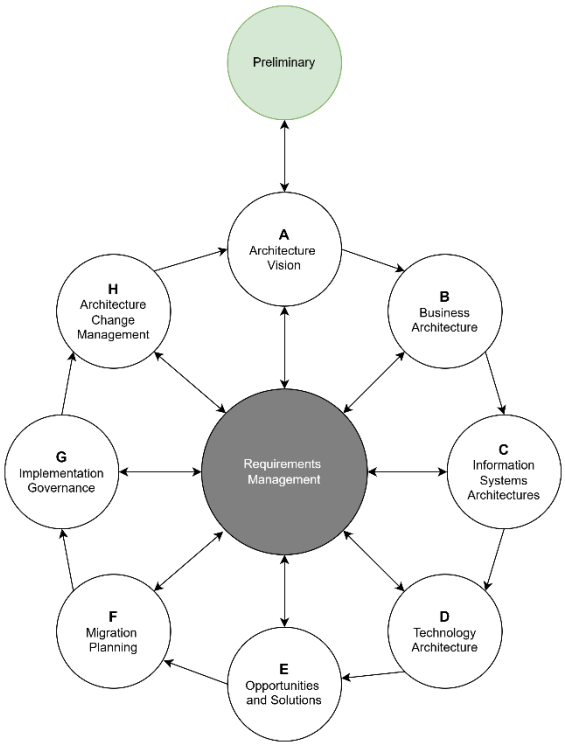
Fase ADM	Artefactos principales (ACF)	Propósito
Preliminary	Principios de arquitectura, Modelo de gobernanza, Repositorio inicial	Preparar la organización y establecer reglas de juego
Fase A – Vision	Documento de Visión, Stakeholder Map, Business Drivers	Alinear arquitectura y estrategia
Fase B – Business	Mapa de capacidades, Diagramas de procesos, Matriz actores-funciones	Definir cómo debe operar el negocio
Fase C – Data & App	Catálogo de aplicaciones, Modelos de datos, Diagramas lógicos	Traducir negocio a sistemas
Fase D – Technology	Arquitectura técnica, estándares, diagramas de despliegue	Definir la plataforma tecnológica
Fase E – Opportunities	Catálogo de soluciones, building blocks reutilizables	Seleccionar soluciones viables
Fase F – Migration	Roadmap, plan de transición, dependencias	Planificar la evolución
Fase G – Governance	Arquitectural compliance reviews, desviaciones	Asegurar cumplimiento
Fase H – Change	Actualización de repositorio, nuevas decisiones	Mantener arquitectura viva

No te preocupes si aún no conoces las fases del ADM o su secuencia exacta: en capítulos posteriores las desgranaremos en detalle, con ejemplos aplicados al caso EcoShop. De momento, basta con que entiendas que el ACF actúa como un "contenedor estructurado" donde se depositan los artefactos generados a lo largo del proceso arquitectónico.

3 ARCHITECTURE DEVELOPMENT METHOD (ADM)

Este es el núcleo **práctico**, donde se vamos a desglosar paso a paso cómo aplicar el **Método de Desarrollo de Arquitectura (ADM)** de TOGAF 10 en un caso real. A diferencia de las partes anteriores, aquí vamos a seguir un único ejemplo común – *una empresa ficticia de comercio electrónico que rediseña su plataforma tecnológica* – para ilustrar cada fase del ADM. El objetivo es mostrar en paralelo la teoría de cada fase y su aplicación práctica, profundizando en objetivos, entradas, salidas, roles involucrados, artefactos clave y decisiones estratégicas de cada fase.

3.1 FASE PRELIMINAR – PREPARAR LA ORGANIZACIÓN



La **Fase Preliminar** del ADM se enfoca en **preparar a la organización** para llevar a cabo iniciativas de arquitectura empresarial. Muchas veces esta fase no recibe la atención que merece, pero es fundamental para el éxito de todo el ciclo ADM. Antes de "construir" cualquier arquitectura, es necesario **alinear, habilitar y organizar** a la empresa para el cambio. En otras palabras, se establecen los cimientos sobre los cuales se desarrollará el trabajo arquitectónico posterior.

3.1.1 Objetivo

Establecer las bases organizativas, estructurales y operativas necesarias para iniciar una práctica de arquitectura empresarial sólida. Esto implica asegurar que la empresa cuenta con la capacidad y la gobernanza adecuadas para ejecutar el ADM. En concreto, la Fase Preliminar busca:

- **Definir la capacidad de arquitectura empresarial** de la organización (recursos, procesos y herramientas necesarios).

- **Alinear el esfuerzo de arquitectura con la estrategia** y el contexto del negocio (asegurando patrocinio ejecutivo y entendimiento del **por qué** de la transformación).
- **Establecer principios de arquitectura** que guiarán las decisiones durante todo el ciclo ADM.
- **Organizar la función de arquitectura**: clarificar roles, responsabilidades y estructuras de gobernanza (por ejemplo, la creación de un Architecture Board o comité de arquitectura).

3.1.2 ¿Qué se hace en esta fase?

En la Fase Preliminar típicamente se llevan a cabo las siguientes **actividades clave**:

- **Comprender el contexto de la organización**: Se analiza la situación actual de la empresa en cuanto a estrategia, estructura y procesos. Se identifican las **motivaciones** para emprender el programa de arquitectura (p.ej., necesidad de transformación digital, resolver ineficiencias, crecimiento, presión competitiva). También se evalúa el **nivel de madurez** arquitectónica existente (¿la empresa ha hecho arquitectura antes? ¿Qué tan formalizados están sus procesos de TI y negocio?). *Se puede emplear aquí una matriz de madurez de arquitectura empresarial para evaluar brechas en capacidades actuales.*

Dimensión	Nivel 1 – Inicial	Nivel 2 – Repetible	Nivel 3 – Definido	Nivel 4 – Gestionado	Nivel 5 – Optimizado
Gobernanza	No formalizada	Algunas reglas locales	Estructura formal definida	Junta de arquitectura activa	Gobernanza ágil y basada en valor
Métodos y procesos	Ad-hoc	Documentados parcialmente	Definidos por dominio	Integrados en todo el ciclo ADM	Revisados y mejorados continuamente
Herramientas	Documentos sueltos	Uso parcial de herramientas	Herramientas compartidas	Repositorio centralizado	Automatización e IA incorporada
Roles y estructura	Sin roles claros	Responsables informales	Roles asignados por dominio	Organigrama arquitectónico activo	Estructura federada y adaptable
Valor aportado	Difícil de demostrar	Casos puntuales	Valor percibido por TI	Reconocido por negocio y TI	Medido y alineado con KPIs de negocio
Competencias del equipo	Desigual y no documentada	Perfil técnico emergente	Mixto técnico-funcional	Evaluación continua y formación	Skills estratégicos y coach internos

- **Definir la capacidad de Arquitectura Empresarial**: Se establece cómo se organizará internamente la práctica de arquitectura. Esto incluye la definición de **roles y responsabilidades** (p. ej., nombrar un arquitecto líder o **Chief Architect**, definir miembros del equipo de arquitectura, identificar stakeholders clave y sus expectativas). También se crea o formaliza el **Architecture Board** (comité directivo de arquitectura) y se asignan responsables para el **repositorio de arquitectura** y la gestión de artefactos. Es decir, se define el **modelo organizativo** para llevar adelante la arquitectura empresarial.
- **Seleccionar frameworks, métodos y herramientas de apoyo**: Se decide con qué **marcos complementarios** trabajar junto con TOGAF. Por ejemplo, se puede combinar TOGAF con marcos de gestión de proyectos (PMBOK, Agile SAFe), gestión de servicios TI (ITIL), u otros estándares como COBIT para gobierno de TI, e incluso lenguajes de modelado como ArchiMate para documentar arquitecturas. Asimismo, se seleccionan las

herramientas que se usarán para el modelado y la documentación (p. ej., herramientas de modelado arquitectónico, repositorios, wikis corporativas) y se prepara un plan de adopción de dichas herramientas.

- **Establecer los principios de arquitectura**: Se elaboran y acuerdan los **principios** que guiarán todas las decisiones de diseño durante el ciclo. Los principios de arquitectura son reglas o declaraciones fundamentales (por ejemplo: *"priorizar la interoperabilidad", "seguridad por diseño", "orientación a servicios (SOA)", "evitar redundancia de sistemas"*). Estos principios deben ser aprobados por la alta dirección y comunicados a todos los involucrados, ya que servirán como criterios de evaluación en las fases posteriores. Lo ideal es documentar cada principio junto con su justificación y las implicaciones de adoptarlo.

ID	Principio	Descripción	Motivación	Implicaciones
P01	Interoperabilidad	Los sistemas deben poder integrarse de forma estándar entre sí.	Evita silos, permite integración más rápida y menos costosa.	Requiere uso de APIs abiertas y protocolos comunes.
P02	Seguridad por diseño	La seguridad debe incorporarse desde las fases iniciales del diseño.	Reduce riesgos y evita costosos rediseños posteriores.	Mayor involucración de seguridad en revisiones de arquitectura.
P03	Reutilización	Los componentes deben diseñarse para ser reutilizables.	Ahorro de tiempo, menor coste y consistencia técnica.	Implica mantenimiento de un catálogo de building blocks y mayor disciplina técnica.
P04	Gestión del cambio controlada	Todo cambio en la arquitectura debe seguir procesos de evaluación.	Reduce el caos y evita impactos inesperados.	Mayor carga en gobierno arquitectónico, uso de herramientas de trazabilidad.
P05	Orientación al usuario final	Las decisiones deben centrarse en la experiencia del usuario final.	Mejora la adopción, satisfacción y valor de negocio.	Deben integrarse métricas UX y feedback en el ciclo de arquitectura.
P06	Modularidad	Las soluciones deben diseñarse como componentes desacoplados.	Facilita mantenimiento, escalabilidad y pruebas independientes.	Implica mayor esfuerzo inicial en diseño técnico y definición de interfaces.

- **Definir la gobernanza inicial y procesos**: Se alinean los procesos de arquitectura con los **procesos de gobierno existentes** en la empresa. Por ejemplo, integrar la arquitectura en el ciclo de planeación estratégica, en los comités de inversión de TI, o en metodologías de desarrollo de proyectos. Se establecen procedimientos para control de cambios, manejo

de **excepciones** (cuando haya que tomar decisiones que se apartan de los estándares) y aseguramiento de la calidad de las arquitecturas. Básicamente, se sientan las reglas de juego para cómo se tomarán decisiones arquitectónicas y cómo se comunicarán.

3.1.3 Entradas típicas

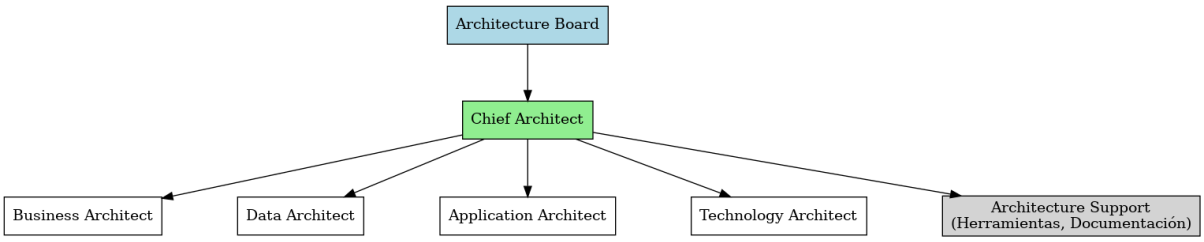
Para iniciar la Fase Preliminar, suele contarse con:

- Un **mandato estratégico** o la necesidad reconocida de transformación (por ejemplo, una directiva ejecutiva de modernizar la plataforma e-commerce).
- **Contexto de negocio**: la estrategia corporativa vigente, planes de negocio, objetivos a mediano plazo.
- **Marcos de referencia internos o externos ya existentes**: políticas corporativas, estándares de TI actuales, modelos de procesos (si los hubiera), estándares de la industria, etc.
- **Documentación de iniciativas previas** de arquitectura o TI: organigramas de TI, inventarios de aplicaciones, dolores identificados en sistemas actuales, auditorías de TI, etc., que ayuden a entender desde dónde se parte.

3.1.4 Salidas clave (Entregables)

Al completar la Fase Preliminar, la organización debería haber generado una serie de artefactos que conforman el cimiento para el resto del ADM:

- **Principios de Arquitectura definidos y aprobados**: Documento formal con los principios acordados, firmado por los patrocinadores adecuados.
- **Modelo de gobernanza de arquitectura establecido**: Esto puede incluir la creación de un **Architecture Board** formal, con sus miembros, roles (p. ej., quién aprueba arquitecturas) y frecuencia de reuniones definida.
- **Organización y roles del equipo de arquitectura claros**: Una estructura definida que indique quién es el arquitecto líder, arquitectos de dominios (negocio, datos, aplicaciones, tecnología) si corresponde, patrocinador ejecutivo (ej. CIO) y otros stakeholders clave. Puede reflejarse en un **diagrama organizativo**.



- **Alcance inicial y enfoque del marco de arquitectura**: Por ejemplo, un marco TOGAF adaptado (tailored) a la organización. Esto podría documentarse como un **charter** o carta de arquitectura que define cómo se usará TOGAF, qué partes se aplicarán más estrictamente, adaptaciones terminológicas, etc.

- **Plan de adopción de herramientas y repositorio**: Un mini-plan que detalle qué herramienta de modelado o repositorio se utilizará, cómo se implementará y entrenará al equipo, y un repositorio base inicial con cualquier artefacto existente cargado.
- **Declaración de Trabajo de Arquitectura (Architecture Work Statement) inicial**: Este es un entregable que a veces se inicia en Fase Preliminar (y se finaliza en Fase A). Incluye la definición preliminar del alcance del esfuerzo de arquitectura que se va a emprender, presupuesto estimado, recursos asignados y aprobación para proceder. En nuestro caso, sería el documento que autoriza formalmente iniciar la arquitectura para la plataforma e-commerce.

3.1.5 Roles involucrados

Durante la Fase Preliminar, suelen participar:

- **Patrocinador Ejecutivo** – por ejemplo, un CIO, CTO o alto directivo que impulse la iniciativa y asegure apoyo organizacional.
- **Arquitecto Jefe (Lead Architect o Enterprise Architect)** – responsable de liderar la definición de la función de arquitectura y del framework a usar.
- **Futuros Arquitectos de Dominio** – como el Arquitecto de Negocio, Arquitecto de Datos, Arquitecto de Aplicaciones, Arquitecto Tecnológico, quienes aportarán a definir lineamientos en sus áreas.
- **Stakeholders clave de negocio y TI** – líderes de unidades de negocio, jefes de TI, etc., que deben estar alineados con los principios y asegurar que la práctica de arquitectura se integra con sus áreas.
- **Equipo de Gobierno de TI** – si existe un área de governance o PMO, para alinear procesos y evitar duplicidades.

3.1.6 Decisiones estratégicas en esta fase

La Fase Preliminar conlleva decisiones que impactarán en todo el proyecto de arquitectura, por ejemplo:

- **Alcance de la arquitectura** – Decidir qué parte de la empresa o qué sistemas cubrirá el esfuerzo arquitectónico. En nuestro caso, ¿solo la plataforma e-commerce en sí, o también los sistemas periféricos (inventario, CRM, logística)?
- **Nivel de compromiso** – Asegurar el patrocinio: ¿quién proveerá recursos y autoridad? Se decide si se requiere un Architecture Board formal y quiénes lo integran.
- **Frameworks y métodos** – Elegir si se usará TOGAF puro, híbrido con otros métodos (p. ej., combinar con metodologías ágiles para ejecución).
- **Herramientas** – Decidir las herramientas de modelado/documentación a emplear (por ejemplo, una suite EA vs. herramientas ofimáticas). Esta decisión afecta la consistencia y colaboración en la documentación.

Funcionalidades típicas de una suite EA

Categoría	Funcionalidades clave
Modelado de arquitectura	Diagramas ArchiMate, BPMN, UML, mapas de capacidades, catálogos de aplicaciones
Repositorio central	Almacenamiento estructurado de artefactos, building blocks, relaciones y metadatos
Trazabilidad	Relación entre estrategias → procesos → aplicaciones → datos → infraestructura
Análisis de impacto	¿Qué cambia si se retira una app o se introduce una nueva tecnología?
Gobernanza y compliance	Control de versiones, flujos de aprobación, validación contra principios
Visualización y reporting	Dashboards, gráficos de relaciones, mapas de calor, análisis de redundancias
Integraciones	Con CMDBs, herramientas DevOps, ITSM, ERP, etc.

Ejemplos de suites EA conocidas: Sparx Enterprise Architect, LeanIX, BiZZdesign Enterprise Studio, entre otras.

Nota personal: ¿Realmente necesito una suite EA?

No soy especialmente partidario de depender exclusivamente de este tipo de herramientas. Muchas veces he visto cómo se convierten en un fin en sí mismas, imponiendo burocracia, procesos rígidos y un alto coste de adopción. Lo que debería ser una ayuda para pensar, modelar y comunicar, termina en algunos casos siendo una barrera para avanzar con agilidad.

Personalmente, creo que la arquitectura no debe depender de la herramienta, sino del criterio, la conversación y la claridad del modelo mental. Y en ese sentido, el mundo open source y las herramientas ligeras nos ofrecen alternativas perfectamente válidas y mucho más accesibles.

Por ejemplo:

- *Archi: una herramienta open source excelente para modelar con ArchiMate. Ligera, multiplataforma, muy utilizada por comunidades TOGAF. Ideal para quienes quieren enfocarse en los modelos, no en la herramienta.*
- *Modelio: otra suite open source más generalista, con soporte para UML, BPMN y extensiones para marcos como TOGAF. Buena opción para quienes quieren algo más visual sin renunciar a estándares.*

Además, he visto equipos hacer arquitectura perfectamente funcional usando Markdown, Git, draw.io o incluso Notion, siempre que haya claridad metodológica, gobernanza ligera y orientación a resultados.

En definitiva, una buena arquitectura nace de las preguntas correctas y de la colaboración entre personas. La herramienta es solo un medio. Escoge la que potencie tu práctica, no la que la complique.

- **Principios fundamentales** – Definir y priorizar principios (por ejemplo, decidir si "cloud-first" será un principio rector tecnológico, o si "orientación al cliente" será un principio de negocio clave). Estas decisiones guiarán todas las fases posteriores, por lo que se toman con cuidado y aprobadas al más alto nivel.

3.1.7 Ejemplo práctico (EcoShop) – Fase Preliminar

- **Fase Preliminar**: EcoShop, una empresa de comercio electrónico en crecimiento, decide emprender la modernización de su plataforma tecnológica para habilitar nuevos servicios digitales y mejorar la experiencia de sus clientes. Durante la Fase Preliminar, EcoShop realiza varias acciones preparatorias:
    - **Patrocinio y visión inicial**: El CIO de EcoShop, junto con la Directora de Negocio Digital, actúan como **patrocinadores ejecutivos**. Plantean la necesidad de actualizar la plataforma e-commerce para soportar un aumento previsto del 50% en transacciones anuales, integrar canales móviles y aplicar análisis de datos para marketing personalizado. Este mandato estratégico provee el **impulso inicial**: la transformación de la plataforma se alinea con la estrategia corporativa de crecimiento y mejora de experiencia del cliente.
    - **Creación de la función de arquitectura**: EcoShop nombra a un **Arquitecto Empresarial** líder, Juan, para encabezar el esfuerzo. Juan conforma un pequeño **equipo de arquitectura** con un arquitecto de aplicaciones, una arquitecta de datos y un arquitecto tecnológico. Juntos definen cómo se organizarán: establecen un **Architecture Board** incluyendo al CIO, a la Directora de Negocio Digital, al Gerente de Desarrollo de Software y al líder de Infraestructura. Este Architecture Board revisará y aprobará las entregas clave de cada fase. También se decide integrar al **Gerente de Marketing** como stakeholder, ya que la nueva plataforma impactará campañas y ventas online.
    - **Framework y herramientas**: El equipo de Juan adopta TOGAF 10 como marco principal y acuerda usar **ArchiMate** para los modelos. Deciden apoyarse en un repositorio existente (confluence corporativo y una herramienta EA) para almacenar todos los artefactos que se generen. Documentan en un breve "**Plan de Arquitectura**" cómo combinarán TOGAF con la metodología ágil ya utilizada por los equipos de desarrollo en EcoShop, de tal forma que la arquitectura y la entrega ágil se complementen.
    - **Principios de arquitectura**: En talleres con los directivos, Juan define un conjunto de principios. Por ejemplo*: "Experiencia omnicanal consistente"* (como principio de negocio), *"API-first e Integración abierta"* (principio de aplicaciones, para garantizar que todos los componentes nuevos expongan API y se integren fácilmente), "*Cloud-Preferred"* (principio tecnológico, priorizando soluciones en la

nube salvo justificación contraria), *"Seguridad por diseño"* (principio transversal). Estos principios se discuten, refinan y finalmente el Architecture Board de EcoShop los aprueba formalmente para guiar todo el rediseño de la plataforma.

- **Evaluación inicial y alcance**: El equipo de arquitectura de EcoShop realiza un **análisis de madurez** rápido: determinan que actualmente los procesos de TI son ad-hoc y no existe un repositorio central de arquitectura; esto refuerza la necesidad de seguir un marco disciplinado. Asimismo, delimitan el **alcance** del esfuerzo: se enfocarán en la plataforma de comercio electrónico (sitio web, aplicación móvil, backend de pedidos) y sus integraciones clave (sistema de gestión de inventario y sistema de atención al cliente), pero excluyen por ahora la parte de infraestructura logística física. Este alcance queda plasmado en la Declaración de Trabajo de Arquitectura, junto con estimaciones de tiempo (se prevé un roadmap a 18 meses) y recursos necesarios. Una vez revisada, el CIO aprueba esta declaración, dando luz verde para pasar a la siguiente fase.

Al finalizar esta fase preliminar, **EcoShop está preparada** para iniciar formalmente el ciclo ADM: tiene un equipo y una estructura de gobernanza en marcha, claridad en qué busca lograr con la arquitectura, y reglas básicas definidas (principios y procesos) para guiar los trabajos.

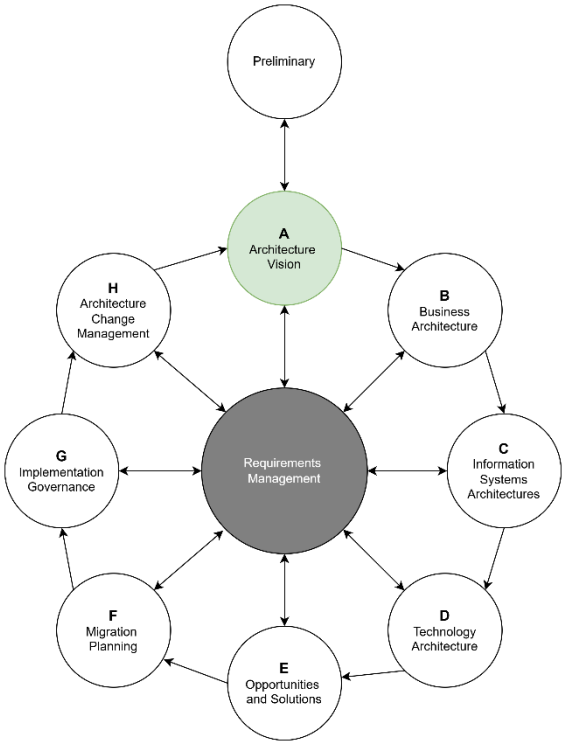
3.1.8 Resultado esperado

Tras la Fase Preliminar, la organización (EcoShop) debe quedar **estructuralmente lista** para ejecutar el ADM. En términos prácticos, esto significa:

- Contar con un **patrocinio claro** y objetivos alineados con la estrategia de negocio.
- Tener **roles definidos** (personas y comités asignados) para conducir y gobernar la arquitectura.
- Disponer de un **marco de arquitectura adaptado**, con principios formales aprobados y herramientas listas para usar.
- Poseer un **repositorio inicial** donde se centralizarán los artefactos a producir.

Estos elementos son el cimiento necesario. *Ignorar* o *hacer superficialmente esta fase suele traducirse en problemas más adelante* — por ejemplo, falta de apoyo ejecutivo, ambigüedad en responsabilidades, o esfuerzos de arquitectura desconectados de la realidad del negocio. EcoShop, al invertir tiempo en la Fase Preliminar, reduce esos riesgos antes de iniciar el trabajo detallado de arquitectura. Figura: Diagrama del Ciclo ADM completo, destacando la posición de la Fase Preliminar como punto de partida.

3.2 FASE A – ARCHITECTURE VISION



Una vez establecida la capacidad arquitectónica en la fase preliminar, comienza el ciclo iterativo del ADM con la **Fase A: Architecture Vision**. En esta fase se define la **visión arquitectónica** que guiará todo el esfuerzo y se asegura el apoyo de las partes interesadas (stakeholders). La Visión de Arquitectura es más que una simple idea inspiradora; es un entendimiento común de **qué problema se va a resolver, qué alcance tendrá la solución, quiénes se verán involucrados y qué valor se espera obtener**. Esta visión actúa como la brújula para las fases siguientes, alineando a negocio y TI en objetivos compartidos y justificando la inversión en la transformación propuesta.

3.2.1 Objetivo

El propósito de la Fase A es desarrollar una visión de alto nivel de la solución futura y obtener aprobación y apoyo para continuar. De manera concreta, se busca:

- **Clarificar la necesidad de cambio** y las metas estratégicas que la arquitectura deberá cumplir.
- **Definir el alcance** inicial de la arquitectura (qué incluirá y qué quedará fuera).
- **Identificar a los stakeholders clave** (interesados del negocio, usuarios, áreas de TI, etc.) y sus preocupaciones o requerimientos principales.
- **Desarrollar y comunicar la Visión de Arquitectura**: una descripción comprensible para negocio y TI de la solución propuesta y sus beneficios. Esto suele incluir visiones tanto de cómo funcionará el negocio como de la tecnología a alto nivel, sin entrar aún en detalles exhaustivos.

- **Formalizar el caso de negocio** y el valor que se espera obtener, para justificar la inversión.
- **Obtener la aprobación de la Declaración de Trabajo de Arquitectura (Architecture Work Statement)** que define formalmente el proyecto arquitectónico a llevar a cabo.

En resumen, en esta fase se define **qué problema se va a resolver, con qué objetivos y bajo qué condiciones**, asegurando el entendimiento y compromiso de todos los involucrados antes de profundizar en los detalles.

3.2.2 Actividades clave

Durante la Fase A, el equipo de arquitectura realiza varias actividades, generalmente en iteración con los stakeholders:

- **Identificar y analizar stakeholders**: Se elabora un **mapa de stakeholders** listando quiénes se verán afectados o interesados en la arquitectura. Por cada stakeholder o grupo, se identifican sus **inquietudes, necesidades y criterios de éxito**. Por ejemplo, se consideran ejecutivos de negocio (enfoque en ROI y mejoras operativas), usuarios finales (experiencia de usuario), áreas de TI (factibilidad técnica), reguladores (cumplimiento normativo), etc. Un resultado común es una matriz de stakeholders que relaciona cada stakeholder con sus principales preocupaciones. Este análisis ayuda a guiar la comunicación y a asegurar que la Visión responda a las expectativas críticas.

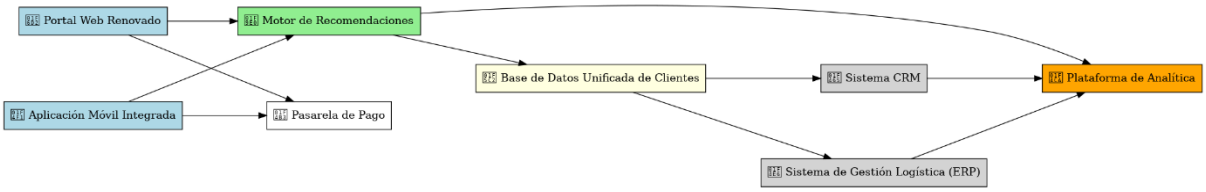
Stakeholder	Rol	Intereses principales	Nivel de influencia	Nivel de interés	Estrategia de involucramiento
CEO	Alta dirección	Alineación estratégica, retorno de inversión, innovación	Alto	Alto	Participación activa en visión y decisiones clave
CIO	Dirección tecnológica	Gobernanza tecnológica, escalabilidad, integración	Alto	Alto	Revisión de arquitectura y validación de roadmap
Director de Marketing	Unidad de negocio	Mejorar conversión, omnicanalidad, experiencia cliente	Medio	Alto	Entrevistas y validación de procesos clave
Responsable de Logística	Unidad de negocio	Automatización de almacenes, visibilidad de pedidos	Medio	Medio	Workshops de procesos y modelado de capacidades
Arquitecto de Software	Técnico	Coherencia técnica, reutilización, calidad del diseño	Alto	Alto	Participación directa en el modelado
Responsable de Seguridad	Técnico	Protección de datos, cumplimiento normativo	Medio	Alto	Validación de requisitos no funcionales
Usuario final (comprador)	Externo	Experiencia fluida, confianza, rapidez	Bajo	Alto	Análisis UX, encuestas, pruebas de usabilidad
Responsable de Atención al Cliente	Operativo	Seguimiento de pedidos, respuesta rápida, menos incidencias	Bajo	Medio	Revisión de procesos y sistemas de soporte

- **Definir el alcance de la arquitectura**: Se establece claramente qué dominios, procesos, sistemas y unidades de la empresa abarcará la iniciativa. Se describe tanto el alcance funcional (por ejemplo, "arquitectura de ventas en línea y gestión de pedidos") como el **alcance organizacional** (qué departamentos o regiones involucra). También se identifican supuestos y restricciones relevantes – por ejemplo, limitaciones de tiempo ("debe estar listo antes de la temporada de ventas navideñas"), de presupuesto o tecnológicas ("se usará la nube pública Azure como base, no se construirán datacenters propios").

- **Desarrollar la Visión de Arquitectura**: Se elabora un documento de Visión o Architecture Vision, que contiene una descripción de la situación actual y de la problemática, así como la visión a futuro. Esta suele incluir:

    - Una **declaración de visión** concisa (ej. "Lograr una plataforma e-commerce omnicanal, escalable y centrada en el cliente, que impulse un crecimiento del 20% en ventas online en 2 años").

    - Un **resumen de la arquitectura propuesta**, posiblemente con algún **diagrama de alto nivel** que muestre los componentes principales de la solución y cómo interactuarán.



    - **Beneficios esperados** claramente enumerados (p. ej., mejora de performance, nuevas capacidades de personalización, reducción de costos de mantenimiento, etc.).

    - **Impacto en la organización** (cambios en procesos, posibles necesidades de formación de personal, etc.).

    - Cualquier factor crítico de éxito y métricas clave para medirlos.

- **Desarrollar el Caso de Negocio**: En paralelo, se articula el valor de la iniciativa. Esto puede presentarse en forma de un caso de negocio que acompaña a la Visión. Incluye estimaciones de costos de implementación vs. beneficios (tangibles e intangibles), análisis de ROI, y riesgos de no hacer nada. El objetivo es responder "¿Por qué vale la pena esta transformación?" con argumentos sólidos para la dirección.

- **Identificar riesgos iniciales y mitigaciones**: Antes de entrar en detalles, se listan los principales riesgos que se vislumbran (p.ej., riesgo de interrupción del negocio durante la migración, resistencia al cambio por parte de ciertas áreas, incertidumbres tecnológicas) y posibles **estrategias de mitigación** o consideración de escenarios alternativos. Esto demuestra proactividad y prepara a la organización para manejar obstáculos.

- **Refinar requisitos y drivers de alto nivel**: Aunque la gestión de requisitos es continua, en esta fase se consolidan los requisitos de alto nivel del negocio y técnicos que la arquitectura deberá satisfacer (ejemplos: "permitir 1.000.000 de usuarios concurrentes", "cumplir con normativas de privacidad GDPR", "integrar con el sistema de logística existente"). Estos requisitos se documentan y vinculan a la visión como objetivos que la solución debe lograr. También se refinan los objetivos estratégicos y drivers de negocio que motivan el cambio (p. ej., "expandir ventas internacionales", "mejorar satisfacción del cliente").

- **Obtener aprobación y compromiso**: Finalmente, se presenta la Visión y el plan de trabajo propuesto a los patrocinadores y stakeholders clave (posiblemente en una **reunión de Architecture Board** o comité ejecutivo). El **hito** de esta fase es lograr la aprobación formal de la **Declaración de Trabajo de Arquitectura (Architecture Work Statement)** y el visto bueno para pasar a diseñar las arquitecturas de detalle en fases B, C y D. La declaración de trabajo detalla el qué (alcance y visión), el cómo (metodología, equipo), el cuándo (plan preliminar, hitos) y cuánto (recursos) del esfuerzo de arquitectura. Con esta aprobación, se asegura el **patrocinio continuo** y los recursos necesarios.

3.2.3 Entradas típicas

La Fase A se nutre de:

- **Resultados de la Fase Preliminar**: Principios de arquitectura acordados (que guiarán la visión), el alcance definido en la declaración de trabajo inicial, y la comprensión del contexto estratégico.

- **Mandatos estratégicos y drivers de negocio**: Por ejemplo, documentos de estrategia corporativa, metas cuantitativas (crecimiento, eficiencia) que la iniciativa de arquitectura deberá apoyar.

- **Necesidades del negocio o problemas identificados**: Cualquier análisis preliminar de problemas que motivaron el proyecto (baja velocidad del sitio web, dificultad para manejar picos de demanda, etc.).

- **Requisitos de alto nivel conocidos**: Puede incluir requisitos regulatorios (cumplimiento legal), requisitos técnicos imperativos (por ejemplo, "debe funcionar con la base de datos corporativa existente"), y expectativas de experiencia de usuario (por ejemplo, tiempo de carga máximo de 2 segundos por página).

- **Modelo de negocio actual**: Información sobre cómo opera hoy la empresa en el ámbito del proyecto (procesos de venta online, roles involucrados, flujos de información), para entender el punto de partida.

3.2.4 Salidas clave (Entregables)

Al concluir la Fase A, típicamente se generan:

- **Documento de Visión de Arquitectura** aprobado: que resume la visión a alto nivel de la arquitectura propuesta, incluyendo las secciones mencionadas (objetivos, alcance, descripción de la solución, beneficios, etc.).
- **Declaración de Trabajo de Arquitectura** aprobada: formaliza el proyecto de arquitectura. Incluye la definición del alcance, los objetivos, los roles y responsabilidades, el plan de trabajo y los recursos acordados para continuar con las fases B–D. Esta aprobación marca el compromiso de la organización para llevar adelante el proyecto.
- **Mapa o matriz de stakeholders**: artefacto que identifica las partes interesadas y sus preocupaciones, utilizado para planificar la comunicación y validaciones durante todo el ciclo.
- **Catálogo de requisitos de alto nivel**: una lista consolidada de requisitos empresariales y técnicos iniciales que la nueva arquitectura deberá cumplir (ésta se irá refinando conforme avancen las fases).
- **Evaluación inicial de riesgos**: un listado (o tabla) de riesgos principales y posibles mitigaciones, para tenerlos en el radar.
- **Actualizaciones al repositorio de arquitectura**: se comienza a llenar el repositorio con estos artefactos iniciales (visión, stakeholders, requisitos, etc.), creando la base de contenido para las siguientes fases.

3.2.5 Roles involucrados

En la Fase A intervienen:

- **Arquitecto Líder / Enterprise Architect**: Facilita las actividades de la fase, elabora la visión junto con las partes interesadas y asegura la alineación con principios y objetivos.
- **Stakeholders de negocio clave**: Por ejemplo, directores de unidades de negocio, gerentes de marketing, ventas, operaciones en nuestro caso e-commerce. Ellos aportan la perspectiva de negocio, validan que la visión atienda sus necesidades y dan apoyo ejecutivo.
- **Stakeholders de TI clave**: CIO/CTO, líderes de desarrollo, de infraestructura, seguridad, etc. Su rol es validar la factibilidad técnica de la visión y comprometer recursos de TI.
- **Equipo de Arquitectura de Negocio/Datos/Aplicaciones/Tecnología**: Los arquitectos de cada dominio contribuyen a delinear los aspectos de la visión relacionados con su área (ej. arquitecto de datos asegura que se considere una visión de cómo se manejarán los datos unificados de clientes).
- **Oficina de Gestión de Proyectos (PMO) o gerente de proyecto**: Puede participar para alinear la visión con los procesos de proyectos y asegurar que el plan de alto nivel sea realista en tiempos y costos.

3.2.6 Decisiones estratégicas

Algunas decisiones importantes que se toman en la Fase A y condicionan el resto del ADM incluyen:

- **Alcance final del proyecto arquitectónico**: Ajustar y confirmar el alcance (qué incluir/excluir) con base en viabilidad y valor. Esta decisión es crítica, pues un alcance demasiado amplio puede hacer el esfuerzo inmanejable, y uno muy estrecho podría no entregar suficiente valor.
- **Prioridad de objetivos y restricciones**: A veces se descubren objetivos conflictivos (por ejemplo, tiempo de entrega vs. profundidad de cambio). En la Visión se debe decidir qué prime. Por ejemplo, ¿es más importante modernizar toda la arquitectura aun si toma más tiempo, o entregar algo rápidamente aunque sea parcial? Estas prioridades estratégicas quedan claras aquí.
- **Aprobación de inversión**: La organización decide en esta fase si se compromete con la inversión necesaria. Este es un punto de **go/no-go**: con la información de la Visión y el caso de negocio, la alta gerencia decide si financia y apoya el programa de arquitectura completo.
- **Estrategia de solución general**: Se toman decisiones iniciales sobre el enfoque de la solución. Por ejemplo, en un e-commerce: ¿se construirá una plataforma completamente nueva o se mejorará incrementalmente la existente? ¿Se optará por soluciones comerciales (COTS) o desarrollo a medida? Aunque los detalles vendrán después, en la visión a veces se establece dirección general (ej., "adoptaremos una solución basada en microservicios en la nube, reutilizando componentes existentes donde sea posible").
- **Aceptación de riesgos conocidos**: Si en la evaluación inicial surgieron riesgos significativos, aquí se decide conscientemente asumirlos con planes de mitigación o ajustar el alcance para evitarlos. Por ejemplo, si un riesgo es la inmadurez de cierta tecnología, se decide si igualmente seguir adelante y monitorear, o si se reduce el alcance para no depender de ella.

3.2.7 Ejemplo práctico (EcoShop) – Fase A

En nuestro caso de EcoShop, la Fase A sirve para concretar la visión del rediseño de la plataforma e-commerce y asegurar que todos estén alineados antes de diseñar la solución detallada. Veamos cómo se desarrolla:

- **Identificación de stakeholders en EcoShop**: Juan (el arquitecto líder) comienza mapeando los stakeholders. Identifica, entre otros, a la Directora de Negocio Digital (owner del canal online), al Gerente de Marketing, al Jefe de Logística, al equipo de TI (desarrollo y operaciones), al departamento de Atención al Cliente, e incluso a un representante de clientes (a través del equipo de experiencia de usuario que recopila feedback de usuarios). Para cada grupo, Juan documenta sus intereses: por ejemplo, Marketing quiere mayor capacidad de personalización de ofertas; Logística, que el sistema gestione mejor los stocks y tiempos de envío; el equipo de TI, que la nueva plataforma sea fácil de mantener e integrar. Este análisis queda plasmado en una **matriz de**

stakeholders de EcoShop, que más adelante servirá para comunicarse de forma dirigida con cada uno.

- **Definición del alcance**: En talleres iniciales, se confirma el **alcance** del proyecto. Se concluye que la arquitectura a desarrollar cubrirá la **plataforma de ventas online completa**, incluyendo el sitio web, la aplicación móvil y todos los sistemas backend directamente relacionados (gestión de productos, carritos, pedidos, pagos, integraciones con inventario y CRM). Se excluyen explícitamente áreas que se tratarán por separado, como el sistema ERP financiero, que aunque conectado, no será rediseñado en este esfuerzo. También se anota la restricción de que la tienda en línea **debe permanecer operativa** durante la transformación (no habrá un apagón total para reemplazarla, sino migraciones graduales), lo cual condicionará las decisiones de implementación más adelante.

- **Elaboración de la visión de arquitectura**: Juan redacta un **Documento de Visión** para la nueva plataforma EcoShop. Incluye una **visión aspiracional**: *"Convertir la plataforma e-commerce de EcoShop en un sistema omnicanal inteligente, capaz de brindar experiencias personalizadas en tiempo real a millones de clientes, soportado por una arquitectura escalable en la nube que facilite la innovación continua."* Este documento describe la situación actual (por ejemplo: *"plataforma monolítica en servidores propios, con tiempos de carga lentos y dificultad para integrar nuevas funciones"*) y la necesidad de cambio (*"competidores ofrecen recomendaciones personalizadas y apps móviles fluidas; EcoShop debe igualar o superar esas capacidades para retener mercado"*). La visión propone a alto nivel soluciones: migrar a microservicios en la nube, implementar un **motor de recomendaciones** basado en IA, habilitar una única vista del cliente unificando datos de navegación, compras e interacciones. Se incluye un **diagrama conceptual** mostrando los componentes planeados: un nuevo frontend web y móvil, una capa de APIs/microservicios (catalogo de productos, servicio de pedidos, servicio de usuarios, pagos, etc.), integraciones con sistemas existentes (ERP, logística), y componentes nuevos como un módulo de análisis de datos y recomendaciones.

- **Beneficios y caso de negocio**: En la visión, Juan destaca beneficios cuantitativos: por ejemplo, *reducción del tiempo promedio de carga de página de 5s a 2s* (lo que incrementaría la tasa de conversión un X%), *aumento de la retención de clientes por personalización* (se espera un +15% en el valor medio por cliente gracias a recomendaciones), reducción de costos demantenimiento al migrar a la nube (evitando inversiones en hardware). También beneficios cualitativos: una mejor experiencia de usuario, mayor agilidad para el negocio al poder añadir funcionalidades rápidamente. Con apoyo de Finanzas, Juan crea un **caso de negocio**: la inversión estimada (en desarrollos, licencias cloud, formación) versus ingresos incrementales proyectados y ahorros operativos. Presenta este caso mostrando que en 3 años el ROI sería positivo, justificando económicamente el proyecto.

- **Riesgos identificados**: El equipo lista riesgos: "*¿Y si la migración a la nube causa interrupciones?*", "*¿y si la nueva plataforma no es adoptada por los usuarios?*", "*¿tenemos el talento interno para construir microservicios?*". Proponen mitigaciones como: realizar pruebas piloto con tráfico controlado, plan de gestión del cambio para usuarios internos, y

considerar contratar o capacitar desarrolladores en tecnologías cloud nativas. Un riesgo notable es el tiempo: la visión establece la meta de 18 meses; si surge retraso, podría impactar la ventana de ventaja competitiva. Se deja claro que la **gerencia acepta** estos riesgos con las mitigaciones propuestas, entendiendo que no actuar también conlleva el riesgo de perder cuota de mercado.

- **Aprobación de la visión**: Juan convoca al Architecture Board de EcoShop (CIO, Directora Digital, etc.) y presenta el Documento de Visión y el plan de trabajo. Hay un debate, por ejemplo: el Gerente de Logística cuestiona si también debería modernizarse la integración con los almacenes; se decide mantenerlo en alcance ya que es crítico para la promesa de entregas rápidas (se ajusta el alcance para incluir optimizaciones en ese punto). Tras afinar detalles, el **comité aprueba la Visión y firma la Declaración de Trabajo de Arquitectura**. En este documento formal, EcoShop se compromete a destinar los recursos necesarios: se asigna un presupuesto significativo y se autoriza dedicar un equipo multidisciplinario al proyecto. Con esta aprobación, la fase A concluye con éxito: hay una visión **compartida**, objetivos claros y luz verde para diseñar la solución en detalle.

Al final de la Fase A, EcoShop cuenta con un **norte claro** (***north start***, concepto aplicado en muchas organizaciones) para su transformación: todos saben qué se pretende lograr (y por qué), existe un entendimiento común entre negocio y TI, y se han sentado las bases (en alcance, valor y apoyo) para avanzar a diseñar las arquitecturas específicas de negocio, datos, aplicaciones y tecnología en las siguientes fases.

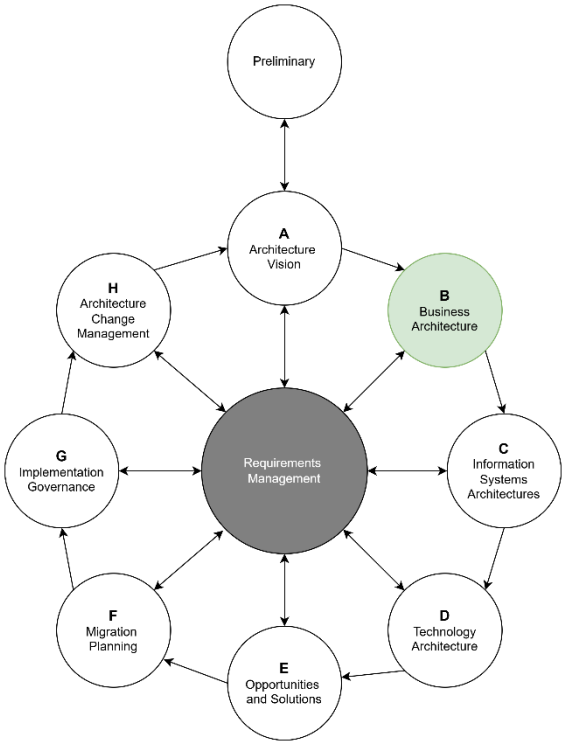
3.2.8 Resultado esperado

Después de la Fase A, se espera que la organización tenga:

- **Una visión de arquitectura aprobada y comunicada** a todos los stakeholders relevantes, alineada con los objetivos de negocio.
- **Compromiso ejecutivo** y de las áreas involucradas, con recursos y presupuesto asignados para continuar.
- **Entendimiento compartido del alcance y valor** del proyecto de arquitectura (qué se hará, qué beneficios traerá).
- **Stakeholders identificados y gestionados**, es decir, cada parte interesada está consciente de la iniciativa y sus expectativas han sido consideradas en la visión.
- **Base para las fases siguientes**: requisitos de alto nivel, supuestos y restricciones bien documentados, que guiarán el trabajo detallado en fases B, C y D.

En resumen, con la Visión establecida, el escenario está listo para "bajar al siguiente nivel de detalle" sabiendo que se trabaja en el **problema correcto con el apoyo adecuado**.

3.3 FASE B – BUSINESS ARCHITECTURE



Tras haber definido qué se quiere lograr con la arquitectura (visión) en la fase anterior, el siguiente paso es concretar cómo el negocio debe funcionar para alcanzar esa visión. **La Fase B: Business Architecture** se enfoca en desarrollar la Arquitectura de Negocio, describiendo tanto la situación actual (baseline) del negocio como la futura (target) en términos de procesos, funciones, organización y capacidades. Esta fase es crucial para **alinear la arquitectura con el negocio**: se traduce la estrategia y la visión en cambios o evoluciones concretas en la forma en que la empresa opera. Un punto importante es que el negocio no vea la arquitectura como algo puramente "de TI"; por el contrario, en la Fase B se involucra fuertemente al negocio, de manera que la arquitectura empresarial se perciba como una **herramienta de transformación** del negocio en sí, y no solo de la tecnología.

3.3.1 Objetivo

Desarrollar una arquitectura de negocio que sirva de base para las posteriores arquitecturas de datos, aplicaciones y tecnología. Esto implica:

- **Describir la situación actual del negocio** (procesos, funciones, estructura organizativa, capacidades, información clave manejada) para entender cómo opera la empresa hoy ("As-Is").
- **Diseñar la situación futura del negocio** que permita lograr los objetivos estratégicos ("To-Be"), definiendo qué cambios en procesos, organización y capacidades son necesarios.

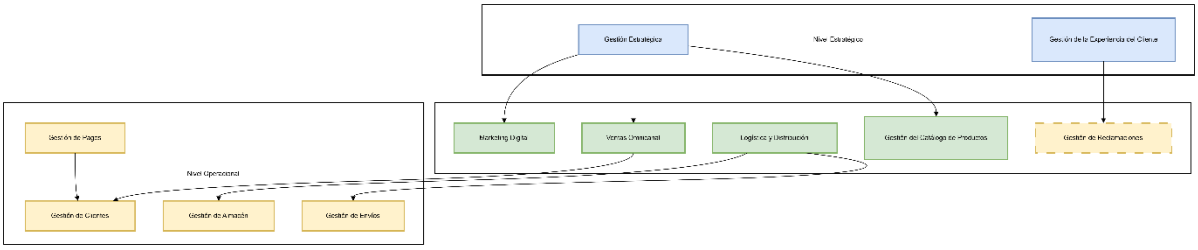
- **Identificar brechas (gaps) entre el estado actual y el estado futuro** del negocio, así como **riesgos y obstáculos** de negocio para el cambio.
- **Especificar requisitos de negocio** que la arquitectura debe cumplir y refinar cualquier requerimiento nuevo descubierto durante el análisis.
- **Asegurar la alineación con la Visión**: que el diseño del negocio futuro corresponde a lo prometido en la fase A y habilita la consecución de los beneficios esperados.

En esencia, la arquitectura de negocio responde a la pregunta: "*¿Cómo debe funcionar el negocio para materializar la estrategia definida?*" y prepara las bases conceptuales para diseñar la arquitectura de sistemas (datos, aplicaciones) que la soportará.

3.3.2 Actividades clave

Durante la Fase B, se realizan las siguientes actividades principales:

- **Seleccionar modelos y técnicas de representación**: Antes de empezar a modelar, el arquitecto de negocio decide qué enfoques y notaciones utilizar para describir el negocio. Por ejemplo, puede usar **mapas de capacidades** de negocio para dar una visión macro de las capacidades actuales y futuras; **diagramas de procesos** (BPMN, flujos de trabajo) para detallar cómo se realizan tareas clave; **modelos de valor** o **value streams** para entender cómo se genera valor al cliente; y estándares de modelado como ArchiMate para representar capas de negocio. También se determina el **nivel de detalle** adecuado: quizás convenga modelar solo procesos de alto nivel, o ir más al detalle en ciertas áreas críticas. Esta decisión depende del alcance, tiempo disponible y madurez de la organización.



Leyenda visual:

- Azul: Capacidades estratégicas actuales
- Verde: Capacidades tácticas actuales
- Amarillo: Capacidades operacionales actuales
- Línea discontinua: Capacidad futura destacada (en este ejemplo: Gestión de Reclamaciones)

- **Desarrollar la arquitectura de negocio actual (Baseline)**: Se recolecta y documenta información de cómo opera el negocio hoy. Esto puede incluir:
    - **Procesos de negocio actuales**: por ejemplo, el flujo "pedido a entrega" en el e-commerce (cómo un pedido pasa desde la compra en la web, procesamiento, envío, hasta la entrega al cliente). Se pueden diagramar los pasos y actores involucrados.

  - **Funciones de negocio y organización**: cuáles son las funciones clave (marketing, ventas, atención al cliente, gestión de inventario, etc.) y cómo se estructuran en unidades o departamentos. Aquí se puede incluir organigramas o descripciones de roles actuales.
  - **Capacidades de negocio existentes**: un capability map identifica qué capacidades tiene la empresa (ej. gestión de catálogo de productos, gestión de almacenes, gestión de pagos, servicio post-venta) y su nivel de madurez o desempeño actual.
  - **Información y recursos clave**: por ejemplo, qué información maneja cada proceso (pedidos, datos de clientes, stock de productos) y qué sistemas o herramientas usa el negocio actualmente para esas funciones (esta parte conecta con la arquitectura de aplicaciones actual, aunque en Fase B se menciona solo a nivel de "qué herramientas emplea el negocio").
  - **Dolores o ineficiencias actuales**: se documentan problemas conocidos en los procesos actuales (cuellos de botella, duplicidades, actividades manuales, etc.).

El objetivo de este análisis es tener claridad *de dónde estamos partiendo*. Muchas veces se descubren incongruencias o áreas no documentadas durante este ejercicio, lo cual ya de por sí agrega valor al dar visibilidad del estado actual.

- **Desarrollar la arquitectura de negocio futura (Target)**: Con la visión estratégica en mente, se diseña cómo debería operar el negocio. Esto abarca:
  - **Nuevos procesos o procesos modificados**: se definen procesos futuros optimizados. Por ejemplo, EcoShop podría diseñar un nuevo proceso de atención omnicanal al cliente, integrando consultas desde web, chat y teléfono en una sola secuencia gestionada por un CRM, a diferencia del proceso actual fragmentado.
  - **Nuevas capacidades requeridas**: se agregan capacidades de negocio que no existen hoy. Por ejemplo, *"personalización de la experiencia"* podría ser una capacidad nueva para EcoShop, que implica poder segmentar clientes y mostrar contenido/productos personalizados. Otra capacidad nueva podría ser *"análisis de comportamiento de clientes en tiempo real"*. Estas capacidades se añaden al mapa de capacidades y se indica que son nuevas o deben fortalecerse.
  - **Cambios organizativos**: si la visión lo requiere, se proponen cambios en la estructura organizativa o en roles. Por ejemplo, tal vez crear un equipo de analítica de datos de clientes dentro de Marketing, o un **nuevo rol de Gestor de Comunidad** para redes sociales, etc., para soportar la estrategia digital.
  - **Mejoras en eficiencia y experiencia**: se rediseñan pasos de procesos para eliminar fricción. Por ejemplo, reducir pasos del checkout, integrar de forma automática la verificación de inventario y cálculo de envío en tiempo real (en lugar de procesos separados).

- **Reglas de negocio y políticas futuras**: se definen nuevas reglas de operación. Por ejemplo, "si un producto está agotado en el almacén principal, automáticamente ofrecer fecha de envío estimada desde otro almacén".

Todas estas definiciones futuras deben alinearse con lograr los objetivos marcados en la Visión. Es útil usar técnicas de diseño de procesos de negocio (BPM) para representar el "To-Be" y compararlo con el "As-Is".

- **Realizar análisis de brechas (gap analysis)**: Con el modelo actual y el modelo futuro del negocio claramente descritos, se identifican las brechas. Esto implica listar las diferencias entre cómo es y cómo deberá ser el negocio. Por ejemplo: "capacidad de ventas omnicanal inexistente actualmente vs. requerida en futuro", "proceso de devoluciones manual actualmente vs. automatizado en futuro". Cada brecha representa algo que deberá abordarse – ya sea mediante cambios en procesos, adquisición de nuevas herramientas, capacitación, etc. Se priorizan brechas según su impacto. También se identifican **implicaciones**: por ejemplo, para lograr cierta capacidad futura se necesitará cierto soporte de sistemas (señal para fases C y D).

- **Revisar y refinar requisitos de negocio**: A partir de las brechas, pueden surgir nuevos **requisitos de negocio** detallados. Por ejemplo, se deriva que "el sistema debe permitir al cliente rastrear su pedido en tiempo real" es un requerimiento de negocio para la nueva plataforma. Todos los requisitos del negocio (funcionales, de rendimiento, regulatorios) se consolidan en el **Catálogo de Requisitos** para ser tenidos en cuenta en fases posteriores.

- **Validar la arquitectura de negocio con stakeholders**: Es fundamental confirmar que el diseño propuesto del negocio futuro tiene sentido y es aceptado. Se realizan workshops o revisiones con dueños de proceso y gerentes de las áreas afectadas. Se discuten los cambios propuestos, verificando viabilidad y ajuste a objetivos. Cualquier retroalimentación se incorpora. Esta validación temprana evita desconexiones entre la teoría arquitectónica y la realidad operativa.

- **Documentar la arquitectura de negocio**: Finalmente, se produce la documentación formal: modelos de procesos, descripciones de capacidades, organigramas futuros, escenarios de negocio, etc., según se requiera. Un entregable típico es la sección de **Arquitectura de Negocio** dentro del **Architecture Definition Document** (Documento de Definición de Arquitectura), la cual se irá construyendo a lo largo de fases B, C y D.

3.3.3 Entradas típicas

Para la Fase B, se consideran elementos tales como:

- **Visión y requisitos de alto nivel de la Fase A**: Los objetivos estratégicos, alcance y principales requerimientos delineados en la Visión guían el diseño del negocio futuro.

- **Modelo de negocio actual existente**: Cualquier documentación previa de procesos, manuales operativos, organigramas, etc., ayudan a armar el baseline. Si no existen, habrá que recabar esa información mediante entrevistas y workshops.
- **Principios de arquitectura relevantes**: En particular los de negocio. Por ejemplo, si un principio definido fue "orientación al cliente", eso influirá en diseñar procesos centrados en la experiencia del cliente.
- **Políticas y reglas de negocio actuales**: Regulaciones, normativas internas, acuerdos de nivel de servicio existentes que no se pueden violar. Por ejemplo, si hay una política de devoluciones que establece ciertos pasos legales, debe respetarse.
- **Datos de desempeño actual**: KPI actuales del negocio (por ejemplo, tasa de conversión, tiempos de entrega promedio, tasa de devoluciones, etc.) para identificar puntos débiles en los procesos actuales y establecer metas de mejora.

3.3.4 Salidas clave (Entregables)

Al finalizar la Fase B, se genera una serie de artefactos que describen cómo evoluciona el negocio:

- **Descripción de la Arquitectura de Negocio Baseline**: Documentación de procesos actuales, capacidades actuales, organigrama vigente relacionado al ámbito del proyecto, etc. Puede incluir modelos de procesos As-Is, diagramas de casos de uso de negocio actuales, etc.
- **Descripción de la Arquitectura de Negocio Target**: Documentación del diseño futuro: procesos To-Be, nuevos flujos de valor, nuevos roles o estructuras, capacidades futuras requeridas y su definición. Es común incluir un Business Capability Model futuro mostrando capacidades añadidas o mejoradas.
- **Matrices de mapeo y catálogos**: Por ejemplo, una matriz que mapee qué procesos actuales se ven impactados por qué componentes futuros, o un catálogo de actividades de negocio. Estos ayudan a trazar la transición.
- **Análisis de brechas de negocio**: Lista de diferencias entre baseline y target en el ámbito de negocio, categorizadas y priorizadas.
- **Requerimientos de negocio actualizados**: Una actualización al catálogo de requisitos incorporando requerimientos refinados y nuevos hallados en esta fase.
- **Impacto en otros dominios identificado**: Un listado de consideraciones para las arquitecturas de datos y aplicaciones; por ejemplo, "se necesita información X en tal proceso futuro, por tanto deberá existir una fuente de datos integrada que hoy no existe" – estos apuntes guiarán la fase C.
- **Aprobación de la arquitectura de negocio**: Aunque no siempre se formaliza por separado, idealmente hay un sign-off (aprobación) de los responsables de negocio para la arquitectura propuesta, antes de proceder.

3.3.5 Roles involucrados

En la Fase B participan principalmente:

- **Arquitecto de Negocio (Business Architect)**: Lidera el desarrollo de la arquitectura de negocio, facilita workshops con áreas de negocio, documenta procesos y capacidades.
- **Representantes de las áreas de negocio involucradas**: Por ejemplo, gerentes/supervisores de Ventas Online, Marketing Digital, Operaciones e-commerce, Atención al Cliente, etc., en el caso de EcoShop. Estos expertos proveen detalle de procesos actuales y validan los diseños futuros.
- **Analistas de negocio o de procesos**: Pueden asistir en mapear procesos, identificar mejoras y documentar reglas de negocio.
- **Arquitecto de Datos / Aplicaciones**: No son protagonistas en esta fase, pero es útil que estén informados o participen en algunas sesiones para empezar a reconocer necesidades que luego se plasmarán en datos y sistemas. Por ejemplo, el arquitecto de datos observando qué datos son clave en los procesos futuros.
- **Patrocinador de negocio**: Un directivo del negocio (ej. la Directora de Negocio Digital en EcoShop) revisa y apoya los cambios propuestos, garantizando alineación estratégica.

3.3.6 Decisiones estratégicas

Algunas decisiones de negocio de alto nivel que suelen tomarse en la Fase B:

- **Definición de nuevas capacidades vs. cambio de modelo de negocio**: Se decide si para alcanzar los objetivos es suficiente con optimizar el modelo actual o si se requiere cambiarlo sustancialmente. En EcoShop, por ejemplo, se pudo haber decidido implementar un modelo de **marketplace** (permitiendo a terceros vender en la plataforma) como cambio de modelo de negocio. Esto tendría implicaciones enormes en procesos y sistemas. Tales decisiones estratégicas se discuten aquí.
- **Prioridad de procesos a transformar**: A veces no se pueden rediseñar todos los procesos a la vez. Se decide cuáles procesos de negocio son críticos de transformar primero. Por ejemplo, ¿es más urgente mejorar el proceso de compra online o el de logística de entregas? Estas prioridades influirán en cómo luego se implementan cambios (fases E/F).
- **Cambios organizacionales significativos**: Si la arquitectura de negocio target implica crear, fusionar o eliminar áreas de la empresa, es una decisión estratégica delicada. En nuestro ejemplo, quizás la creación de un Equipo de Personalización de Experiencia dentro de Marketing es un cambio organizativo decidido en esta fase. Cualquier reestructuración organizativa requiere aprobación y planes de gestión del cambio.
- **Buy-in de áreas de negocio**: Decisiones sobre *cómo* lograr la adopción de los nuevos procesos. Por ejemplo, se decide involucrar desde ahora a ciertos líderes de opinión internos o equipos piloto para asegurar que las áreas de negocio adopten el cambio. Aunque es más un enfoque que decisión puntual, es clave planificarlo estratégicamente.

- **Alineamiento con regulaciones y políticas**: Si el negocio futuro propuesto roza temas regulatorios, aquí se decide cómo abordarlo. Por ejemplo, si se planea usar datos de clientes para personalización, se decide implementar políticas de privacidad estrictas para cumplir GDPR. Son decisiones de cumplimiento que determinan qué es viable o no en el diseño de negocio.

3.3.7 Ejemplo práctico (EcoShop) – Fase B

En EcoShop, la Arquitectura de Negocio determinará cómo debe cambiar la operación de la tienda online para soportar la visión definida. Esto es lo que ocurre durante la Fase B en nuestro caso práctico:

- **Arquitectura de negocio actual de EcoShop**: El arquitecto de negocio (junto con analistas de procesos) levanta la información de cómo opera EcoShop hoy en su canal e-commerce. Documentan el flujo actual de **venta online**: un cliente navega en la web, agrega productos al carrito, realiza la compra; el pedido se registra en el sistema A (propio desarrollado internamente), luego personal de logística extrae manualmente la orden de ese sistema y la ingresa en el sistema de almacén, etc. Se descubre que hay muchos pasos manuales y desconexiones entre sistemas. Por ejemplo, el **proceso de devoluciones** es totalmente manual: el cliente envía un email, un agente debe autorizar, coordinar por separado el envío de vuelta, etc. También se elabora un mapa de capacidades actual: EcoShop actualmente tiene capacidades sólidas en "Gestión de Catálogo de Productos" y "Gestión de Pedidos", pero capacidades débiles o inexistentes en "Personalización de ofertas" o "Inteligencia de clientes". Este análisis muestra claramente dónde el negocio enfrenta limitaciones. Se identifican "dolores": por ejemplo, la tasa de carritos abandonados es alta (60%) posiblemente por fricción en el checkout; la actualización de inventario en el sitio es lenta (se actualiza cada noche, no en tiempo real), causando posibles ventas de productos agotados.
- **Diseño del negocio futuro de EcoShop**: Con los jefes de área, se define cómo debería funcionar EcoShop tras la transformación:
    - Se diseña un **proceso de compra optimizado**: el checkout será en un solo paso, integrando distintas opciones de pago y envío fluidamente; se incluirá la opción de "compra con un clic" para clientes recurrentes (lo que implica guardar pagos seguros).
    - **Proceso de devoluciones automatizado**: en la nueva arquitectura, un cliente podrá solicitar la devolución en línea, generando automáticamente una etiqueta de envío de retorno y actualizando el stock cuando llegue el producto devuelto. Esto elimina varios pasos manuales.
    - **Integración omnicanal**: Se planifica que si en el futuro EcoShop abre tiendas físicas o vende por redes sociales, el proceso y stock sean integrados. Por ende, se agrega la capacidad de "Gestión de inventario omnicanal".
    - **Nuevas capacidades de negocio**: Se añade formalmente la capacidad de "Personalización y recomendaciones" dentro de Marketing: EcoShop definirá cómo

ofrecer recomendaciones de productos en tiempo real basadas en la navegación y compras del cliente. Otra capacidad nueva es "Análisis predictivo de demanda" para que el negocio anticipe qué productos almacenar según patrones de compra (esto involucra tanto Marketing como Operaciones).

- **Organización futura**: Se propone crear un **equipo de Analítica y Personalización** dentro del área digital, responsable de administrar las herramientas de recomendación y segmentación de clientes. También se decide reforzar el equipo de Atención al Cliente con especialistas en chat en vivo, ya que se planea integrar chatbots y agentes en la web para soporte inmediato.

- **Gap analysis de negocio**: Al comparar lo actual vs. lo futuro en EcoShop, se registran brechas como:
  - "No existe un sistema de recomendaciones personalizado" (brecha de capacidad).
  - "El proceso de devoluciones es manual vs. se necesita automático" (brecha de proceso).
  - "Datos de clientes dispersos en varios sistemas vs. se necesita vista unificada del cliente" (brecha de información, que también es entrada para fase de datos).
  - "Equipo de analítica inexistente vs. se necesita equipo especializado" (brecha organizativa).

  Cada brecha se anota con prioridad. Por ejemplo, la capacidad de recomendaciones se considera prioritaria porque impacta directamente ingresos; la reorganización del equipo puede planificarse para más adelante pero debe iniciarse pronto para estar listo al implementar.
- **Requisitos de negocio refinados**: De estas brechas, surgen requisitos concretos: por ejemplo, *"el sistema debe permitir recomendaciones de productos en la página principal y página de producto basadas en comportamiento"*; "debe haber tracking en tiempo real del estado de pedido para los clientes"; *"el proceso de devolución no debe requerir intervención humana salvo excepciones"*. También requisitos no funcionales: *"la tasa de abandono de carrito debe bajar del 60% a <40%"*, *"tiempo de respuesta del servicio de recomendación < 100ms para no ralentizar la carga de página"* (este último es más técnico pero deriva de un objetivo de negocio de rapidez). Todos estos se añaden al catálogo de requisitos.

- **Validación con el negocio**: Juan y el arquitecto de negocio presentan el modelo futuro a los gerentes. La Directora de Negocio Digital está entusiasmada con la capacidad de personalización, ya que competidores lo hacen; también valida que mejorar devoluciones probablemente aumentará la confianza de los clientes. El Gerente de Logística advierte que automatizar devoluciones requerirá coordinar con el operador logístico para integrar sus sistemas – se anota este punto como consideración para la fase de aplicaciones/integraciones. En general, los líderes de EcoShop **aprueban la arquitectura de negocio propuesta** ya que claramente responde a problemas actuales y agrega capacidades alineadas con la estrategia. Un punto de debate es el equipo de Analítica:

¿debe contratarse nuevo personal? Se decide que inicialmente se formará a dos analistas internos, y se considerará contratar más adelante según la carga de trabajo.

- **Documentación de la arquitectura de negocio**: El resultado se plasma en varios documentos y diagramas: un **nuevo mapa de capacidades** que resalta en color las capacidades a desarrollar o mejorar; diagramas BPMN de los nuevos procesos clave (compra, devoluciones, atención al cliente), comparados lado a lado con los diagramas actuales para ver la mejora; descripciones de roles nuevos y cambios organizativos. Estas salidas se guardan en el repositorio de arquitectura. Se actualiza asimismo la sección de business architecture del documento global de arquitectura con toda esta información.

Con la Fase B concluida, EcoShop ha definido **cómo deberá operar el negocio** para sostener la plataforma e-commerce renovada. Tiene claridad en qué cambios de procesos y capacidades son necesarios, y esta comprensión de negocio guiará las decisiones de diseño de sistemas en las próximas fases. Además, los líderes de negocio están involucrados y comprometidos, pues ven reflejadas sus necesidades en el diseño futuro.

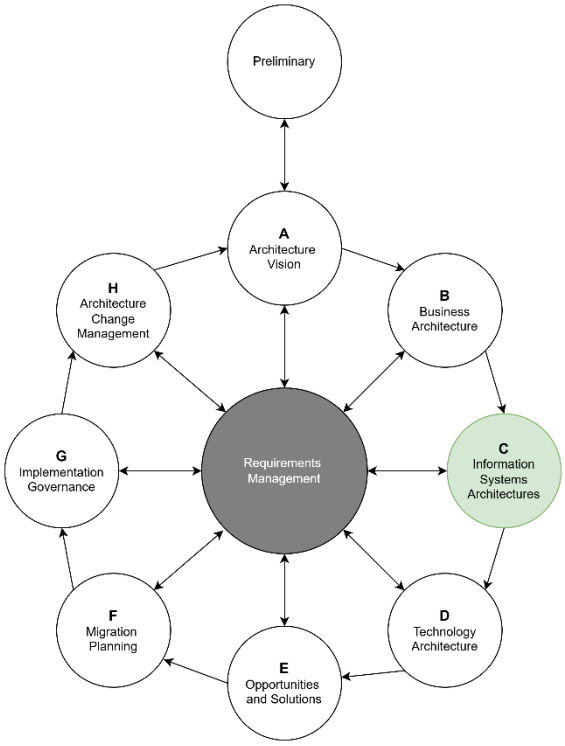
3.3.8 Resultado esperado

Al terminar la Fase B, EcoShop (o cualquier organización) debería contar con:

- Una **descripción completa del negocio actual vs. futuro** en el ámbito del proyecto, evidenciando claramente las diferencias.
- **Procesos de negocio definidos** para el estado objetivo, listos para servir como requisitos para el diseño de sistemas e implementación.
- **Nuevas capacidades identificadas** que serán habilitadas (queda claro "qué hará el negocio" una vez implementada la arquitectura).
- Un **listado de brechas** de negocio priorizadas, que servirá para planificar cómo cerrarlas con soluciones (dando entrada a fases E y F posteriormente).
- **Requisitos de negocio bien documentados** y validados, que se transportarán a las siguientes fases asegurando que las arquitecturas de TI los satisfagan.
- **Alineamiento con stakeholders de negocio**: aprobación o entendimiento por parte de las áreas afectadas, lo cual facilita la adopción posterior.

En resumen, la Fase B entrega un **mapa de ruta del negocio**: define hacia dónde debe moverse el negocio y qué necesita, de forma que los esfuerzos en datos, aplicaciones y tecnología se enfoquen en habilitar precisamente eso.

3.4 Fase C – Information Systems Architecture (Datos y Aplicaciones)



La **Fase C** del ADM abarca el desarrollo de la Arquitectura de Sistemas de Información, que incluye dos aspectos fundamentales: la **Arquitectura de Datos** y la **Arquitectura de Aplicaciones**. TOGAF permite abordar estos dos dominios en conjunto o por separado (Fase C puede dividirse en C1 Datos y C2 Aplicaciones), pero para efectos prácticos suele integrarse el trabajo dado lo estrechamente relacionados que están. En esta fase, el objetivo es traducir las necesidades del negocio (definidas en Fase B) en diseños lógicos de **cómo los datos se organizan y fluyen y qué aplicaciones o sistemas de software** se requieren para soportar los procesos y capacidades futuras. Es, en cierto modo, pasar de la perspectiva "qué hace el negocio" a "qué deben hacer los sistemas y cómo manejan la información".

Dado que son dos sub-disciplinas, estructuraremos esta fase en dos partes: **C1: Arquitectura de Datos** y **C2: Arquitectura de Aplicaciones**, manteniendo la coherencia entre ambas.

3.4.1 Sobre la Fase C y su evolución en TOGAF

La **Fase C del ADM** se centra en el desarrollo de la **Arquitectura de Sistemas de Información**, la cual comprende dos dominios fundamentales: **Arquitectura de Datos (C1)** y **Arquitectura de Aplicaciones (C2)**. TOGAF permite abordarlos de forma conjunta o separada, y aunque están estrechamente relacionados, a efectos prácticos suele ser conveniente tratarlos de manera integrada. El propósito de esta fase es claro: traducir las necesidades del negocio, definidas previamente en la Fase B, en **diseños lógicos** que especifiquen cómo deben organizarse y fluir los datos, y qué aplicaciones o sistemas serán necesarios para dar soporte a los procesos y capacidades futuras.

Es, en definitiva, un tránsito desde el "qué hace el negocio" hacia el "cómo deben responder los sistemas y qué información deben manejar".

Ahora bien, **soy plenamente consciente de que con la aparición de TOGAF 10 se ha abierto la puerta a nuevos dominios arquitectónicos**, como la Arquitectura de Inteligencia Artificial, que no encajan de forma estrictamente limpia dentro de "Datos" o "Aplicaciones". Y eso está bien. No debemos llevarnos las manos a la cabeza por ello. La realidad es que la mayoría de las disciplinas emergentes —ya sea IA, automatización, robótica o digital twins— **siguen apoyándose, en última instancia, en datos estructurados y/o aplicaciones que los procesan**. Por tanto, más que romper el modelo, **lo extienden o lo especializan**.

Lo importante, como arquitectos, es **mantener la coherencia y trazabilidad** entre estas dimensiones, sin perdernos en etiquetas o modas. Lo esencial sigue siendo diseñar soluciones que conecten propósito de negocio con capacidades tecnológicas.

3.4.2 C1: Arquitectura de Datos

Objetivo

Desarrollar una Arquitectura de Datos que soporte los requerimientos de información del negocio futuro, asegurando integridad, accesibilidad, calidad y seguridad de los datos. Esto implica:

- **Catalogar y entender los datos actuales** (qué datos existen, dónde residen, quién los usa) – Arquitectura de Datos baseline.
- **Diseñar la estructura de datos futura** (modelos de datos conceptual/logico, fuentes maestras de datos, integraciones, flujos de datos) – Arquitectura de Datos target.
- **Identificar brechas de datos**: datos nuevos que se necesitarán, datos redundantes o de baja calidad que se deben mejorar, etc.
- **Definir cómo se gobernarán los datos**: lineamientos de calidad, seguridad, propiedad de los datos en la nueva arquitectura.
- **Asegurar alineamiento con principios de datos**: por ejemplo, si un principio es “dato único y consistente”, diseñar acorde a ello.

En resumen, la Arquitectura de Datos especifica *qué datos se requieren y cómo estarán organizados* para habilitar al negocio.

Actividades clave

- **Inventario y modelado de datos actuales**: Se identifican las **fuentes de datos** actuales relevantes. Por ejemplo, en EcoShop: base de datos de productos, base de datos de pedidos, hojas de cálculo con información de clientes (si no había CRM formal), etc. Se documenta el **modelo de datos** actual a alto nivel – entidades principales y sus relaciones (Clientes, Pedidos, Productos, Inventario, Pagos, etc.), y se señala dónde residen esos datos (en qué sistemas o bases de datos). También se evalúa la **calidad** de datos actual:

por ejemplo, ¿hay duplicados de registros de clientes? ¿datos incompletos? Esto se anota para ser abordado si es crítico.

- **Requerimientos de datos futuros**: A partir de la arquitectura de negocio futura y los requisitos, se determinan las necesidades de datos. Por ejemplo, para personalización se requerirán datos de navegación del cliente, datos de historial de compras, preferencias, etc., que quizá no se almacenaban antes. Se lista qué **nuevos datos** serán necesarios y qué volumen o frecuencia de actualización se espera (p. ej., "datos de comportamiento de usuario en tiempo real, provenientes del sitio web y app"). También se consideran requisitos como retención de datos (¿cuánto tiempo almacenar?), normativas (p.ej., GDPR exige anonimización de ciertos datos).

- Diseño de modelo de datos target (conceptual/lógico): Se crea un **modelo de datos conceptual** para la arquitectura objetivo, incorporando las nuevas entidades y relaciones necesarias. Por ejemplo, EcoShop podría tener entidades como *Cliente, Pedido, Producto, Carrito, Recomendación, VisitaWeb, etc.*, definidas en un diagrama ER o en ArchiMate. Se definen las **fuentes maestras**: por ejemplo, Cliente tendrá una "single source of truth" en una base de datos unificada de clientes (quizá un CRM o base de datos central). Producto puede mantenerse en un catálogo maestro. Es clave definir dónde estará cada tipo de dato en la arquitectura target para evitar duplicaciones y asegurar consistencia. También se diseña a nivel lógico cómo fluirán datos entre aplicaciones (esto se solapa con la parte de aplicaciones).

- **Arquitectura de almacenamiento y acceso a datos**: Aquí se decide el **tipo de almacenamiento** adecuado para cada conjunto de datos en la nueva arquitectura. Por ejemplo: ¿Se usará una base de datos relacional central para pedidos y clientes? ¿Una base NoSQL para almacenar sesiones de navegación? ¿Un data lake para los datos analíticos? En EcoShop, quizás se decida implementar un **data warehouse** o **data lake** donde consolidar toda la información de interacción de clientes para análisis avanzado. Se diseña la arquitectura de datos incluyendo repositorios, lagos, bases transaccionales, caches, etc., y cómo se replicarán o sincronizarán datos entre ellos. Esto también toca decisiones de tecnología específicas (fase D profundizará), pero conceptualmente en fase C se traza el panorama.

- **Definir políticas de datos (seguridad, calidad)**: Se incorporan lineamientos sobre cómo se tratarán los datos en la nueva arquitectura:

    - **Seguridad de datos**: clasificación de datos sensibles (p. ej., datos personales de clientes, datos de tarjetas de pago – estos requieren cifrado y cumplimiento PCI), controles de acceso (quién/qué aplicaciones pueden ver qué datos), etc.

    - **Calidad de datos**: asegurar que datos críticos (por ejemplo direcciones de envío) tengan validaciones, que se establezcan procesos de limpieza de datos duplicados, etc. Puede definirse que se implementará un *Data Governance* con responsables de calidad para los datos maestros.

  - **Gobernanza de datos**: establecer propietarios para conjuntos de datos (por ejemplo, el área de Marketing es dueña de los datos de clientes, TI es dueño de datos de sistemas, etc.), y reglas para introducir cambios en el modelo de datos.

- **Analizar brechas de datos**: Comparando la situación actual con la futura, se identifican brechas como: datos que no existen actualmente, pero serán necesarios; datos que existen en múltiples lugares y deberán unificarse; problemas actuales que hay que resolver (por ejemplo, hoy no se capturan ciertos datos de clientes que se necesitarán, así que habrá que empezar a capturarlos). Cada brecha se tomará luego como input para planificación (quizá algunas implican proyectos de migración de datos).

- **Documentar la Arquitectura de Datos**: Se produce la documentación correspondiente: modelos de datos, diccionarios de datos (descripciones de cada entidad/atributo), diagramas de flujo de datos (que muestren cómo se moverán datos entre aplicaciones, a veces esto se denomina **diagrama de fuentes de datos a consumidores**). Esto se integra en el Architecture Definition Document bajo la sección de Data Architecture.

3.4.3 C2: Arquitectura de Aplicaciones

Objetivo

Desarrollar una Arquitectura de Aplicaciones que describa los **sistemas de software y componentes de aplicación** necesarios para implementar los procesos de negocio y gestionar los datos definidos en fases previas. Se enfoca en:

- **Inventariar las aplicaciones actuales** y su estructura – Arquitectura de Aplicaciones baseline.

- **Diseñar el panorama de aplicaciones futuro** (qué aplicaciones nuevas se requerirán, cuáles cambiarán o serán eliminadas, y cómo interactuarán entre sí) – Arquitectura de Aplicaciones target.

- **Definir las funcionalidades de cada aplicación** y las interdependencias (integraciones, interfaces) entre aplicaciones en la solución futura.

- **Asegurar que las aplicaciones futuras cumplen los requerimientos de negocio** (funcionales y no funcionales) y se alinean con los principios (por ejemplo, principio de "aplicaciones modulares").

- **Identificar brechas y plan de transición**: qué hacer con las aplicaciones existentes (modernizar, reemplazar, retirar) para llegar al estado futuro.

En síntesis, se diseña la "arquitectura de software" que soportará al negocio, definiendo los roles de cada sistema y cómo cooperan.

Actividades clave

- **Catalogar la Arquitectura de Aplicaciones actual**: Se elabora un **Catálogo de Aplicaciones** existente que soporte los procesos actuales. Por ejemplo, en EcoShop:
    - Sistema de comercio electrónico actual (por ejemplo, una plataforma custom o un paquete comercial) – que abarca catálogo, carrito, pago.
    - Sistema de gestión de almacenes (WMS) – quizás un software separado usado por Logística.
    - Sistema de gestión de clientes (¿hay CRM? supongamos que actualmente usan base de datos o sistemas ad-hoc).
    - Herramientas satélite (hojas Excel para ciertos reportes, sistema de email marketing de terceros, etc.).

  Para cada aplicación, se documenta su propósito, tecnologías, usuarios, y relaciones/integraciones (p. ej., la plataforma e-commerce actual envía pedidos al ERP manualmente, etc.). Este catálogo da una vista de qué tenemos hoy.

Nombre de la Aplicación	Función Principal	Tecnología	Dueño Funcional	Problemas Conocidos
CRM Cloud	Gestión de clientes y contactos	Dynamics 365	Dirección Comercial	Lentitud en cargas; integración limitada con eCommerce
ERP Central	Contabilidad y gestión financiera	SAP ECC	Departamento Financiero	Altos costes de mantenimiento; interfaz poco amigable
EcoShop Web	Portal de ventas online	React + Node.js	Departamento de Ventas	No responsive en móviles antiguos; caídas ocasionales
Gestor de Pedidos	Procesamiento de órdenes	.NET	Logística	Lógica de negocio duplicada; difícil de escalar
BI Manager	Reporting y dashboards	Power BI + Azure SQL	Dirección General	Falta de datos en tiempo real; bajo conocimiento de uso interno
ReclamApp	Gestión de quejas y devoluciones	PHP Legacy	Atención al Cliente	Código obsoleto; sin trazabilidad integrada
Mobile EcoShop	App móvil para usuarios finales	Flutter	Marketing Digital	Bugs en Android 13; falta de métricas de uso

Opcionalmente puedes añadir columnas como *Prioridad de Reemplazo*, *Nivel de Riesgo* o *Integraciones Clave* si lo necesitas para una evaluación más profunda.

- **Evaluar aplicaciones existentes**: Además de listarlas, se evalúa su estado: ¿Cuál es su **idoneidad** para los requisitos futuros? Por ejemplo, la plataforma actual de EcoShop quizás es monolítica y difícil de escalar, lo que se anota como deficiencia. Tal evaluación puede ser informal o usando una **matriz de evaluación** (p. ej., valorar cada aplicación en funcionalidad, escalabilidad, costo, etc.). Esto ayudará a decidir qué aplicaciones se conservarán o no.

Aplicación	Funcionalidad (1-5)	Escalabilidad (1-5)	Coste Total (1-5)	Alineación con Requisitos Futuros (1-5)	Notas/Observaciones	Recomendación
CRM Cloud	4	3	3	3	Falta integración con eCommerce	Mejorar integración
ERP Central	3	2	2	2	Sistema rígido, alto coste de mantenimiento	Evaluar migración a S/4HANA
EcoShop Web	3	2	4	2	Monolítica, difícil de escalar	Rearquitecturar a microservicios
Gestor de Pedidos	3	2	4	2	Lógica duplicada, falta de desacoplamiento	Refactorizar parcialmente
BI Manager	4	3	3	4	Interfaz intuitiva, requiere mejor tiempo real	Consolidar y ampliar
ReclamApp	2	1	5	1	Sistema obsoleto, sin trazabilidad	Sustituir
Mobile EcoShop	4	3	3	4	Fallos menores en Android, buena percepción de usuarios	

Escalas:

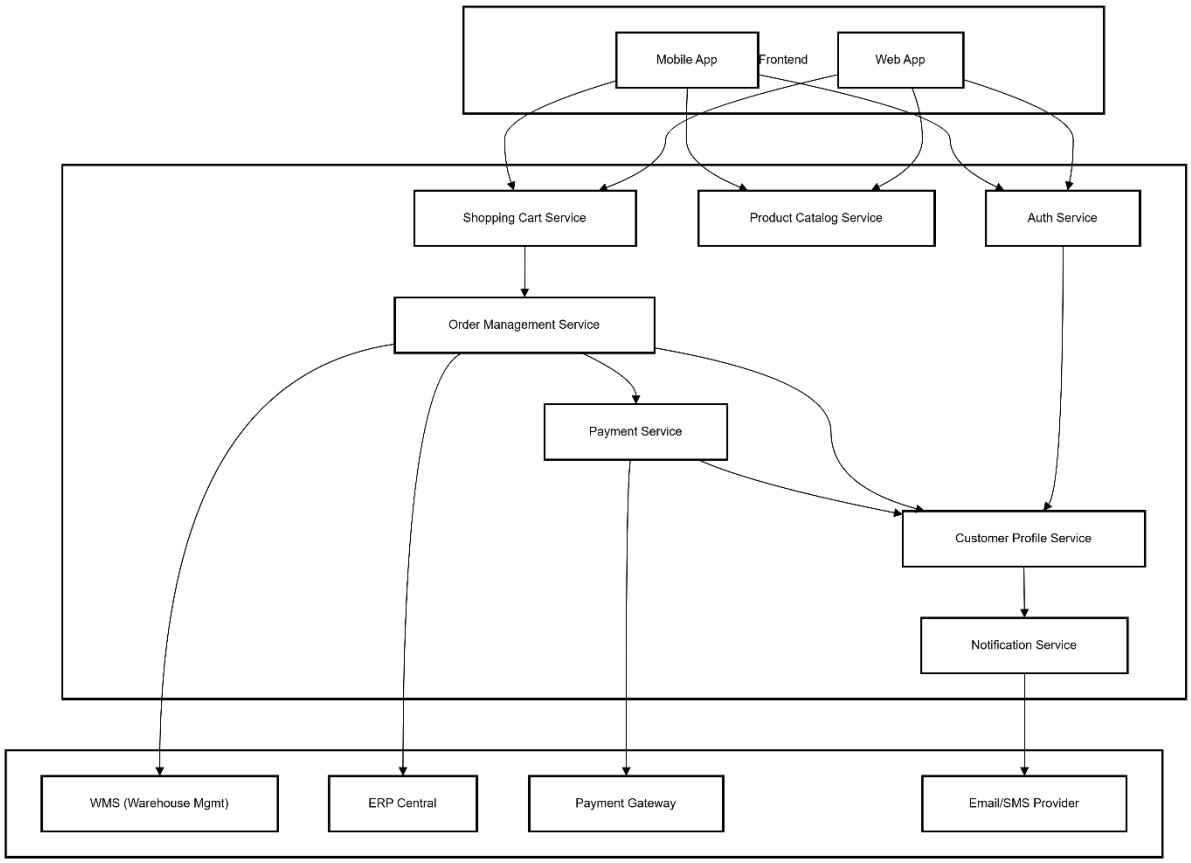
- 1 = Muy deficiente / Alto coste / Nada alineado
- 5 = Excelente / Bajo coste / Altamente alineado

Recomendación: Puedes usar colores tipo semáforo en Excel (verde-amarillo-rojo) o aplicar fórmulas de puntuación ponderada.

- **Diseñar la Arquitectura de Aplicaciones futura**: Se define qué aplicaciones compondrán la solución final. Aquí surgen típicamente decisiones como:
    - **¿Comprar o desarrollar?** (make or buy): Por cada gran componente, decidir si se usará un producto comercial o desarrollo propio. Ej: "Para la nueva plataforma web, ¿usamos un paquete e-commerce existente o desarrollamos a medida microservicios?".
    - **Lista de nuevas aplicaciones** a implementar: Por ejemplo, EcoShop decide implementar un **CRM** moderno para gestionar clientes, un **motor de recomendaciones** (quizás un servicio cloud de IA o un módulo desarrollado, observais como no es necesario el dominio del arquitectura de IA), una aplicación

móvil nativa, etc. También unificarán ciertas cosas, por ejemplo retirar las hojas Excel de marketing e integrar todo en el CRM.

- **Arquitectura modular de aplicaciones**: Se define la estructura lógica: por ejemplo, **una plataforma e-commerce basada en microservicios**. Esto implicaría subdividir la aplicación en componentes: Servicio de Catálogo, Servicio de Carrito/Pedido, Servicio de Pagos, Servicio de Recomendaciones, etc., cada uno quizás desplegable por separado. Alternativamente, si no van por microservicios, tal vez adoptan un paquete SaaS integrado. En nuestro caso supongamos microservicios.
- **Mapa de Interacciones**: Se diseña cómo se comunicarán las aplicaciones. Por ejemplo: la aplicación web y la app móvil hablarán con los microservicios vía APIs REST; los microservicios entre sí compartirán datos a través de una cola de mensajería o API gateway; el sistema de almacén (WMS) se integrará vía APIs también con el servicio de pedidos para actualizar stocks; el CRM intercambiará datos de clientes con la base de datos principal mediante integraciones batch o streaming. Un **diagrama de aplicaciones** ayuda a visualizar esto, mostrando cada aplicación/servicio y líneas de integración entre ellos.



- **Asignación de funcionalidad a aplicaciones**: Se clarifica qué función del negocio se cubre con qué aplicación. Por ejemplo: gestión de catálogo la hará el

Servicio de Catálogo; la experiencia de navegación la cubre la Aplicación Web (frontend) y la App móvil; la personalización la hará el Motor de Recomendaciones; la analítica de clientes se hará en un nuevo módulo de BI vinculado al data warehouse. Esta asignación asegura que todas las actividades definidas en procesos de negocio tengan un soporte de sistema.

- **Considerar interoperabilidad y estándares**: Se deciden también estándares de integración: en nuestro ejemplo, usar APIs REST/JSON, posiblemente eventos (arquitectura orientada a eventos) para ciertas comunicaciones (p. ej., "pedido creado" notifica a otros componentes). Si se integrará con terceros (pasarela de pagos, servicio de emails), se define cómo (normalmente via APIs o SDKs). También se aplica el principio de interoperabilidad: asegurar que las nuevas apps puedan intercambiar datos fácilmente, quizás definiendo un modelo de datos común para integraciones.

- **Evaluar impacto en aplicaciones existentes**: Se determina qué aplicaciones actuales seguirán en la arquitectura futura y cuáles no. Por ejemplo, si EcoShop tiene un ERP financiero, quizás ese se queda pero se integra de otro modo; la plataforma e-commerce vieja posiblemente será reemplazada en su totalidad. El WMS (sistema de almacén) externo quizás no se cambia pero se conecta via API. Se produce un **plan de transición de aplicaciones**: por ejemplo, "Aplicación X será retirada en la fase 1, Aplicación Y será reemplazada en fase 2, Aplicación Z (nueva) introducida en fase 1", etc., para luego planificarlo.

- **Definir requerimientos de aplicaciones**: A partir del diseño, se pueden detallar **requisitos técnicos** para las aplicaciones: capacidad de usuarios concurrentes, tiempos de respuesta (p. ej., "el microservicio de catálogo debe responder en <200ms"), requerimientos de disponibilidad ("99.9% uptime"), consideraciones de usabilidad para el frontend (diseño responsivo, ADA compliance, etc.). Estos complementan los requisitos de negocio en el catálogo de requisitos.

- **Analizar brechas de aplicaciones**: Se enumeran brechas como: aplicaciones nuevas que no existen hoy (ej. "no hay CRM actual, se deberá implementar uno"); funcionalidades que actualmente ninguna aplicación cumple (p. ej., "no hay sistema de recomendaciones, habrá que desarrollarlo o adquirirlo"); integraciones que no existen ("WMS no conectado, se deberá integrar"). También puede haber brechas en cuanto a habilidades: "no hay experiencia en desarrollar microservicios, se requerirá capacitar o contratar" – esto si bien no es aplicación en sí, es gap para la implementación. Estas brechas ayudan a dimensionar esfuerzo y se llevarán a fases E/F.

- **Documentar la Arquitectura de Aplicaciones**: Se completa la documentación: el **Catálogo de Aplicaciones target**, con descripciones de cada sistema planeado; **diagramas de aplicaciones** ilustrando la estructura y las dependencias; especificaciones de interfaces (aunque los detalles pueden dejarse a diseños posteriores, aquí al menos se define qué integraciones deben existir). Todo se consolida en la sección de Application Architecture del documento de arquitectura.

3.4.4 Entradas típicas: Datos & Aplicaciones

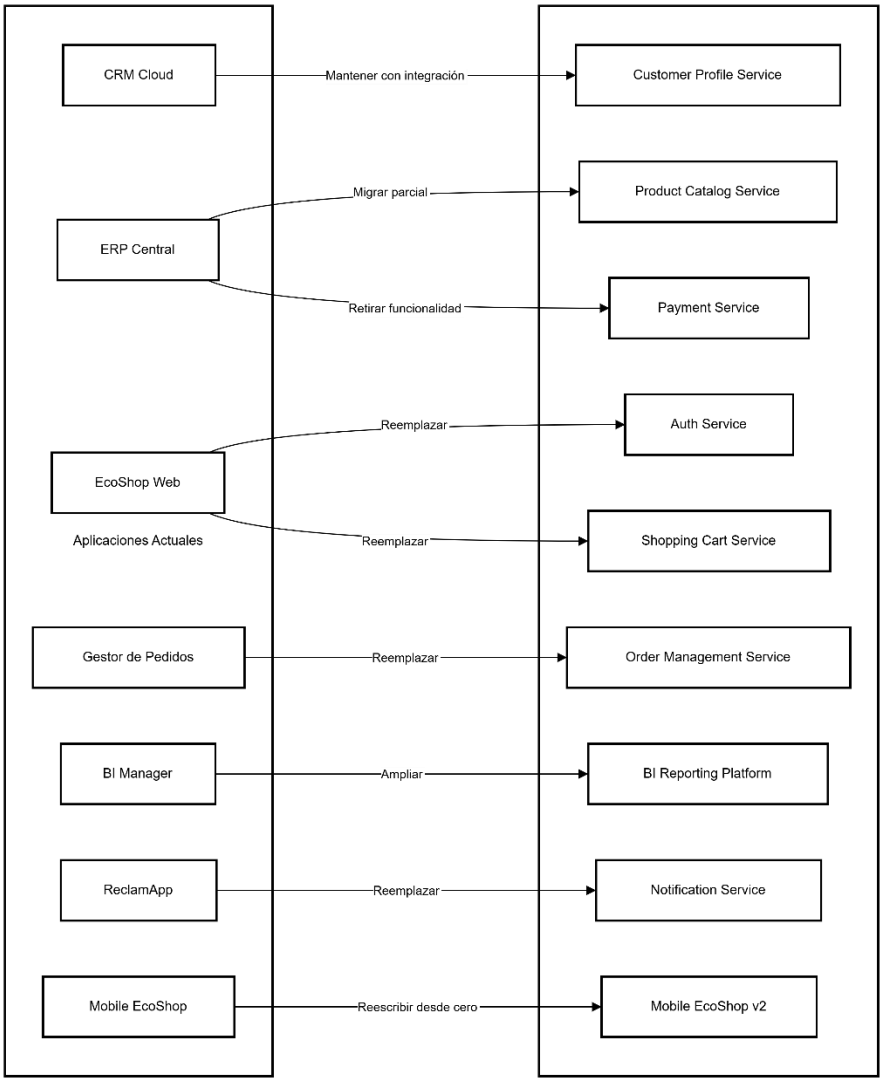
- **Arquitectura de Negocio target y requerimientos**: Esta es la principal guía: los procesos y capacidades futuras dictan qué sistemas y datos se necesitan. Por ejemplo, si se definió capacidad de personalización, eso genera requerimiento de un sistema (o funcionalidad) de recomendaciones y datos de comportamiento.
- **Arquitectura de Datos/Aplicaciones actual**: Los catálogos de sistemas actuales y modelos de datos actuales sirven como base para ver qué se aprovecha y qué se cambia.
- **Principios de datos y aplicaciones**: Por ejemplo, "datos compartidos antes que aislados" influye en diseñar un repositorio común en lugar de bases aisladas, "aplicaciones deben ser modulares" empuja a una arquitectura por componentes desacoplados, etc.
- **Estrategias corporativas de TI**: Si la empresa tiene lineamientos como "preferir soluciones SaaS" o "cloud-first" (de la fase preliminar), estos lineamientos se aplican ahora en las decisiones.
- **Cualquier modelado de información existente**: A veces hay modelos de datos corporativos, diccionarios, o ontologías ya definidas, que se deben considerar para no reinventar definiciones (por ejemplo, definiciones de "cliente", "producto" ya establecidas).

3.4.5 Salidas clave (Entregables): Datos & Aplicaciones

- **Modelo de Datos Baseline y Target**: Diagramas o descripciones de las estructuras de datos actuales y futuras. El modelo futuro conceptual/lógico de datos suele ser un entregable importante, indicando entidades maestras, relaciones y cardinalidades clave.
- **Diccionario o Catálogo de Datos**: Lista de entidades de datos (y posiblemente atributos principales) que serán gestionados, con su definición. Por ejemplo, definición de qué es "Cliente" en este contexto, qué atributos lo componen (nombre, email, histórico de compras, etc.).
- **Arquitectura de Datos – Diagramas de flujo**: Por ejemplo, diagramas que muestren cómo los datos fluyen del sitio web hacia la base de datos, al data lake, al sistema de analítica, etc., en la solución target.
- **Catálogo de Aplicaciones actual y futuro**: Un listado de todas las aplicaciones involucradas hoy y las planeadas a futuro, con su descripción y estado (nueva, modificar, eliminar, retenida). A menudo se presenta en forma tabular con columnas como Nombre de aplicación, Función, Tecnología, Propietario, Estatus (a retirar/retener/nueva).

Nueva Aplicación / Servicio	Función Principal	Tecnología Objetivo	Sistema Actual Vinculado	Destino del Sistema Actual
Auth Service	Gestión de autenticación y acceso	.NET Core + IdentityServer	Integrado en EcoShop Web	Reemplazar
Product Catalog Service	Gestión del catálogo de productos	Node.js + MongoDB	ERP Central	Migrar funcionalidad parcial
Shopping Cart Service	Carrito de compra	Java + Redis	EcoShop Web	Reemplazar

Order Management Service	Procesamiento de pedidos	Spring Boot + PostgreSQL	Gestor de Pedidos	Reemplazar
Payment Service	Integración con pasarelas de pago	Python + Stripe API	Integrado en ERP Central	Retirar funcionalidad
Notification Service	Envío de alertas y comunicaciones	Azure Functions + SendGrid	ReclamApp	Reemplazar
Customer Profile Service	Gestión de datos de clientes	.NET Core + SQL Azure	CRM Cloud	Mantener con integración
BI Reporting Platform	Informes y dashboards	Power BI + Azure Synapse	BI Manager	Ampliar
Mobile EcoShop v2	App móvil renovada	Flutter + GraphQL	Mobile EcoShop	Reescribir desde cero



- **Diagramas de Arquitectura de Aplicaciones**: Representaciones gráficas de la disposición de los sistemas en la arquitectura futura y sus interacciones (por ejemplo, un diagrama de componentes o de despliegue lógico).
- **Especificación de Interfaces / Integraciones**: Lista de las integraciones requeridas entre aplicaciones (puede ser una matriz de aplicaciones x integraciones detallando qué información pasa entre qué sistemas).

- **Requisitos técnicos actualizados**: El catálogo de requisitos se complementa con requisitos técnicos para datos y aplicaciones surgidos (ej. requerimientos de capacidad, performance, seguridad de aplicaciones).
- **Análisis de brechas (Datos & Apps)**: Documento o tabla enumerando las brechas detectadas en datos y aplicaciones (nuevos sistemas a implementar, sistemas a retirar, necesidades de migración de datos, etc.) que pasarán a considerarse en la planificación de implementación.

3.4.6 Roles involucrados: Datos & Aplicaciones

- **Arquitecto de Datos**: Responsable de modelar los datos, identificar fuentes maestras, diseñar la estrategia de datos (bases de datos, almacenamiento, calidad).
- **Arquitecto de Aplicaciones / Soluciones**: Responsable de diseñar la arquitectura de aplicaciones target, decidir la modularización, integraciones y tecnologías a alto nivel. Aquí se puede ayudar con arquitectos de IA que estarían bajo la supervisión
- **Arquitecto de Aplicaciones / Soluciones**: Responsable de diseñar la arquitectura target de aplicaciones, incluyendo la definición de módulos, patrones de integración, tecnologías a alto nivel y alineación con los objetivos del negocio. Este rol lidera la visión funcional y técnica del ecosistema de aplicaciones y toma decisiones estratégicas sobre interoperabilidad, desacoplamiento y evolución tecnológica. En algunos casos, puede contar con el **apoyo de arquitectos especializados** —como los de Inteligencia Artificial, datos o movilidad— que operan bajo su supervisión. Estos perfiles, más técnicos y focalizados, aportan profundidad en sus áreas respectivas, pero se subordinan a la arquitectura global definida por el Arquitecto de Aplicaciones / Soluciones, quien actúa como figura de referencia transversal en la toma de decisiones.
- **Expertos en sistemas actuales**: Por ejemplo, el desarrollador principal de la plataforma e-commerce actual, el administrador de bases de datos, etc., para obtener información exacta del baseline y restricciones técnicas.
- **Usuarios clave o analistas de negocio**: Pueden aportar en validar que los datos identificados y las funcionalidades asignadas a aplicaciones cubrirán los requerimientos del negocio. Por ejemplo, confirmando que en la nueva solución no se olvide ninguna funcionalidad que hoy es crítica (a veces, funcionalidades pequeñas de sistemas legados pueden pasar inadvertidas si no se consulta con usuarios).
- **Equipo de seguridad de la información**: Para asegurar que los diseños contemplen las políticas de seguridad (en datos especialmente).

3.4.7 Decisiones estratégicas: Datos & Aplicaciones

- **Estrategia de plataformas: build vs buy**: Una de las decisiones más estratégicas aquí es qué componentes serán desarrollos a medida vs qué se adoptará como producto comercial. En EcoShop, por ejemplo, se decide no desarrollar un CRM propio sino comprar una solución SaaS reconocida para el manejo de clientes, acelerando esa

capacidad. En cambio, deciden desarrollar internamente la plataforma de venta online con microservicios porque quieren diferenciarse y tener control total sobre la experiencia. Estas elecciones afectan costos, tiempos y dependerán de la estrategia de la empresa (diferenciación vs estandarización).

- **Estándares tecnológicos base**: Aunque los detalles de tecnología se afinan en Fase D, aquí se suele ya decidir líneas generales: ¿será una arquitectura cloud? ¿se usará arquitectura orientada a servicios (SOA) o microservicios? ¿Qué lenguaje/plataforma principal se usará para desarrollar (Java, .NET, Python, etc.)? Estas decisiones estratégicas dan forma a la solución. En EcoShop, se pudo decidir por ejemplo "iremos con soluciones cloud de Azure para la mayoría de componentes" o "usaremos microservicios en contenedores Docker". Tales decisiones pueden venir de principios (cloud-first) o de preferencia por el ecosistema existente de la empresa.
- **Destino de aplicaciones existentes (retirar, retener, reemplazar)**: Se toman decisiones sobre el futuro de cada sistema actual. Esto es estratégico porque puede implicar cambios grandes: por ejemplo, "retirar la plataforma actual completamente en el go-live de la nueva" vs "mantener ambas en paralelo durante una transición más larga". Decidir retirar un sistema legado puede tener impactos de personal (quizá ciertas personas que mantenían el sistema necesitarán reubicación), e impactos de riesgo (reemplazo total vs incremental). En EcoShop, probablemente deciden reemplazar la antigua plataforma e-commerce por la nueva gradualmente; retener el WMS existente pero integrarlo; incorporar un nuevo CRM y apagar los sistemas manuales de clientes.
- **Estrategia de datos centralizados vs distribuidos**: Decisiones como "¿Tener un repositorio de datos central (ej. data lake) unificado o permitir múltiples fuentes integradas?" son estratégicas. EcoShop puede decidir crear un Data Lake central que consolide datos de clientes, ventas, navegación, etc., para análisis global. O bien, podría decidir mantener sistemas separados pero federados con integraciones. Estas decisiones impactan la arquitectura en robustez y complejidad.
- **Políticas de seguridad y privacidad**: Se determina el alineamiento con estándares de seguridad corporativa. Por ejemplo, "adoptaremos una política Zero Trust en aplicaciones – cada servicio autenticará y autorizará cada petición" o "los datos personales estarán tokenizados en base de datos y solo servicios autorizados podrán detokenizar". Estas son decisiones de alto nivel que guiarán implementaciones.
- **Prioridad de implementación de componentes**: A veces, al diseñar, se identifica qué componentes son core y cuáles complementarios. Estrategicamente se puede decidir que ciertos módulos se implementen primero porque desbloquean otros. Por ejemplo, decidir "el servicio de catálogo y la base de datos de productos serán los primeros construidos, ya que todo el sitio depende de eso", es una priorización estratégica que luego alimentará la planificación.

3.4.8 Ejemplo práctico (EcoShop) – Fase C

Continuando con EcoShop, veremos cómo aplican esta fase para definir datos y aplicaciones de su futura plataforma:

- **Datos actuales de EcoShop**: El arquitecto de datos de EcoShop revisa dónde residen los datos hoy: Los **datos de productos** están en una base de datos MySQL del sistema e-commerce actual; los **datos de pedidos** también en MySQL pero con algunos detalles duplicados en el ERP financiero; los **datos de clientes** están dispersos – parte en la base de datos e-commerce (cuentas de usuario con email), parte en un sistema de email marketing externo (listas de correos), y parte en planillas que usa el equipo de ventas para seguimiento. Esta dispersión indica un problema: no hay una vista única del cliente. Además, los datos de navegación (qué productos ve cada cliente) no se almacenan en ninguna parte actualmente. Juan y el arquitecto de datos catalogan estas fuentes. Verifican calidad: encuentran, por ejemplo, que hay clientes duplicados (mismo email registrado dos veces con variaciones en mayúsculas, etc.), y que hay productos en la base cuyo stock no coincide con la realidad (indicando integraciones débiles con almacén).
- **Modelo de datos futuro**: Para la nueva EcoShop, diseñan un **modelo unificado de información del cliente**: cada cliente tendrá un solo registro maestro en un nuevo **CRM** o base de clientes, que incluirá sus datos personales, preferencias y un ID único. Toda actividad (compras, visitas, interacciones) se relacionará a ese ID. Se introducen entidades nuevas como "Sesión de Navegación", "Recomendación" (para almacenar qué recomiendan al cliente y su resultado), "Devolución" (para registrar devoluciones formalmente, antes no había entidad para esto). Plasman en un diagrama entidades y relaciones: un Cliente realiza muchos Pedidos; cada Pedido tiene Detalles (productos comprados); Cliente tiene muchas VisitasWeb; cada Visita puede generar varias Recomendaciones mostradas; un Pedido puede tener una Devolución asociada, etc. Definen que la "**fuente maestra" de Productos** seguirá siendo una base única (sea parte de un PIM - Product Information Management - o una base común en un microservicio de catálogo). Los datos de inventario residirán en el sistema de almacén pero con replicación a un servicio de stock para mostrar disponibilidad en tiempo real en la web. Deciden implementar un **Data Warehouse** donde cada noche volcarán datos consolidados de pedidos, clientes y visitas para análisis de negocio, manteniendo historiales.
- **Seguridad de datos**: Como se manejarán datos personales (PII) y datos de pagos, se planifica cumplir con GDPR y PCI-DSS. Por tanto, se decide que en la base de datos de clientes nueva, datos sensibles (como números de tarjeta, aunque esas quizá ni se guardarán por delegar en pasarela, pero digamos direcciones, teléfonos) estarán cifrados. Definen niveles de acceso: solo el equipo de Atención Cliente puede ver direcciones completas, Marketing puede ver email pero no tarjetas, etc. Implementarán control de acceso a nivel de aplicaciones para esto. Además, deciden tokenizar las IDs de sesión para no exponer información de clientes en URLs, etc.
- **Aplicaciones futuras definidas**: EcoShop toma algunas decisiones clave: adoptarán un **CRM SaaS** (por ejemplo Salesforce o similar) para gestionar toda la información de clientes, contactos y marketing, eliminando la necesidad de planillas separadas. Para la plataforma principal, deciden construir un **nuevo frontend web** con tecnología moderna (por ejemplo, React) y una **nueva aplicación móvil** nativa, ambas consumirán APIs de backend. En cuanto al backend, optan por una arquitectura de **microservicios**

desplegados en la nube (por ejemplo, contenedores en Azure). Listan los microservicios que desarrollarán: Servicio de Catálogo de Productos, Servicio de Pedidos, Servicio de Carrito, Servicio de Pagos (aunque pagos quizá lo delegan a un tercero tipo Stripe, integrando via API), Servicio de Usuarios (perfil, autenticación, etc.), Servicio de Recomendaciones (posiblemente integrando una librería de machine learning). También deciden adquirir un **motor de búsqueda** (para mejorar la búsqueda de productos en la web) en lugar de construirlo, e integrarlo. El sistema de **WMS (almacén)** existente no se reemplazará, pero se desarrollará un conector para intercambiar datos de pedidos y stocks con él en tiempo real.

- **Interacciones entre aplicaciones**: En un diagrama, Juan dibuja: el **Frontend Web** y la **App Móvil** llaman al **API Gateway** que redirige a los distintos microservicios. El **Servicio de Usuarios** autenticará (posiblemente integrando con un servicio de identidad). El **Servicio de Catálogo** proveerá los datos de productos al frontend, consultando la base de datos de productos. El **Servicio de Pedidos** se encargará del checkout: cuando un cliente confirme compra, este servicio crea el pedido, lo guarda en su base (o base compartida con el ERP), y enviará un evento o notificación al WMS para preparar el envío. También notificará al **Servicio de Recomendaciones** para recalcular recomendaciones post-compra, etc. El **CRM SaaS** se integrará: tal vez cada noche se sincronizarán nuevos clientes y pedidos al CRM, o en tiempo real via API cuando un cliente se registra o hace una compra, esa info se envía al CRM. El **Servicio de Recomendaciones** necesitará datos de comportamiento: se decide implementar un componente de tracking en el frontend que envíe cada click o vista al backend (quizá a un servicio de tracking que almacene en una base de datos de eventos). Esos datos alimentarán el modelo de recomendaciones (podría ser un sistema de machine learning externo o uno propio entrenado con esos datos). Todas estas interacciones se delinean.
- **Catálogo de aplicaciones target**: Juan realiza una la lista:
    - **Nuevo EcoShop Web** (aplicación web e-commerce),
    - **EcoShop Móvil** (app),
    - **Microservicios EcoShop**: Catálogo, Pedidos, Carrito, Usuarios, Recomendaciones, etc.,
    - **CRM (SaaS)**,
    - **WMS actual** (integrado),
    - **Pasarela de Pagos** (tercero),
    - **Motor de Búsqueda** (componente añadido),
    - **Data Warehouse/Analytics** (plataforma de BI para reportes, posiblemente usando una herramienta BI comercial).

    Para cada uno indica si es nuevo, retención, externo, etc. La plataforma e-commerce antigua obviamente está marcada para retirar tras la migración.

- **Decisiones de make vs buy en EcoShop**: Se decidió adquirir un CRM en lugar de desarrollar, dado que no es el core del negocio desarrollar un CRM y hay soluciones maduras. También usar la pasarela de pagos de terceros para no gestionar datos de tarjetas directamente, y aprovechar su seguridad. En cambio, la tienda online en sí (frontend + microservicios) se hará a medida, para permitir a EcoShop personalizar la experiencia al máximo y evitar limitaciones de plataformas pre-hechas. Esto fue estratégico: evaluaron brevemente plataformas e-commerce comerciales, pero decidieron que la inversión en un desarrollo propio era justificada para diferenciadores (por ejemplo, lógica de recomendaciones propietaria).
- **Brechas y necesidades identificadas**: EcoShop anota varias: necesitarán desarrolladores capacitados en microservicios y cloud (actualmente su equipo solo conoce desarrollo monolítico on-premise, así que hay una brecha de skills). A nivel de datos, necesitarán migrar los datos de clientes y pedidos desde la base vieja y planillas hacia el nuevo CRM y bases unificadas – un esfuerzo de **migración de datos** significativo. Otra brecha: **integración en tiempo real** – hoy no existe y en futuro todo debe ser en tiempo real; implica implementar colas de mensajería o APIs muy robustas, lo que es nuevo para la empresa. Todas estas brechas serán tomadas en cuenta para planificar recursos y cronogramas en fases siguientes.
- **Documentación final**: El equipo documenta los diagramas de la nueva arquitectura de aplicaciones, el nuevo modelo de datos conceptual, y el catálogo con todos los componentes. Por ejemplo, crean un **diagrama de componentes** que muestra cada microservicio y su base de datos, las interconexiones (marcadas con tecnologías si ya decidieron, ej. REST API, Kafka events, etc.), la integración con CRM (p.ej., mediante API o ETL), y con WMS/pasarela. También un **diagrama de despliegue lógico** indicando que todos los microservicios estarán en la nube Azure, el CRM es SaaS nube, etc., para tener idea de distribución. Este conjunto de artefactos describe la solución de TI planeada. Antes de avanzar, lo revisan con equipos de TI y arquitectura para asegurar viabilidad (por ejemplo, involucraron al arquitecto de infraestructura para confirmar que Azure es aceptable y la empresa tiene cuenta, etc.). Con todos de acuerdo, la fase C concluye.

3.4.9 Resultado esperado

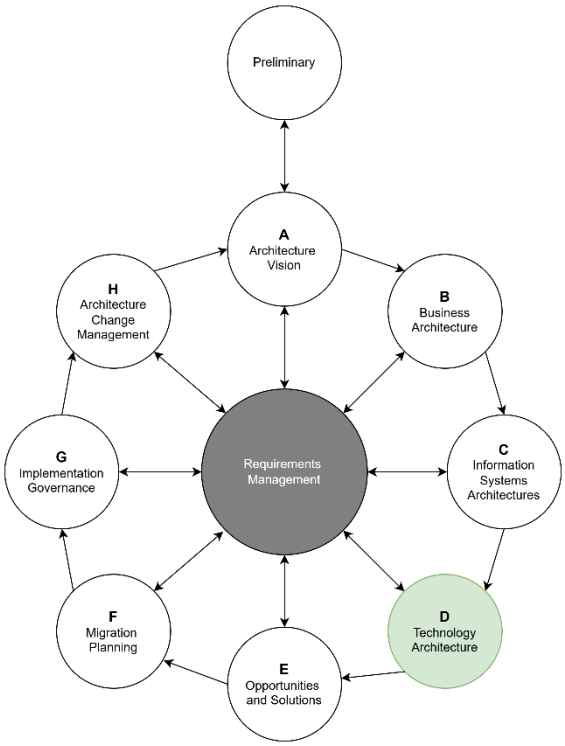
Al finalizar la Fase C, EcoShop tiene claramente definidos:

- Un **modelo de datos unificado** que soportará los procesos de negocio futuros, con entendimiento de qué datos son críticos y cómo se gestionarán.
- Un **diseño de sistema de aplicaciones completo**, identificando todos los componentes de software requeridos (nuevos o existentes), sus roles y cómo interactúan.
- Decisiones base de arquitectura de software tomadas (p. ej. microservicios vs monolito, compra vs desarrollo, cloud vs on-premise) alineadas con principios y estrategia.
- **Catálogo de aplicaciones** futuras y plan para la transición de las actuales a ese estado objetivo.

- **Requisitos refinados** en cuanto a funcionalidad de sistemas e información (muy importante: todos los requerimientos del negocio tienen una respuesta en algún componente de datos/aplicación en el diseño).
- Conocimiento de **brechas** tecnológicas y de datos a resolver, que servirá para planificar proyectos (por ejemplo, proyectos de migración de datos, entrenar personal, adquirir herramientas específicas).

En otras palabras, tras la Fase C, se dispone de la arquitectura lógica de la solución de TI necesaria: sabemos qué hay que construir o configurar en términos de sistemas y datos. Esto prepara el terreno para la Fase D, donde se detallará la infraestructura tecnológica que sostendrá estos sistemas.

3.5 FASE D – TECHNOLOGY ARCHITECTURE



Se centra en diseñar la **Arquitectura Tecnológica** necesaria para implementar las soluciones de datos y aplicaciones definidas. En esta fase se determinan los componentes de infraestructura (servidores, redes, almacenamiento, middleware, plataformas) y servicios tecnológicos que soportarán las aplicaciones, considerando además requerimientos de rendimiento, disponibilidad, seguridad y otros atributos de calidad. Básicamente, se define en qué entorno y con qué recursos tecnológicos van a funcionar las aplicaciones planificadas.

Si en la Fase C dijimos "qué sistemas de software necesitamos", en Fase D decimos "cómo los desplegaremos en términos de tecnología física/virtual y qué herramientas de base usaremos".

3.5.1 Objetivo

Desarrollar la Arquitectura Tecnológica target que provea la **plataforma de infraestructura** adecuada para las aplicaciones, datos y procesos. Esto incluye:

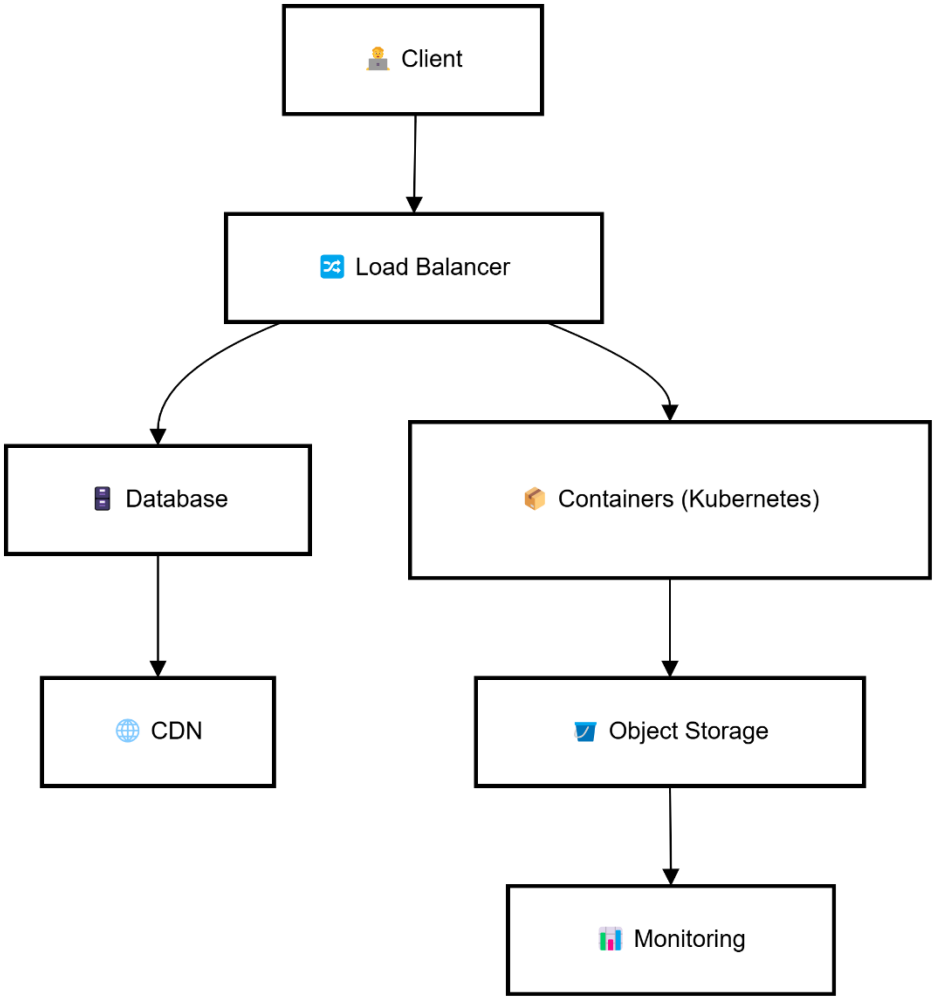
- **Describir la infraestructura actual** (baseline): qué hardware, redes y plataformas se usan actualmente y cómo están configuradas.
- **Diseñar la infraestructura futura** (target): definir si será on-premise, en la nube, híbrida; qué componentes específicos se necesitarán (ej. servidores, contenedores, base de datos, balanceadores, firewalls, etc.); cómo se estructurarán ambientes (dev/test/prod).
- **Asegurar que la tecnología soportará los requisitos**: de escalabilidad (¿cuántos usuarios concurrentes soporta?), rendimiento (¿tiempos de respuesta?), disponibilidad (¿HA, redundancia?), seguridad (¿cifrado, segmentación de red?), etc.
- **Identificar brechas tecnológicas**: por ejemplo, nuevas tecnologías a adquirir/aprender, obsolescencia de infraestructura actual a resolver, etc.
- **Planificar la migración tecnológica**: cualquier transición de infraestructura (por ejemplo, migrar de datacenter propio a nube) se considera en esta fase para luego integrarla al plan global.

3.5.2 Actividades clave

- **Revisión de la Arquitectura Tecnológica actual**: Se documenta la situación actual de la infraestructura. En EcoShop, por ejemplo, antes del cambio: hay un **datacenter propio** con 5 servidores físicos corriendo máquinas virtuales; una base de datos en un servidor físico dedicado; un firewall y balanceador de carga básico; etc. Se mapea cómo están desplegadas las aplicaciones actuales: p.ej., la plataforma e-commerce monolítica corre en 2 servidores web + 1 servidor DB; el WMS corre en un servidor en oficinas de logística; etc. Se identifican también componentes de middleware existentes (por ejemplo, si tienen un ESB - bus de servicios, o si usan servidores de aplicaciones JEE, etc.). Esta revisión sirve para ver qué se puede reutilizar y qué no.
- **Requerimientos técnicos clave**: A partir de fases previas, se consolidan los requisitos no funcionales que la tecnología debe soportar. Ejemplos: *Escalabilidad*: "la plataforma debe soportar hasta 10 veces el tráfico actual durante campañas promocionales"; *Disponibilidad*: "99.5% de uptime mensual mínimo, con recuperación ante desastres en < 1 hora"; *Seguridad*: "cumplir estándares OWASP, protección DDoS, cifrado extremo a extremo de datos sensibles"; *Desempeño*: "tiempo de respuesta < 2s en páginas principales bajo carga nominal"; *Capacidad*: "manejar base de datos de hasta X GB y crecimiento Y% anual"; etc. Estos criterios guiarán las decisiones de tecnología.
- **Diseño de la Arquitectura Tecnológica target**: Se decide la plataforma de deployment para la solución:
    - **Nube vs. On-premise**: EcoShop opta por mover a la **nube pública** (por ejemplo Azure) para aprovechar escalabilidad flexible. Se considera un modelo híbrido si

algo se queda on-premise (tal vez inicialmente el WMS sigue on-premise, pero aplicaciones web irán a nube).

- **Infraestructura como servicio vs. plataforma como servicio**: ¿Se usarán máquinas virtuales en la nube, o contenedores, o servicios serverless? Por ejemplo, deciden usar **contenedores Docker orquestados con Kubernetes** para desplegar los microservicios (eso les da portabilidad y escalabilidad). También deciden usar servicios PaaS para la base de datos (una base de datos relacional administrada en la nube) y para la cola de mensajería (usar un servicio tipo Azure Service Bus/Kafka administrado).
- **Componentes tecnológicos específicos**: Listan los componentes necesarios: servidores de aplicación (quizá no se llaman así si es microservicios, pero por ejemplo un cluster Kubernetes), un **balanceador de carga** para distribuir tráfico web, servicios de red (DNS, CDN para contenido estático de la web – deciden usar un CDN para acelerar la entrega de imágenes a nivel global), un **firewall web (WAF)** para proteger de ataques, **sistemas de monitoreo** (por ejemplo, implementar herramientas de observabilidad tipo logging centralizado, métricas, alertas). También la **arquitectura de redes**: segmentar la red en subredes seguras (front-end en DMZ, backend en subred privada), conexiones VPN a on-premise para comunicar con WMS si es local, etc.
- **Ambientes**: se define cómo se gestionarán ambientes de desarrollo, prueba y producción. En la nube, podrían replicar infraestructura en distintas cuentas o usando sistemas de orquestación.



- **Tecnologías de base para cada componente de aplicación**: Por cada elemento de la arquitectura de aplicaciones (identificado en fase C), se asigna una tecnología/plataforma:
    - Por ejemplo, **Base de Datos de productos y pedidos**: se decide usar Amazon Aurora (MySQL) o un servicio de base de datos relacional, configurado en cluster multi-AZ para alta disponibilidad.
    - **Servicio de recomendaciones (IA)**: quizá se decide usar un servicio de aprendizaje automático gestionado (como Azure Cognitive Services) – se detalla qué se necesita.
    - **Aplicaciones Web**: se decide desplegarlas en contenedores detrás de un servicio de autoscaling. O si fuera serverless, usando Azure Functions para ciertas funciones (podría ser que funciones de backend se implementen serverless, es otra decisión).

  - **Integraciones**: si se requieren conectores específicos, por ejemplo, para integrar con el WMS on-premise, se planifica una conexión segura (VPN o Direct Connect) entre la nube y la red local, y quizás un middleware (como un API Gateway que haga de puente).

- **Seguridad y conformidad tecnológica**: Se diseña cómo proteger la infraestructura: uso de **WAF** (Web Application Firewall) para la web, encriptación de datos en reposo (activar cifrado en bases de datos y storages de almacenamiento), gestión de identidades y accesos (IAM roles para servicios en nube), segmentación de red (tier web separado de tier data con listas de control). También considerar conformidad: si hay que cumplir PCI, se diseña la red aislando el módulo de pagos en un segmento altamente restringido, etc.

- **Plan de Continuidad y Recuperación**: Decisiones sobre alta disponibilidad y disaster recovery: por ejemplo, desplegar en múltiples zonas (o regiones) de la nube para tolerancia a fallos, backups automáticos, plan de recuperación ante desastres (¿tiempo de recuperación objetivo RTO y punto de recuperación RPO?). EcoShop por ejemplo decide que su e-commerce es crítico, así que configurarán redundancia multi-zonal y backups diarios de base de datos con retención de 30 días.

- **Evaluar tecnología actual vs. nueva**: Si EcoShop tenía infraestructura propia, deciden qué hacer con ella: probablemente quedará para legacy o se irá apagando. Si había contratos con proveedores (hosting, data center), se planifica terminarlos. Este análisis de impacto es necesario.

- **Analizar brechas tecnológicas**: Ejemplos: EcoShop nota que no tiene experiencia administrando Kubernetes en la nube – necesitarán capacitación o contratar un especialista (brecha de skills). También que su personal de TI necesitará formarse en la nube Azure. Otras brechas: quizá hardware actual se volverá obsoleto o repurposed, así que planifican su decomission. Listan todos los gaps en tecnología y conocimientos.

- **Documentar la Arquitectura Tecnológica**: Se crea la documentación target: diagramas de infraestructura, listado de componentes tecnológicos (muchas veces un catálogo de tecnologías o de infra con detalles de cada uno: ej. “Base de datos transaccional: MySQL en Azure, 2 nodos”; “Contenedores: AKS cluster, 5 nodos” etc., con capacidad estimada; “Red: VPC con 3 subnets, etc.”). Esta es la sección de Technology Architecture en el documento.

3.5.3 Entradas típicas

- **Arquitectura de Aplicaciones target:** Sabemos qué apps van a correr, eso nos dice qué necesitamos (ej. microservicios -> contenedores; CRM SaaS -> conexión internet robusta; base de datos unificada -> un buen servidor DB o servicio cloud).

- **Requisitos no funcionales:** Especialmente los de performance, capacidad, disponibilidad, seguridad – estos son fundamentales para dimensionar la tecnología.

- **Inventario de tecnología actual:** Para ver qué puede migrar o reutilizar. A veces componentes actuales se siguen usando en la nueva srquitectura (e.g., EcoShop podría reutilizar algún servidor existente para pruebas o su sistema de monitoreo si sirve).

- **Principios de tecnología:** Por ejemplo, "cloud-first" o "tecnología estándar de la industria", "evitar vendor lock-in" – estos afectan decisiones (podría llevar a preferir soluciones abiertas vs propietarias).
- **Tendencias tecnológicas y estándares corporativos:** Si la empresa ya estandarizó cierto vendor (por ejemplo, todo en Azure Cloud en vez de AWS), hay que seguir esa línea. O si hay preferencia por cierto lenguaje (Java vs .NET).

3.5.4 Salidas clave (Entregables)

- **Modelo de Arquitectura Tecnológica target:** Diagramas que muestren la infraestructura futura: topología de red, componentes de computo, almacenamiento y sus relaciones. Puede incluir un diagrama de *deployment* (despliegue) mostrando qué componentes de aplicación van en qué nodo o servicio.
- **Catálogo de componentes tecnológicos:** Lista detallada de todos los elementos tecnológicos necesarios (servidores, dispositivos, servicios cloud, software base) en la solución final. Incluye sus especificaciones clave (por ejemplo, cantidad de CPU, RAM, tipo de almacenamiento, etc., para dimensionar).
- **Arquitectura de entornos:** Descripción de cómo se organizan los distintos entornos (Dev, QA, Prod), y las diferencias entre ellos.
- **Documento de patrones o estándares tecnológicos:** Si se definen ciertos patrones (ej. "usar contenedores para todo desarrollo custom") o estándares (sistema operativo Linux X, base de datos Y, etc.), se listan.
- **Consideraciones de migración tecnológica:** Notas o estrategias sobre cómo pasar de la infra actual a la nueva (por ejemplo, "configurar túnel VPN antes de migrar datos", "provisionar entorno cloud paralelo para pruebas"), aunque el plan detallado vendrá en fases E/F, aquí se documenta la estrategia.
- **Actualización del catálogo de requisitos:** Incorporar requisitos adicionales que surgieron, como capacidad exacta, etc., que se van refinando.
- **Análisis de brechas:** Lista de brechas en tecnología (herramientas que la empresa no tiene y deberá obtener, skills faltantes, etc.).

3.5.5 Roles involucrados

- **Arquitecto de Tecnología/Infraestructura**: Lidera esta fase, proponiendo la arquitectura tecnológica, evaluando opciones de plataforma, dimensionando recursos.
- **Arquitecto de Soluciones/Aplicaciones**: Colabora para asegurar que las decisiones de tecnología soportan las aplicaciones (p. ej., verificar que la plataforma elegida soporta el lenguaje/framework de desarrollo del software).
- **Equipo de Infraestructura/Operaciones de TI**: Si existe un equipo de IT Ops, deben participar para aportar conocimiento de la infraestructura actual, y para validar la

viabilidad del plan tecnológico (y luego serán quienes operen la nueva plataforma, así que necesitan estar de acuerdo).

- **Equipo de Seguridad de TI**: Revisa y asesora sobre los mecanismos de seguridad en la arquitectura (firewalls, cifrado, segmentación, monitoreo de seguridad).
- **SER/DevOps/Cloud Engineers**: Si la empresa tiene expertos en cloud o DevOps, serán clave para diseñar las tuberías CI/CD, la orquestación de contenedores, automatización de infraestructura, etc., que son parte de la arquitectura tecnológica.
- **Proveedores o consultores:** Si se opta por una nube, a veces se involucra a arquitectos de la nube (ejemplo, arquitecto de soluciones de Azure) para recomendaciones, o consultores de productos específicos (CRM) para entender requisitos técnicos de integración.

3.5.6 Decisiones estratégicas

- **Elección de plataforma cloud/proveedor:** Decidir si se va con AWS, Azure, GCP u otro, o nube privada. Es una decisión estratégica de gran impacto (costos, ecosistema). En EcoShop, quizás ya tenían AWS por otro proyecto y siguen con ello.
- **Grado de tercerización vs. in-house:** Decidir qué tanto se terceriza la operación de la infraestructura. Por ejemplo, ¿contratar un servicio gestionado para operar Kubernetes, o formamos al equipo interno? ¿Usar base de datos gestionada (menos control pero menos admin) vs montar una en máquina propia (más control pero más carga operativa)? Estas decisiones definen el modelo operativo futuro.
- **Herramientas de gestión y DevOps:** Optar por ciertas herramientas/pipelines (por ejemplo, usar Jenkins/GitLab for CI/CD, usar Terraform para infraestructura como código). Son elecciones técnicas pero estratégicas para la eficiencia de entregas.
- **Costos y modelos de gasto:** Decidir la arquitectura tecnológica también lleva a proyecciones de costo (capex vs opex). En la nube, se estima cuánto costará operar la plataforma. Si es muy alto, quizás se ajustan diseños (por ejemplo, tal vez en lugar de mantener 10 servidores 24/7 se decide usar funciones serverless para escalar por demanda y reducir costo base). Estas decisiones optimizan la relación costo-beneficio.
- **Tecnologías estándar corporativas:** Confirmar alineamiento o decidir excepciones. Por ejemplo, si la empresa usaba solo bases de datos Oracle, pero en la nueva solución se quiere MySQL en cloud por costo, es una decisión de cambiar estándar. Debe tomarse con aprobación de arquitectura corporativa.
- **Roadmap de infraestructura:** Decidir si la transición tecnológica será Big Bang o gradual. Por ejemplo, ¿migramos todo a la nube de una vez, o coexistirá la infra actual con la nueva un tiempo? Esto se decide junto al plan de migración en fases E/F, pero es estratégico: EcoShop podría decidir mantener servidores actuales para entorno de testing mientras productivo va a nube inicialmente, por ejemplo.

3.5.7 Ejemplo práctico (EcoShop) – Fase D

La Fase D del ADM de TOGAF concreta con qué tecnología se materializará la arquitectura objetivo. En este caso, EcoShop decide abandonar su datacenter on-premise y migrar progresivamente a la nube pública de Microsoft Azure.

Infraestructura actual vs futura

EcoShop operaba su plataforma e-commerce desde un pequeño datacenter en sus oficinas, con servidores físicos que ya habían causado interrupciones en picos de demanda. La directiva decide apostar por la nube. Aunque algunas áreas habían experimentado servicios AWS (como S3), se opta por Microsoft Azure como proveedor cloud estratégico por sinergias con otros sistemas corporativos (Office 365, Active Directory, etc.).

Diseño en Azure

El arquitecto diseña una Red Virtual (VNet) segmentada en subredes públicas y privadas. En la subred pública se despliega un Azure Application Gateway con WAF para recibir tráfico web y enrutarlo con balanceo de carga. Tras él, se alojan los frontends y microservicios como contenedores en Azure Kubernetes Service (AKS) en la subred privada. Por diseño, estos no son accesibles desde internet directamente, sólo a través del API Management y el Application Gateway.

Se utiliza Azure API Management (APIM) como puerta de entrada segura y unificada para las APIs de los microservicios, habilitando políticas de seguridad, control de cuota y transformación.

Base de datos y caché

Para la base de datos de productos y pedidos se escoge Azure Database for MySQL Flexible Server, con replicación en zona secundaria para alta disponibilidad. Las sesiones y datos de acceso rápido se almacenan en Azure Cache for Redis.

Motor de Recomendaciones

Se opta por Azure Cognitive Services, un servicio de recomendación basado en aprendizaje por refuerzo. Los eventos de interacción del usuario se envían a este servicio, y se reciben sugerencias personalizadas en tiempo real. Esta opción evita construir un modelo ML desde cero y permite iterar rápidamente.

Integraciones seguras

El sistema de gestión de almacenes (WMS), aún on-premise, se conecta a través de una VPN Site-to-Site con la VNet de Azure. El microservicio de pedidos puede así consultar disponibilidad de inventario o actualizar datos en el sistema interno de la bodega central.

Las pasarelas de pago se integran por internet, pero el flujo PCI se delega completamente al proveedor de pagos (como Stripe o Adyen), garantizando que las tarjetas nunca tocan la infraestructura de EcoShop, minimizando riesgos.

Seguridad en Azure

Se activa el Web Application Firewall (WAF) del Application Gateway con reglas OWASP. Se configuran NSGs (Network Security Groups) para aislar microservicios y evitar accesos innecesarios. Todo almacenamiento (bases de datos, blobs, etc.) está cifrado con claves gestionadas en Azure Key Vault. Además, se implementa logging centralizado con Azure Monitor y Log Analytics, y métricas para autoescalado.

Escalabilidad

AKS se configura con autoescalado horizontal de pods, asegurando que servicios como el de catálogo o pedidos pueden escalar en función de CPU o latencia. Además, se usa Azure Container Apps para funciones más ligeras o backend auxiliares. Para eventos pico como Black Friday, se valida la capacidad de duplicar recursos automáticamente.

Ambientes y DevOps

Se definen tres entornos: Dev, UAT y Prod, separados lógicamente por resource groups e incluso suscripciones distintas. El despliegue se orquesta con pipelines CI/CD en Azure DevOps (o GitHub Actions), utilizando Bicep o Terraform para IaC y Helm para despliegue sobre AKS. Aunque no es obligatorio en TOGAF, este pipeline forma parte del diseño tecnológico.

Plan de migración

Inicialmente se mantiene el sistema antiguo en paralelo. Se levanta todo el entorno cloud, se realizan pruebas funcionales y de carga, y se programa un cutover controlado cambiando el DNS al nuevo frontend. Los servidores antiguos permanecen activos temporalmente como respaldo ante imprevistos.

Brechas identificadas

EcoShop identifica que su equipo no tiene experiencia con Kubernetes, por lo que contratan un partner especializado para el despliegue inicial y la capacitación. También reconocen la necesidad de formar al equipo de seguridad en el uso de Azure WAF y de adquirir ciertas licencias de monitoreo o CRM en modelo SaaS.

Costos estimados

Se calcula el costo mensual estimado en Azure (instancias AKS, almacenamiento, tráfico, bases de datos, etc.). Aunque el OPEX mensual puede ser ligeramente superior al coste del

datacenter, se justifica por la flexibilidad, escalabilidad automática y reducción de CAPEX inicial. Este análisis se presenta a dirección para asegurar el presupuesto de operación.

Con la Arquitectura Tecnológica definida, EcoShop tiene un blueprint claro de la infraestructura a montar. Satisface los requisitos de soportar picos de tráfico, provee alta disponibilidad (clúster multi zona), mejora la seguridad (WAF, cifrado) y es flexible para crecimiento. El equipo de arquitectura valida esta propuesta con el equipo de TI y obtiene aprobación. Ahora todas las piezas de la arquitectura (negocio, datos, aplicaciones, tecnología) están definidas. El siguiente paso será pasar de la arquitectura al **plan de implementación**, priorizando proyectos y estableciendo cómo se hará realidad gradualmente.

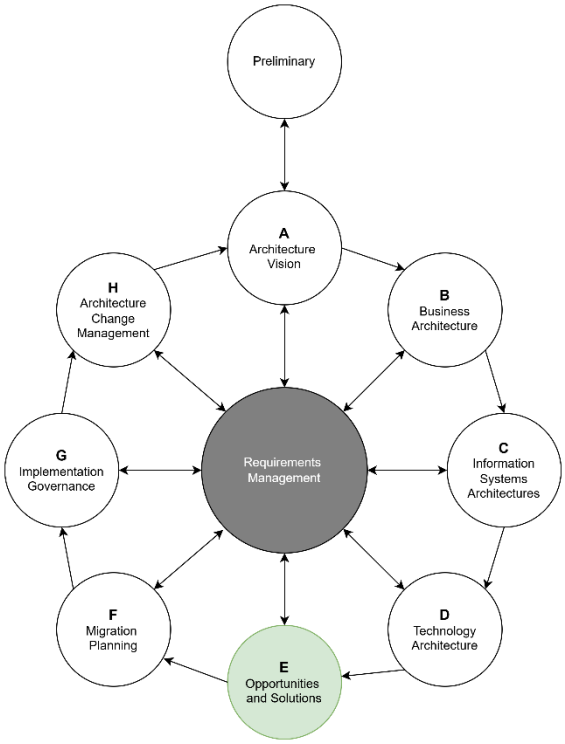
3.5.8 Resultado esperado

Tras la Fase D, EcoShop cuenta con:

- Un **diseño completo de infraestructura** para la solución: se sabe con qué tecnologías específicas se implementarán los componentes, dónde residirán (nube) y cómo se conectarán.
- **Especificaciones técnicas** para dimensionar: número de servidores/instancias, capacidades requeridas, etc., alineadas con los requisitos de rendimiento y disponibilidad.
- **Modelo de seguridad y gestión**: definido cómo se protegerá la solución a nivel infraestructura (red, acceso, datos) y cómo se operará (monitoring, devops).
- **Mapa de transición tecnológica**: claridad sobre qué elementos actuales se migrarán o apagarán y cómo se introducirá la nueva infra (aunque el plan detallado viene en fases siguientes, la estrategia ya está).
- Un entendimiento de **costos de infraestructura** y necesidades de habilidades, para asegurar que la empresa esté preparada para soportar la nueva plataforma a largo plazo.

En definitiva, la Fase D asegura que la arquitectura diseñada no es solo deseable, sino **viable técnicamente**. Con esto, la fase de diseño arquitectónico concluye; EcoShop tiene una arquitectura empresarial definida desde lo conceptual hasta lo físico. A continuación, se enfocarán en **cómo pasar de la situación actual a esa situación futura**, es decir, planificar la implementación y las migraciones necesarias.

3.6 FASE E – OPPORTUNITIES AND SOLUTIONS



Con las arquitecturas objetivo ya definidas en detalle (negocio, datos, aplicaciones y tecnología), el ADM entra en una fase de **planificación de la realización**: la **Fase E: Opportunities & Solutions**. En esta fase se identifican las **opciones de implementación** para alcanzar la arquitectura objetivo, se evalúan distintas formas de cerrar las brechas encontradas y se consolida una **estrategia de transición**. No se trata aún de calendarizar (eso es fase F), sino de determinar **qué soluciones o proyectos específicos** se necesitan para convertir la arquitectura actual en la deseada, y en qué orden o combinaciones podrían implementarse para maximizar valor y minimizar riesgos. En esencia, se transforma la arquitectura objetivo en un conjunto de **iniciativas viables**.

3.6.1 Objetivo

Desarrollar una hoja de ruta de soluciones identificando las oportunidades de implementación. Los objetivos principales son:

- **Identificar las brechas a cerrar** y agruparlas en potenciales **soluciones o proyectos**. Cada brecha (de negocio, datos, aplicaciones, tecnología) detectada en fases B, C, D debe abordarse con alguna iniciativa. Se buscan oportunidades de mejora que quizá no requieran esfuerzos gigantes o que puedan agruparse lógicamente.
- **Explorar alternativas de solución** para cada necesidad: por ejemplo, si hay que implementar cierta capacidad, ¿qué opciones de solución existen? (p.ej., desarrollar internamente vs contratar servicio externo). Evaluar las opciones en términos de costo, beneficio y viabilidad.

- **Priorizar las soluciones**: evaluar cuál debería abordarse primero en función de valor aportado, dependencias, urgencia y riesgo.
- **Definir las Transitional Architectures si necesarias**: determinar si la empresa alcanzará la arquitectura objetivo en un paso o requerirá **arquitecturas intermedias** (estados intermedios alcanzables en el camino). Muchas veces pasar del estado actual al objetivo es demasiado complejo para un solo salto, por lo que se planifican uno o más estados de transición.
- **Consolidar la hoja de ruta (Roadmap) de alto nivel**: esbozar una secuencia lógica de implementación de proyectos con sus objetivos. Esta no es la plan detallado (fase F lo hará), pero sí un roadmap ilustrativo de cómo la organización avanzará hacia la meta.
- **Evaluar el impacto y garantizar que las soluciones propuestas cumplen los objetivos**: básicamente validar que las iniciativas planeadas efectivamente realizarán la visión de arquitectura y generan los beneficios esperados.

3.6.2 Actividades clave

- **Revisar y consolidar las brechas identificadas:** Se recopilan todas las brechas de fases B, C, D. Cada brecha representa algo que falta o debe cambiar. En EcoShop, por ejemplo: "capacidad de recomendaciones inexistente", "plataforma actual no escalable", "datos de clientes dispersos", "falta equipo de analítica", etc. Esta lista es la base para crear iniciativas.
- **Agrupar brechas en soluciones lógicas:** Muchas brechas pueden resolverse con una sola **solución/proyecto**. Por ejemplo, las brechas "no hay CRM" y "datos de clientes dispersos" y "equipo de analítica inexistente" podrían agruparse en una iniciativa de "Implementar CRM y capacidades de customer analytics". Otras brechas como "plataforma no escalable" y "no se soportan picos" se agrupan en "Desarrollar nueva plataforma cloud escalable". Se forma así un listado preliminar de **Work Packages** (paquetes de trabajo o proyectos). Cada paquete representará un proyecto realizable.
- **Identificar oportunidades de reutilización o quick-wins:** En esta fase, se busca si alguna brecha ya puede cerrarse con algo existente o menor esfuerzo (**oportunidad**). Por ejemplo, puede descubrirse que la empresa ya tiene licencias de una herramienta que puede usarse. O que se puede obtener valor temprano implementando una parte de la solución rápidamente (quick win). Por ejemplo, antes de la gran reconstrucción, EcoShop podría, como quick win, habilitar su sitio actual con un CDN y mejorar tiempos de carga en semanas mientras la nueva plataforma viene (pequeña oportunidad paralela). Estas oportunidades se anotan porque aportan valor inmediato.
- **Desarrollar *Soluciones candidatas*:** Para cada bloque de solución identificado, se detalla qué implicaría. Por ejemplo, para "Implementar CRM", se describe: evaluar proveedores CRM, seleccionar uno, migrar datos de clientes, entrenar equipo, etc. Para "Nueva plataforma e-commerce", obviamente es la mayor, con desarrollo de

microservicios, migración a nube, etc. Cada solución candidata se documenta a alto nivel (objetivo, componentes incluidos, rough cost, duración estimada).

- **Evaluación de alternativas:** Donde haya opciones, se evalúan. Por ejemplo, para dotar de capacidad de recomendaciones, las alternativas podrían haber sido: construir un motor propio vs usar Amazon Personalize vs comprar un software especializado. En Fase E se podría hacer una **matriz de evaluación** de estas alternativas, considerando criterios (costo, tiempo, alineamiento con estrategia, riesgos).

Criterio	Opción A: Dynamics 365	Opción B: Salesforce	Opción C: CRM Open Source
Costo	3	2	5
Funcionalidad	4	5	3
Riesgo	2	3	4
Escalabilidad	5	5	3
Facilidad de Integración	4	5	2
Soporte / Comunidad	4	5	3

Esta evaluación lleva a seleccionar la opción preferida para llevar adelante.

- **Definir Arquitecturas de Transición (si aplica):** Si la brecha entre baseline y target es muy grande, se establecen escalones. Por ejemplo, EcoShop podría planear primero una Arquitectura de Transición 1: "Nueva plataforma core implementada pero sin módulo de recomendaciones aún, e integrando con sistema viejo parcialmente"; luego Transición 2: "añadir motor de recomendaciones y retirar totalmente el sistema viejo". Cada transición es un estado intermedio que debe ser viable y agregar valor incremental. Se delinean las características de cada estado intermedio y qué soluciones entran en cada etapa.

- **Construir el Roadmap de alto nivel:** Ahora, se ordenan las iniciativas identificadas en una secuencia temporal lógica. Se considera: dependencias (no puedo hacer X sin antes tener Y), valor (hacer primero las de alto valor o quick wins para obtener beneficios tempranos), cargas de trabajo (no saturar al mismo equipo con dos proyectos grandes a la vez), ventanas (por ejemplo, evitar implementar cambios mayores en plena temporada alta de ventas). Se puede realizar una **matriz de priorización** valor vs. esfuerzo para ordenar proyectos. Finalmente se elabora un **Roadmap** (puede ser un diagrama tipo timeline o simplemente una tabla por fases). Por ejemplo, EcoShop puede definir: *Fase 1 (próximos 6 meses): Implementar CRM y unificar datos clientes; Fase 2 (6-12 meses): Desarrollar nueva plataforma core y lanzarla en paralelo a la vieja; Fase 3 (12-18 meses): Migrar todos los usuarios a nueva plataforma, apagar plataforma antigua, implementar motor de recomendaciones avanzado.* Este Roadmap es una propuesta que luego en fase F se convertirá en plan detallado.

- **Refinar costos y beneficios por solución:** Para presentar un caso convincente, se asocian estimaciones de costos y beneficios a cada iniciativa. Por ejemplo, "Implementar CRM costará X y dará beneficios Y (mejora marketing, eficiencia atn cliente)", "Nueva

plataforma costará Z y dará incremento de ventas W". Esto es importante para priorización y para obtener aprobación presupuestaria.

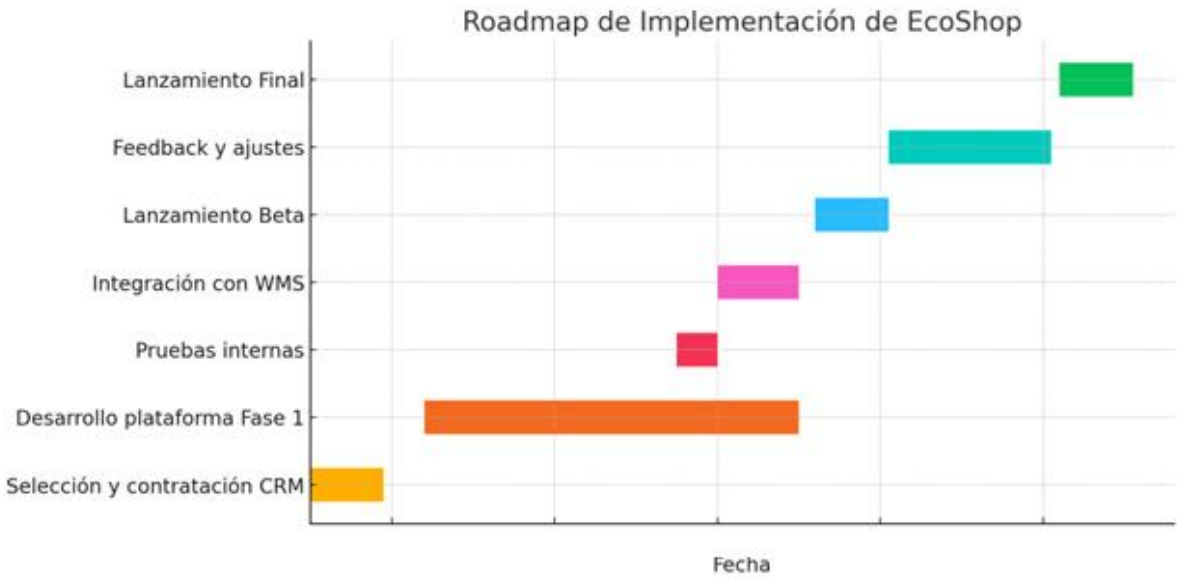
- **Validar con stakeholders:** Se revisa este conjunto de soluciones y roadmap con la alta dirección para asegurarse de que tiene sentido desde negocio y es aprobable. Puede haber ajustes (ej., algún sponsor puede decir "prefiero que primero salga lo que da cara al cliente - front-end - antes que meternos con CRM", etc.). Se busca consenso y apoyo para las iniciativas propuestas.

3.6.3 Entradas típicas

- **Deliverables de fases B, C, D:** Especialmente las **brechas identificadas** y los **requisitos**. Sin estos, no sabríamos qué falta por hacer.
- **Caso de negocio y drivers estratégicos:** Aún en esta fase, se refieren a la visión de valor: priorizar lo que más contribuya a los objetivos estratégicos definidos.
- **Capacidad y restricciones de la organización:** Por ejemplo, presupuesto disponible, capacidad de equipos internos, cronogramas externos (p.ej., "debemos tener algo listo antes de Black Friday de tal año"). Estas realidades influyen en cómo se planifican las soluciones.
- **Mercado de soluciones tecnológicas:** Información sobre opciones en el mercado (si consideramos comprar herramientas, etc.). Pueden tener RFI/RFP (ver Anexo) preliminares o estudios para decidir.

3.6.4 Salidas clave (Entregables)

- **Portfolio de Soluciones (lista de work packages):** Documento que enumera cada iniciativa/proyecto identificado, con su descripción, alcance en términos de brechas que cubre, y relación con la arquitectura objetivo (qué parte de la arquitectura implementa).
- **Matriz de mapeo brechas-soluciones:** Frecuentemente, un artefacto útil es una matriz que muestra cada brecha asociada a qué solución la cubre, para asegurar que todas las brechas están cubiertas y que no hay soluciones sin propósito.
- **Evaluación de alternativas (opcional):** Si se realizó análisis de opciones, anexar las matrices o criterios utilizados y la justificación de la elección de las soluciones propuestas.
- **Definición de Arquitecturas de Transición:** Descripciones de los estados intermedios planificados, si los hay, incluyendo qué funcionalidades existen en cada estado y cuáles no hasta llegar al final.
- **Roadmap de Arquitectura (alto nivel):** Una representación (gráfico de Gantt simplificado o timeline) mostrando las distintas iniciativas, sus secuencias y dependencias temporales, hitos principales. Podría resaltar por etapas qué capacidades se van logrando.



- **Documento de Estrategia de Implementación:** A veces se consolida en un texto que explica cómo se pasará del estado actual al futuro, resumiendo todo lo anterior, para comunicar a la organización la aproximación.
- **Actualización del caso de negocio:** Un refinamiento del análisis de valor, ahora que se tienen proyectos concretos. Podría incluir un análisis de retorno por fases, para mostrar beneficios que se van obteniendo a medida que cada iniciativa completa.

3.6.5 Roles involucrados

- **Arquitecto Enterprise/Planner:** Toma liderazgo en consolidar la hoja de ruta, viendo la foto completa.
- **Arquitectos de dominio (Negocio, Datos, Apps, Tech):** Aportan detalle en cómo sus recomendaciones se agrupan en soluciones, y evalúan alternativas específicas en su área (ej. arquitecto de datos sugiere proyecto de gobernanza de datos si vio brechas allí).
- **Gerentes de proyecto/PMO:** Pueden ser involucrados para dar input sobre esfuerzo y duración de iniciativas, ya que luego ellos gestionarán la implementación.
- **Sponsors y Stakeholders de negocio:** Para priorizar según valor y verificar la secuencia. Sus preferencias cuentan: pueden insistir en cierto quick win primero por razones políticas o de mercado.
- **Equipo financiero:** Puede ayudar a refinar costos y asegurarse de que las propuestas encajan en presupuestos.
- **Proveedores potenciales:** En algunos casos, se consulta a proveedores durante esta fase para entender tiempos/costos de ciertas soluciones (por ej., pedir cotización de un CRM, o tiempo estimado de un integrador para un módulo), ayudando a formar el plan.

3.6.6 Decisiones estratégicas

- **Secuencia de transformación:** Decidir si la transformación será *incremental* o *big bang*. Muy a menudo, incremental es preferida para reducir riesgo. En EcoShop, deciden incremental: primero implementar nuevas partes mientras la vieja convive, etc. Esta decisión influye en la definición de arquitecturas de transición.

- **In-house vs external implementation:** Decidir qué iniciativas se harán con equipo interno y cuáles con apoyo externo (proveedores, consultoras). Por ejemplo, EcoShop puede decidir contratar una empresa externa para acelerar el desarrollo del frontend, mientras el backend se hace interno. Son decisiones de ejecución estratégica (relacionado con el Anexo de la RFP).

- **Tiempo vs. alcance:** Podría haber decisiones como "¿apuntamos a tener todo en 18 meses con más recursos o hacerlo en 24 meses con recursos actuales?". Son consideraciones de estrategia de entrega, pesando rapidez vs costo vs riesgo.

- **Gestionar riesgos en la hoja de ruta:** Decisiones de mitigar riesgos vía la planificación: por ejemplo, abordar primero un módulo menos crítico como piloto para probar la tecnología, es una estrategia decidida aquí para reducir riesgo antes de acometer piezas críticas.

- **Comunicación y cambio organizacional:** A nivel estratégico, también se decide cómo se introducirá el cambio. Por ejemplo, plan de comunicación a usuarios internos, capacitación, etc., ligado a las iniciativas (no se detallará totalmente en ADM pero se considera en feasibilidad de las soluciones).

3.6.7 Ejemplo práctico (EcoShop) – Fase E

Ahora, concretemos cómo EcoShop define sus oportunidades y soluciones:

- **Listar brechas principales:** Juan reúne todo lo que falta hacer para llegar a la visión:

    1. Plataforma e-commerce actual no soporta escalabilidad ni nuevas funciones → (brecha de aplicaciones/tecnología).
    2. Ausencia de recomendación/personalización → (brecha de capacidad negocio y de sistema).
    3. Datos de cliente dispersos, sin CRM → (brecha de datos/app).
    4. Proceso de devoluciones manual → (brecha de negocio/proceso, implicando necesidad de sistema soporte).
    5. Equipo no tiene ciertas habilidades (cloud, microservicios) → (brecha organizativa/recursos).
    6. etc. (podemos enumerar más, pero digamos esas).

- **Agrupar en iniciativas:**

- **Proyecto 1: Implementación de CRM y Plataforma de Datos de Cliente.** Este abarca cerrar la brecha 3 (unificar datos cliente) y apoya la personalización. Incluye seleccionar e implementar un CRM SaaS, migrar todos los datos de clientes a él, integrar con la tienda, y entrenar a los equipos de marketing/ventas en su uso. También cubre establecer políticas de calidad de datos cliente. Quick win: unificar base de clientes mejorará campañas de marketing casi de inmediato.
- **Proyecto 2: Desarrollo de Nueva Plataforma E-commerce (Fase 1).** Este es el gran proyecto: Construir la nueva tienda online (frontend web, app móvil, microservicios core) e infraestructura cloud. Se planea hacerlo en fases: en Fase 1, implementar funcionalidades esenciales (navegación de productos, carrito, checkout básico, integración con pagos y WMS) para lanzar una versión Beta pública de la nueva plataforma. Esta iniciativa cubre brecha 1 y 4 en parte (la devoluciones se podrían incluir ya o quizás en fase 2). También aborda formar al equipo en microservicios mediante aprender-haciendo (apoyados por consultores).
- **Proyecto 3: Capacitación y Adopción Cloud/DevOps.** Si bien esto podría verse transversal, lo tratan como iniciativa: contratar consultoría para capacitar al equipo en Azure, contenedores, y establecer pipeline DevOps. Cubre brecha 5 (habilidades). Podría iniciarse inmediatamente para que el equipo esté listo mientras se arranca el desarrollo.
- **Proyecto 4: Personalización y Recomendaciones (Fase 2 de la Plataforma).** Aquí se implementa el motor de recomendaciones y las funcionalidades avanzadas de personalización. Idealmente comienza paralela a final de Fase 1 del proyecto plataforma, pero se activa plenamente una vez que el CRM y la plataforma base están integrados. Cubre brecha 2 y finaliza la parte de devoluciones automatizadas también si no entró antes.
- **Proyecto 5: Migración final y retirar de plataforma antigua.** Este proyecto es básicamente poner todo en producción final y retirar lo viejo. Incluye un plan para migrar clientes activos a la nueva plataforma (ej. invitar a todos a resetear contraseña en el nuevo sistema, etc.), cambiar DNS, hacer monitoreo intensivo tras el cambio, y apagar servidores antiguos una vez estable.

- **Alternativas consideradas:** Para Proyecto 1 (CRM), evaluaron 2 proveedores principales de CRM vs extender la plataforma actual: seleccionan la mejor por costo-funcionalidad (y de hecho, extienden la existente no era viable). Para motor de Recomendaciones, evaluaron construir su propio modelo IA vs usar Amazon Personalize vs un tercero; deciden Amazon Personalize (menor tiempo y suficiente potencia). Documentan brevemente estas decisiones. También para hosting, consideraron Azure vs AWS ya en fase D – Azure se quedó por expertise previa.

- **Arquitecturas de transición:** Definen al menos dos hitos: **Transición 1** – Nueva plataforma operando en Beta con funcionalidad básica, corriendo en paralelo a la plataforma vieja (que sigue siendo principal). **Transición 2** – Nueva plataforma con todas funciones (incluyendo recomendaciones, devoluciones auto) reemplaza totalmente a la antigua (estado final). Entre ellas está un periodo en que ambas están en uso (Beta program, quizá redirigiendo un % de usuarios a la nueva para pruebas A/B).
- **Roadmap de alto nivel:** Juan dibuja un timeline de 18 meses:
    - Q1–Q2: Implementación CRM (Proyecto 1) y Capacitación Cloud (Proyecto 3) suceden en paralelo. CRM puede estar listo en 4 meses.
    - Q3–Q4: Desarrollo Nueva Plataforma Fase 1 (Proyecto 2) – tras capacitar, el equipo construye core features; a fin de Q4 se lanza Beta (Transición 1).
    - Q5 (año 2 Q1): Feedback de Beta, mejoras, añadir módulo devoluciones si no estuvo; Integración con CRM finalizada (datos fluyendo bien).
    - Q6–Q7: Personalización (Proyecto 4) – implementar motor recomendaciones, afinar modelo con datos del CRM, lanzar funcionalidad de recomendaciones en web.
    - Q8: Migración final (Proyecto 5) – mover tráfico total a nueva plataforma, retirar antiguo sistema, Transición 2 completa.
      Este roadmap también muestra qué beneficios se logran en cada etapa: después de CRM, mejores campañas (esperan +5% ventas); después de Plataforma Fase1, mejor performance y experiencia (medirán bounce rate, etc.); después de recomendaciones, incremento venta cruzada (+10% tal vez).
- **Validación y ajustes:** Presentan el plan al CIO y comité. El CIO sugiere quizás acelerar la salida de la nueva plataforma antes de la temporada alta navideña del segundo año, por lo que preguntan si se puede recortar el timeline. Analizan que quizás se puede traslapar más el desarrollo del motor de recomendaciones con la construcción base (Proyecto 4 solapado con fin de Proyecto 2) para terminar un poco antes. Sin embargo, advierten de riesgos de paralelizar demasiado. Al final, acuerdan reforzar equipo con 2 desarrolladores externos para acortar unos meses. Con ese cambio, el roadmap queda aprobado.
- **Cálculo de valor agregado:** EcoShop revisa los números: Proyecto 1 (CRM) costará X dinero pero se espera mejore retención de clientes en un 5%, generando Y ganancias anuales; Proyecto 2 (plataforma) es el más costoso, pero permitirá soportar crecimiento del 50% en ventas en 2 años; Proyecto 4 (personalización) debería aumentar ticket promedio un 10%. Sumando todo, el caso de negocio sigue siendo positivo. Estas cifras ayudan a justificar el orden priorizado (por ejemplo, CRM primero porque mejora retención a corto plazo, mientras la plataforma nueva se construye).

Con Fase E concluida, EcoShop tiene un **portafolio de proyectos claramente definido** y una ruta orientativa para ejecutarlos. Ya no es una visión difusa, sino un conjunto concreto de iniciativas que juntos lograrán la transformación. Esto permite pasar a la Fase F, donde ese roadmap se convertirá en un **plan de migración detallado con cronogramas y recursos**.

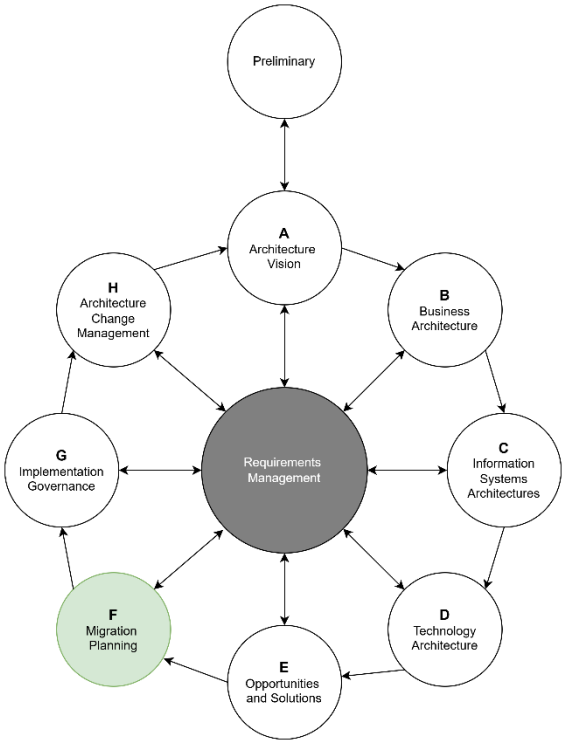
3.6.8 Resultado esperado

Después de la Fase E, la organización cuenta con:

- **Un listado de iniciativas/proyectos aprobados** necesarios para implementar la arquitectura objetivo.
- **Entendimiento de cómo cada proyecto contribuye** a cerrar brechas y lograr capacidades (trazabilidad brecha→solución→resultado).
- **Secuencia sugerida de ejecución** (roadmap) que equilibra valor, riesgo y factibilidad.
- **Acuerdo organizativo** sobre estas prioridades y sobre el enfoque de transformación (los stakeholders clave saben qué esperar primero, qué después).
- **Claridad en alternativas elegidas**: se han tomado decisiones sobre qué tecnologías/soluciones emplear para cada necesidad (ej. qué CRM, qué enfoque de recomendaciones, etc.), al menos a nivel de preferencia para proceder.
- **Fundamentación financiera y estratégica**: las iniciativas propuestas están respaldadas por un racional de valor vs costo, facilitando la obtención de presupuestos y apoyo para la siguiente fase.

En resumen, la Fase E traza el **puente entre la arquitectura en papel y la realidad**: define los proyectos con los que se construirá ese puente. Ahora, es momento de planificar ese cruce en detalle en la Fase F.

3.7 FASE F – MIGRATION PLANNING



El enfoque está en desarrollar un **plan detallado de migración e implementación** de las soluciones definidas en la fase E. Mientras que la fase E produjo un roadmap conceptual de qué iniciativas abordar y su secuencia general, la fase F aterriza eso en un plan de acción concreto, asignando tiempos, recursos y consideraciones prácticas para llevar a cabo los proyectos. Básicamente, la fase F responde: *"¿Cómo llevaremos a cabo la transformación, quién lo hará, cuándo y con qué recursos, asegurando mínima disrupción al negocio?"*.

3.7.1 Objetivo

Elaborar un **Plan de Migración** completo que guíe la ejecución. Esto incluye:

- **Priorizar formalmente los proyectos** definidos, cuantificando esfuerzo, duración y dependencias de cada uno.
- **Armar cronogramas** con hitos para cada iniciativa, integrándolos en un calendario general que refleje la secuencia óptima.
- **Asignar recursos y responsabilidades**: definir qué equipos o terceros ejecutarán cada proyecto, estimar personal necesario, presupuestos detallados por proyecto.
- **Gestionar riesgos de la implementación**: identificar riesgos operativos durante la migración (ej. posibles interrupciones de servicio, retrasos) y planificar mitigaciones.

- **Sincronizar con operaciones del negocio**: asegurar que el plan de migración se alinee con las ventanas del negocio (evitar tiempos críticos, planificar pilotos, periodos de prueba, capacitación de usuarios).
- **Obtener compromisos finales**: presentar el plan a dirección para aprobación de inversión, calendarización oficial y lanzamiento de los proyectos.

3.7.2 Actividades clave

- **Detallar cada iniciativa/proyecto:** Para cada proyecto identificado en fase E, se detalla su **alcance, tareas principales, estimación de esfuerzo** (horas/hombre o puntos de historia si usan ágil), duración estimada (en semanas/meses), requerimientos de personal (roles necesarios: desarrolladores, analistas, etc.), y dependencias. Puede elaborarse un **Project Charter** para cada iniciativa. Por ejemplo, Proyecto "Implementar CRM" contendrá tareas: seleccionar vendor (4 sem), configurar, migrar datos (8 sem), entrenamiento (2 sem), etc., con responsables asignados.
- **Refinar prioridades:** Aunque ya hay un orden propuesto, aquí se termina de priorizar con más exactitud. Se aplican técnicas de gestión de portafolio: por ejemplo, si dos proyectos podían solaparse, se decide si hacerlo en paralelo o secuencial según recursos. Si hay limitaciones de personal, quizás se re-prioriza. Este paso final produce la secuencia **confirmada** de proyectos.
- **Secuenciación y cronograma global:** Con duraciones y dependencias, se construye un **cronograma master**. Posiblemente usarán herramientas tipo Gantt o un plan a nivel portafolio. Se identifican hitos clave (por ejemplo, "CRM live", "Beta e-commerce live", "Old system decommission"). Se marcan también **puntos de integración** entre proyectos: por ejemplo, Proyecto CRM debe terminar antes de que en el de Plataforma integren esa parte. Se pueden solapar proyectos hasta cierto punto, pero con cuidado de que no choquen recursos críticos. EcoShop quizá hace CRM (6 meses) y arranca desarrollo plataforma en paralelo después de 2 meses de CRM (solapando) para ganar tiempo, pero asegurando que los datos de CRM estarán listos para integrarse cuando toque. Estas decisiones se plasman en el plan.
- **Plan de migración técnica por fases de transición:** Dado que definieron arquitecturas de transición, planean *cómo* pasar por ellas. Por ejemplo, planifican un periodo de beta donde correrán dos sistemas a la vez, cómo se hará eso: (tal vez dirigir cierto porcentaje de tráfico a la nueva plataforma con un flag, etc.). También planifican la migración final: cuándo migrar usuarios, cómo se migrarán datos en vivo (posiblemente una migración incremental o una ventana de corte). Cada transición requiere un **mini-plan** detallado: por ejemplo, para cambiar al nuevo sistema, plan de retroceso (rollback) si sale mal, equipos de soporte listos, comunicación a clientes ("hemos actualizado nuestro sitio, disculpe inconvenientes iniciales"…). Todo esto se diseña para minimizar disrupciones.

- **Asignación de recursos y presupuestos:** Se hace la **planificación de capacidad**: cuántos desarrolladores, arquitectos, testers se necesitan en cada proyecto y están disponibles. Si hay escasez en algún momento, se ajusta secuencia o se considera contratar externos. Presupuesto: se afina cuánto costará cada proyecto (licencias, consultores, horas extra, etc.) y se elabora un **presupuesto total** con cashflow por trimestre quizás. Esto se preparará para aprobación.
- **Plan de gestión del cambio organizacional:** Aunque el ADM se enfoca en arquitectura, un buen plan de migración también incluye cómo se preparará al *negocio* para los cambios. Ejemplo: plan de capacitación de usuarios finales (atención al cliente deberá aprender el nuevo CRM, etc.), comunicación interna y externa (anuncios de nuevas funcionalidades a clientes), políticas de soporte durante la transición (reforzar mesa de ayuda cuando se lance la nueva plataforma, esperando consultas). Estas actividades se calendarizan junto con los proyectos técnicos.
- **Actualización del Risk Register:** Se revisan los **riesgos** de ejecución: Ej. "posible retraso en desarrollo microservicios si la curva de aprendizaje es alta", "riesgo de baja adopción del CRM por usuarios si no se entrena bien", "riesgo de solapamiento con peak season de ventas". A cada riesgo se le asigna un plan: mitigación o contingencia. Tal vez deciden: tener un buffer de 1 mes en cronograma antes de la peak season para acomodar retrasos, o planear un lanzamiento parcial. Estos planes se integran en el cronograma (por ejemplo, buffer time).
- **Formalizar el Plan de Migración:** Se consolida todo en un documento o presentación de **Plan Maestro de Implementación**, que incluye los cronogramas, asignaciones, presupuestos, riesgos y mitigaciones, gobernanza de la implementación (quién supervisará la ejecución, probablemente un steering committee).
- **Obtener aprobación final:** Se presenta este plan a la dirección y stakeholders para asegurar compromiso de los recursos y presupuesto. Con la aprobación, se da luz verde para ejecutar los proyectos.

3.7.3 Entradas típicas

- **Roadmap y soluciones de Fase E:** base fundamental, aquí se los detalla.
- **Capacidad/Disponibilidad de recursos:** Información de RRHH sobre cuántos equipos de desarrollo se tienen, etc., y de Finanzas sobre presupuesto anual disponible, para ajustar plan realista.
- **Calendario de negocio:** Fechas importantes (picos de ventas, cierres de año fiscal, etc.) para evitar implementaciones muy disruptivas en esos periodos.
- **Políticas de la organización para proyectos:** Si hay un PMO, probablemente exija cierto formato de plan, criterios de prioridad, etc., integrarlo con portafolio global de la empresa.

- **Contratos con proveedores actuales y futuros:** Por ejemplo, saber que contrato del datacenter actual vence tal fecha – quizás alinear decommission con eso; o que oferta de precio del CRM hay que aprovechar antes de X.

3.7.4 Salidas clave (Entregables)

- **Plan de Migración (Implementación) detallado:** documento principal, puede incluir:
    - **Cronograma consolidado** (posiblemente un Gantt) de todas las iniciativas con hitos.
    - **Descripción de proyectos** (objetivos, alcance, tiempos, responsables).
    - **Recursos asignados** (equipos, roles, quizá un RACI matrix indicando responsabilidades).
    - **Presupuesto** y plan financiero (inversión por fase, ROI esperado por fase si aplicable).
    - **Riesgos y mitigaciones** específicos de la implementación.
    - **Planes de comunicación y capacitación** vinculados a hitos.
    - **Mecanismos de governance** durante la implementación (ej. reuniones de seguimiento, métricas de progreso).
- **Matrices de asignación:** a veces se entrega una **matriz de asignación de recursos a proyectos** y **matriz de dependencias** más detallada.
- **Actualización del Repositorio de Arquitectura:** El plan de migración también se registra como parte de la arquitectura. En TOGAF, un **Implementation and Migration Plan** es un deliverable formal que se añade.
- **Requests for Architecture Change (si surgieron):** a veces al planear, surgen pequeños cambios a la arquitectura objetivo por practicidad (ej. "quizá mantengamos tal sistema un año más, así que la arquitectura target se alcanza en 2 fases"). Estas decisiones se documentan como ajustes a la arquitectura o como solicitudes de cambio si difieren de lo diseñado; son realimentadas en el repositorio para mantener consistencia entre diseño y plan.

3.7.5 Roles involucrados

- **Arquitecto líder/Enterprise Architect:** Sigue liderando, asegurando que el plan no comprometa los objetivos arquitectónicos (por ejemplo, si se decide posponer indefinidamente alguna parte, él advierte si eso afectará la visión).
- **Gerentes de proyecto/PMO:** Toman co-liderazgo, aportando técnicas de planificación, deadlines realistas, y se preparan para ejecutar los proyectos. En esta fase, a menudo el liderazgo transiciona de "arquitectura" hacia "gestión de proyectos".

- **Sponsors de cada iniciativa:** Deben estar involucrados para aprobar tiempos y recursos de su área, y para apoyar en priorización.
- **Equipos técnicos y de negocio clave:** Son consultados para estimar esfuerzos. Por ejemplo, el líder de desarrollo estima cuánto tardará construir X, el jefe de marketing estima cuánto tiempo necesitan para adoptar el CRM. Esta información bottom-up hace el plan más acurado.
- **Finanzas:** Aprueban presupuesto final y aseguran que se alinea con proyecciones de la empresa.
- **Recursos humanos (si existe cambios organizativos):** Si se requiere contratar personal o reasignar, RH participa para programar reclutamientos o reubicaciones en el tiempo.

3.7.6 Decisiones estratégicas

- **Trade-offs tiempo/costo/scope:** Puede que al hacer el plan haya que ajustar: Si todos los proyectos suman más recursos o tiempo que aceptable, se decide recortar scope o invertir más. Por ejemplo, tal vez incluir todas las funcionalidades en fase 1 es inviable, se decide estratégicamente cuáles pueden esperar a fase 2.
- **Uso de metodologías de implementación:** Decidir si los proyectos se ejecutarán con metodologías ágiles, cascada o híbridas, y reflejarlo en plan. EcoShop podría optar por ejecutar el desarrollo de la plataforma con metodología ágil (sprints), CRM quizá con enfoque waterfall (configuración), etc. Esto afecta cómo se planifica (ágil a veces planifica releases en vez de todas las tareas upfront).
- **Gobernanza de la transformación:** Decidir cómo se hará la supervisión estratégica: se crea un **Steering Committee** con el CIO y líderes de negocio que se reunirá mensualmente para revisar avance, etc. Se define un **Change Control**: cómo manejar si surgen cambios de alcance en proyectos, asegurando alineación con arquitectura (ej. requerir aprobación del Architecture Board para desviaciones significativas).
- **Comunicación pública del plan:** A nivel estratégico, decidir cuándo y cómo se comunicará a la organización (y a clientes, si aplica) los hitos principales. Por ejemplo, quizás no se anuncia nada a clientes hasta el Beta, pero internamente se comunica pronto para alinear a todos.
- **Alignment con otros programas:** Ver si este plan colisiona o depende de otros proyectos de la empresa. Estratégicamente, puede requerir re-priorizar otras iniciativas corporativas para liberar recursos para esta, o coordinar con ellas (por ejemplo, si TI también está migrando infra de otras apps a la nube al mismo tiempo, coordinar esfuerzos).

3.7.7 Ejemplo práctico (EcoShop) – Fase F

- **Detallar proyectos:** EcoShop asigna gerentes de proyecto provisionales para cada iniciativa. Junto con Juan, detallan cada plan. Ejemplo: Proyecto “Nueva Plataforma Fase 1” se dividirá en subproyectos: desarrollo frontend, desarrollo backend, infraestructura. Le

asignan 10 desarrolladores, 2 QA, un arquitecto (Juan) y un Scrum Master para hacerlo ágil en sprints de 2 semanas. Estiman ~8 meses de desarrollo intensivo para tener un MVP beta. Proyecto CRM: 1 jefe de marketing liderará con 2 analistas de negocio, trabajar con un partner del CRM elegido, ~4 meses. Documentan entregables de cada.

- **Cronograma global:** Hacen un Gantt integrando:
    - Marzo – Junio: Implementación CRM (4 meses).
    - Marzo – Octubre: Desarrollo Plataforma Fase1 (8 meses). Notan solapamiento: CRM acaba en junio, plataforma sigue hasta oct. Dependen en parte, pero integrarán CRM API en plataforma alrededor de julio, así que planifican ese como punto de integración (dep: CRM done by July 1 to integrate by Sprint 10, digamos).
    - Julio: Capacitación Cloud terminó en mayo, así que el equipo está apto; continúa apoyo consultor hasta julio.
    - Septiembre: Preparar beta launch – decide lanzar beta interna a empleados primero por 2 semanas (hito en cronograma).
    - Octubre: Beta pública (Transición 1). Cronograma coloca esta antes de Noviembre (peak navidad) intencionalmente para probar en entorno real pero controlar antes de Black Friday.
    - Nov – Dic: Congelación de cambios mayores por peak season, solo mejoras menores y soporte (lo reflejan en plan: se delimita ventana de congelamiento).
    - Enero – Marzo siguiente año: Desarrollo Plataforma Fase2 (Recomendaciones, Devoluciones auto, etc.), ahora que base estable.
    - Abril: Integración motor recomendaciones live.
    - Mayo: Planificar cambio total: migrar usuarios, apagar viejo. Programan eso para Mayo, época más tranquila de ventas. Hito Transición 2: Nuevo sistema full, Legado off.
    - Junio – Julio: Proyecto de decommission de sistemas antiguos y optimizaciones finales.

Plan holguras: agregan 2-3 semanas buffer tras grandes entregas en caso de retrasos.

- **Recursos y presupuesto:** Ven que en pico desarrollo (julio-sept) necesitan 12 desarrolladores pero solo tienen 8 internos, así que contrataron 4 consultores temporales (lo planearon en presupuesto). CRM partner cost X, consultores Y, Azure infra Z (calcula coste nube durante desarrollo y tras go-live). Suman ~N millones total, escalonados en dos años. Presentan ROI: con incremento ventas proyectado, se recupera en 3 años.

- **Riesgos:** Listan: “Riesgo: retraso en desarrollo plataforma cause lanzamiento Beta más cerca de peak -> Mitigación: contratar recursos extra (hecho) y control estricto de scope (dejar funcionalidades no críticas para fase2)”; “Riesgo: usuarios no adoptan CRM -> Mitigación: hacer entrenamiento intensivo y champion users en cada equipo, fase de adaptación paralela”; “Riesgo: fallo durante cambio final -> Mitigación: plan de rollback: mantener sistema viejo standby por 2 semanas, y hacer cambio un domingo noche con equipo completo vigilando, con posibilidad de revertir DNS en 30 min si algo crítico falla”.
- **Plan de comunicación y capacitación:** Deciden: en Septiembre tras beta interna exitosa, anunciarán a toda la empresa la nueva plataforma en camino, con demos para equipos de atención cliente. A clientes externos, plan de marketing: en Beta pública, invitar a clientes leales a probar la “nueva experiencia”. Capacitación: en marzo, el equipo de atención cliente se entrenará en CRM; en Abril, equipo de almacén entrenado en nueva interfaz de pedidos; justo antes del go-live final, manuales actualizados y tutoriales en línea. Todo eso es calendarizado con responsables (RH apoya en entrenos, marketing en comunicación).
- **Aprobación:** Presentan el plan al Steering (CIO, Director Digital, CEO). Ven el costo y cronograma. CEO pregunta si se puede adelantar go-live total antes de mayo para que resultados impacten Q1 año 2. Equipo argumenta prudencia: hacerlo en marzo sería en medio de implementaciones, con más riesgo. CEO concede pero pide tratar de ganar unas semanas en beta -> Plan ajusta Beta pública a final de Sept (en vez de Oct) asumiendo consultores extra. Aprobado.
- **Luz verde:** Con esto, se formaliza lanzamiento del “Programa de Transformación eCommerce EcoShop 2025”, se asignan oficialmente líderes de proyecto, y el ADM “de diseño” da paso a la ejecución supervisada por la gobernanza de arquitectura.

3.7.8 Resultado esperado

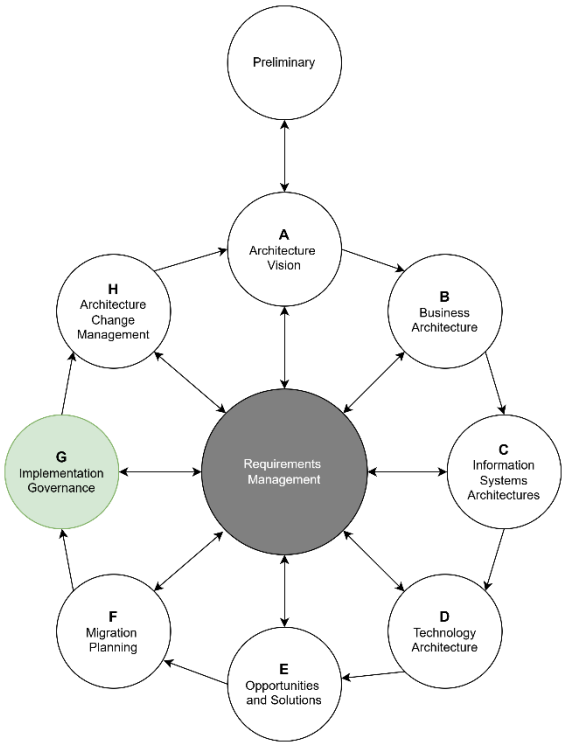
Tras la Fase F, EcoShop tiene:

- **Un plan de implementación integral**, aprobado y respaldado, que sirve de guía para todos los involucrados en los próximos meses.
- **Proyectos definidos con timelines y responsables**, listos para iniciar (o algunos ya iniciados inmediatamente).
- **Compromiso de recursos y presupuesto asegurado**, ya que dirección aprobó formalmente la inversión y los equipos fueron asignados.
- **Riesgos conocidos con planes de contingencia**, aumentando la probabilidad de una ejecución sin sorpresas graves.
- **Alineamiento total**: tanto equipo de arquitectura, de TI, de negocio y dirección están en sintonía sobre qué se hará, cuándo y cómo.

- **Continuidad del rol de arquitectura:** Se sabe cómo el Architecture Board/arquitectos seguirán participando en la fase de implementación (por ejemplo, revisiones de diseño, gobernanza de cambios), preparando terreno para la fase G.

Con ello, se sale de la fase de planificación y se entra en la **ejecución de los proyectos**. Aquí el ADM interactúa con los procesos de desarrollo y gestión de proyectos, pero asegura mantener el control arquitectónico a través de la fase G.

3.8 FASE G – IMPLEMENTATION GOVERNANCE



Corresponde a la etapa en que los proyectos definidos están en marcha (fase de implementación) y se debe asegurar que la **realización** de esas soluciones se mantiene alineada con la arquitectura planificada. En otras palabras, es la fase de **gobernanza de la implementación**: se implementan mecanismos de supervisión, control de cambios y cumplimiento para garantizar que lo construido corresponde a lo diseñado y aprobado. Dado que la ejecución puede durar meses o años con múltiples equipos trabajando, la gobernanza arquitectónica evita desviaciones que comprometan objetivos o principios.

3.8.1 Objetivo

Supervisar y dirigir la ejecución de los proyectos para garantizar la **conformidad arquitectónica** y gestionar cualquier cambio necesario. Esto incluye:

- **Asegurar cumplimiento** de la arquitectura: que las soluciones desarrolladas sigan los principios, estándares y modelos definidos (por ejemplo, que los desarrolladores

efectivamente construyan microservicios según la arquitectura, que no introduzcan atajos que violen principios como reutilización o seguridad).

- **Gestionar solicitudes de cambio** arquitectónico: durante la implementación, pueden surgir necesidades de modificar la arquitectura (p.ej., una tecnología no funciona como se esperaba y hay que cambiarla). El objetivo es tener un proceso formal para evaluar y aprobar/rechazar esos cambios, manteniendo integridad global.
- **Realizar revisiones de arquitectura** en hitos clave: por ejemplo, revisar diseños detallados de solución antes de construirlos (design review), revisar que las implementaciones cumplen requisitos (pre-GoLive review).
- **Capturar desviaciones y definir dispensas**: si por alguna razón se decide conscientemente no seguir un estándar en una instancia particular, documentarlo (dispensación) y evaluar su impacto.
- **Monitorizar la entrega de beneficios**: asegurarse de que los proyectos implementados están logrando los valores esperados (o al menos que se están configurando los KPI para medirlos).
- **Mantener involucrados a los stakeholders**: comunicación regular del avance, logros y también de ajustes de plan si los hubiera.
- **Asegurar la transición al estado operativo**: que los equipos de operaciones reciban los entregables, conocimientos, documentación y que la solución pueda ser sostenida a largo plazo.

3.8.2 Actividades clave

- **Establecer la función de Architecture Governance:** Antes o al inicio de la ejecución, se pone en marcha formalmente el modelo de gobernanza definido en Preliminary. Esto suele implicar:
    - Un **Architecture Board** activo que se reúne periódicamente (ej. mensualmente) para revisar el progreso de arquitectura.
    - Definir **puntos de control** (checkpoints) en cada proyecto donde los arquitectos revisarán entregables clave. Por ejemplo: revisión de diseño detallado, revisión de modelo de datos físico, revisión de pruebas de seguridad, etc.
    - Establecer los **criterios de conformidad**: listas de chequeo o criterios que cada proyecto debe cumplir (por ejemplo, "todas las APIs deben tener autenticación token", "debe usar cifrado AES128 para datos en reposo", etc., según los principios y estándares).
- **Revisiones de Arquitectura de soluciones:** A medida que los equipos de proyecto producen diseños detallados, los arquitectos (Juan y colegas) realizan **Architecture Compliance Reviews**. En EcoShop, por ejemplo, antes de que el equipo de plataforma

implemente, Juan revisa que la arquitectura de cada microservicio cumple con lo definido: verifica que usan la estructura correcta, que las integraciones siguen el modelo (usando el API gateway, etc.), que no están introduciendo un lenguaje o base de datos no aprobada sin análisis. Estas revisiones pueden generar hallazgos (ej. "el diseño propuesto del módulo X viola el principio Y; por favor corregirlo" o "necesita excepción aprobada"). Documentan cada revisión.

- **Gestionar cambios de arquitectura:** Supongamos durante implementación surgió un cambio: por ejemplo, el proveedor de CRM requería una integración diferente a la planificada. O el equipo descubre que cierto microservicio necesita subdividirse en dos por razones de rendimiento. Estos cambios se documentan como **Architecture Change Requests**. El Architecture Board evalúa: ¿impacta negativamente a largo plazo? ¿Sigue alineado a principios? Si es aceptable, aprueba el cambio y actualiza la documentación de arquitectura para reflejarlo (ajustando quizá el modelo de aplicaciones o datos). Si no, busca alternativas. Un registro de todas las solicitudes y su resolución se mantiene. Esto es importante para mantener la trazabilidad y aprender lecciones.
- **Monitorear cumplimiento en construcción:** Además de revisiones formales, se puede integrar la gobernanza en el flujo: por ejemplo, incluir arquitectos en las dailies o demos (en métodos ágiles) para detectar desviaciones pronto. Herramientas de CI/CD pueden tener checks automáticos (linting contra estándares, etc.). Todo para garantizar compliance continuo.
- **Gestionar riesgos y issues arquitectónicos:** Durante la ejecución, la gobernanza también se enfoca en riesgos detectados: por ejemplo, si el desempeño de un módulo no alcanza lo planificado, se aborda si requiere un rediseño. O si un proveedor se retrasa, plan de contingencia (¿buscar otro?). La gobernanza se coordina con la gestión de proyecto en la resolución de problemas que afecten la arquitectura global.
- **Comunicación de progreso a stakeholders:** Se prepara informes periódicos del avance de la implementación en términos de capacidades logradas y arquitectura implementada. Por ejemplo, tras lanzar la Beta, se reporta al Architecture Board y sponsors: "Se ha implementado transacción en nube con éxito, principios X, Y, Z se cumplieron, pero tuvimos que hacer una excepción menor en…". Mantener transparencia genera confianza.
- **Preparar la organización para el cambio:** La gobernanza también vigila que se cumplan planes de gestión del cambio (capacitaciones realizadas, manuales creados, etc.). No es solo aspecto técnico; se asegura que la empresa esté lista para adoptar la nueva arquitectura cuando entre en operación permanente.
- **Formalizar entregas finales y cierre de proyectos:** A medida que los proyectos terminan, la gobernanza verifica que todos los **artefactos finales** se han entregado: documentación actualizada de arquitectura, código fuente almacenado, contratos actualizados, etc. También se comprueba que los **beneficios** prometidos empiezan a

medirse (por ejemplo, sistema de métricas de negocio en marcha). En EcoShop, tras go-live total, medirán la mejora en tiempos de carga, conversión, etc., comparándolos con objetivos. La gobernanza se asegura de que esa medición esté planificada.

- **Actualizar el Repositorio de Arquitectura:** Muy importante: conforme se implementan cambios, se actualiza en el repositorio los modelos de arquitectura "as built" (como quedó finalmente). La arquitectura objetivo teórica puede haber cambiado un poco; ahora se documenta la arquitectura actual real resultante. Esto es la base para fase H.

3.8.3 Entradas típicas

- **Plan de Migración (fase F) y arquitectura objetivo detallada:** Es la referencia de lo que se debe lograr y cómo.
- **Políticas de gobernanza definidas en Preliminary:** Por ejemplo, procedimientos de compliance review, roles del Architecture Board, etc., que ahora se ejecutan.
- **Artefactos de arquitectura generados en fases previas:** Modelos, principios, requisitos. Todo esto sirve para los checklists de cumplimiento.
- **Requerimientos actualizados:** Si en fase E/F se ajustaron requerimientos, tenerlos claros para validar que se satisfacen.

3.8.4 Salidas clave (Entregables)

- **Registros de revisión de arquitectura:** Documentos o listas de verificación de cada revisión con resultados (cumple/no cumple, acciones requeridas, excepciones concedidas).
- **Log de Solicitudes de Cambio de Arquitectura:** Un registro de todos los Architecture Change Requests procesados, con su decisión (aprobado, denegado) y justificación.
- **Actualizaciones de la documentación de arquitectura:** Cualquier cambio aprobado se refleja en los modelos. Por ejemplo, el catálogo de aplicaciones se actualiza si se añadió/eliminó una aplicación no prevista, etc.
- **Reporte de conformidad final:** Al final de la implementación, se puede generar un informe de **Cumplimiento de Arquitectura** que resume qué tan alineado quedó el producto final con la visión inicial, enumerando dispensas si las hubo.
- **Sistemas y capacidades operativas**: La entrega real de sistemas en producción listos. (Si bien esto es más un resultado de proyectos, desde la vista del ADM, la salida es "soluciones implementadas en conformidad con la arquitectura").
- **Lecciones aprendidas en términos arquitectónicos:** Pueden documentarse hallazgos para retroalimentar la metodología o la arquitectura futura (por ej., "El principio X era muy estricto, tuvimos que flexibilizarlo; aprender para futuros ciclos").

3.8.5 Roles involucrados

- **Architecture Board/Comité de Arquitectura:** Toman decisiones sobre cambios y aseguran la conformidad general.
- **Arquitectos de dominio:** Participan en revisiones técnicas específicas de su área (por ej. arquitecto de seguridad revisa configuración de seguridad; arquitecto de datos valida modelos físicos de BD).
- **Líderes de proyecto / Scrum Masters:** Colaboran en incluir puntos de control en su plan y facilitan la interacción con los arquitectos (por ej., planificar que el sprint X tenga tiempo para la revisión de arquitectura).
- **Equipo de QA / testers:** A veces ayudan a verificar cumplimiento, especialmente en temas de rendimiento o seguridad (e.g., hacen pruebas de seguridad para ver si se cumplieron las políticas).
- **Usuarios claves / product owners:** Pueden alertar desviaciones desde la perspectiva de requisitos de negocio ("Esto que están entregando no cumple lo que pedimos" - y escalaria a un tema de arquitectura si es serio).
- **Gestores de configuración y release:** Para asegurarse que las versiones que pasan a producción han pasado por los checks de arquitectura necesarios (imponer gate de aprobación de arquitectura antes de despliegues, por ejemplo).

3.8.6 Decisiones estratégicas

- **Manejo de excepciones:** Decidir hasta qué punto se toleran desviaciones. Por ejemplo, si un proyecto argumenta que para su entrega a tiempo necesita saltarse un estándar, el Architecture Board debe decidir: ¿se concede la excepción temporal/permanente? ¿o se retrasa entrega para cumplir? Esto es estratégico porque implica balance entre ideales arquitectónicos y pragmatismo.
- **Repriorización sobre la marcha:** Si emergen circunstancias (ej: un competidor lanza una función nueva, o un riesgo materializa retrasando un proyecto), la gobernanza puede recomendar ajustar la secuencia del plan. Por ejemplo, tal vez adelantar una pequeña funcionalidad para responder al competidor, incluso si no era prioritario antes. Estas decisiones de cambio de plan, manteniendo la arquitectura integrada, pasan por la gobernanza.
- **Comunicación de cambios:** Decidir cómo y cuándo comunicar a la organización variaciones significativas. Ej: si se aprobó un cambio de tecnología por dificultades técnicas, se informa al sponsor con la justificación para mantener confianza.
- **Calidad vs. tiempo:** Muchas decisiones de governance se reducen a este trade-off. Debe decidirse estratégicamente cuándo insistir en la calidad arquitectónica (porque ceder comprometería la visión a largo plazo) y cuándo permitir alguna flexibilidad para no

frenar demasiado la entrega. Esto requiere criterio y respaldo de la alta dirección para los arquitectos.

- **Cierre formal de arquitectura vs. mejoras continuas:** Hacia el final, la gobernanza decide cuándo la arquitectura implementada es "suficientemente buena" y los proyectos pueden cerrarse, vs. qué puntos quedan para mejora en ciclo futuro. Esta transición al BAU (Business as Usual) es estratégica para no eternizar proyectos.

3.8.7 Ejemplo práctico (EcoShop) – Fase G

- **Operando la gobernanza:** EcoShop establece reuniones quincenales del Architecture Board (Juan, CIO, líderes de TI) durante la ejecución. Ahí Juan presenta estado arquitectónico. Por ejemplo, tras unos sprints, notifica: "Se revisó diseño de microservicio de pagos, cumple principios. Observación: equipo consideró usar un servicio externo para notificaciones a clientes, no estaba en arquitectura original, lo evaluamos y aprobamos ya que no afecta negativamente." Todo queda en acta.

- **Revisiones concretas:** Antes del Beta, se hace una revisión de arquitectura integrada: los arquitectos comprueban que todos los componentes desplegados (frontend, servicios, integraciones) cumplen el diseño: por ej., verifican que las APIs documentadas en la arquitectura se implementaron, que no se creó ninguna integración "oculta" punto a punto saltándose el API Gateway. Encuentran una cosa: el equipo, por rapidez, conectó la aplicación web directamente al servicio de recomendaciones de Amazon, saltándose el API Gateway. Esto viola la regla de que todo pase por el gateway (principio de control centralizado de APIs). Se discute: la razón fue simplicidad, pero arquitectónicamente preferían encapsular. Deciden registrar una **excepción temporal**: permiten que en Beta esté así, pero con la condición de corregirlo antes de producción final, integrando la llamada via el gateway. Documentan la excepción.

- **Change requests:** Durante desarrollo, surgió un problema con el proveedor de CRM: no ofrecía una funcionalidad de segmentación en tiempo real que se quería. Se plantea hacer una pequeña base de datos adicional para complementar. Juan levanta un **Architecture Change Request**: agregar un *Customer Segmentation Service* no previsto. El Board evalúa: es un componente nuevo, ¿afecta mucho? Concluyen que es alineado al objetivo (mejor personalización), aunque añade complejidad. Lo aprueban, piden actualizar el modelo de aplicaciones para incluirlo y asegurar su integración con CRM y data lake. Esta es una modificación al plan original, pero controlada y documentada.

- **Monitoreo de proyectos:** La gobernanza también vigila no solo "qué" se hace sino "cómo va todo". Por ej., nota que el Proyecto plataforma va a tardar 2 semanas más de lo previsto en Beta. Se discute en Architecture Board si eso compromete la ventana pre-Black Friday. Deciden que es aceptable retrasar Beta 2 semanas aún antes de periodo crítico, pero actualizan plan de migración en consecuencia y comunican a marketing que Beta pública será un poco más tarde. Transparencia para evitar sorpresas.

- **Preparación para go-live final:** Previo al cambio definitivo, el Board revisa un **checklist**: ¿Se hicieron pruebas de carga suficientes? (Arquitecto de tecnología: sí, alcanzamos 5k concurrentes sin problemas); ¿Backups configurados? (sí, nightly on cloud); ¿Plan de rollback? (documentado); ¿Soporte 24/7 listo? (el jefe de TI confirma personal en guardia la semana del cambio). Solo tras confirmar todo, autorizan proceder con el go-live en la fecha planificada.
- **Post-implementación:** Tras el go-live completo, la gobernanza convoca una última revisión retrospectiva. Analizan: la arquitectura implementada coincide 95% con la visionada. Quedaron algunos cambios: se introdujo el servicio de segmentación extra, y el WMS finalmente también se migrará a nube en próximo año (decisión surgida al ver beneficios en e-commerce). Esos puntos se documentan. El Board aprueba que la arquitectura final se considere "baseline actual" para adelante. También recogen lecciones: por ejemplo "Deberíamos haber involucrado antes al proveedor CRM para conocer limitaciones – mejora para próximo ciclo".
- **Beneficios medidos:** Pasados 3 meses del nuevo sistema, la gobernanza revisa indicadores: velocidad de página mejoró a 1.5s, conversión subió 5%, ventas +10% comparado con año anterior, retención +3%. No todos los objetivos al 100% todavía (personalización aún aprendiendo, se espera suba en meses). Pero claramente se logró gran parte del valor. Reportan al patrocinador: arquitectura cumplió su promesa – con evidencias.

EcoShop finaliza la implementación con éxito bajo la tutela de la gobernanza de arquitectura, que aseguró que la visión estratégica no se perdiera en la vorágine de la ejecución.

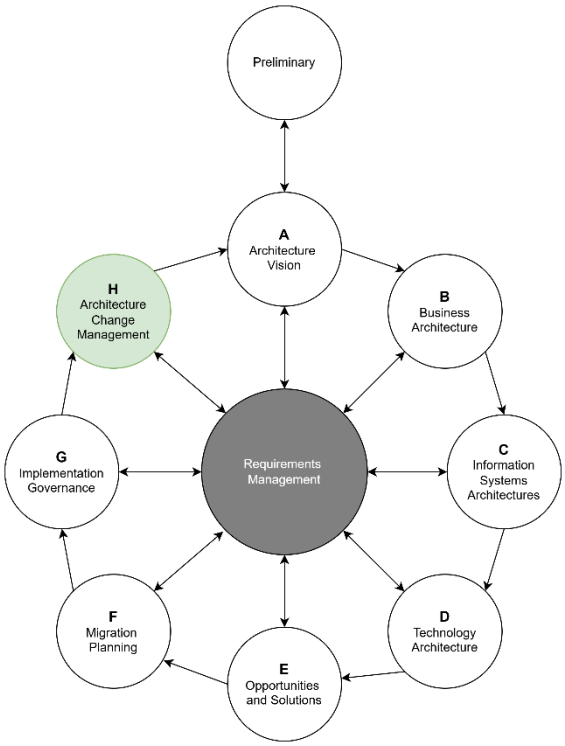
3.8.8 Resultado esperado

Al culminar la Fase G (y los proyectos asociados), se obtiene:

- **Las soluciones implementadas en producción**, que cumplen con los requisitos y principios establecidos, entregando las capacidades esperadas.
- **Registro de conformidad**: documentación de cómo se implementó, qué desviaciones hubo, con aprobaciones formales para ellas.
- **Arquitectura actualizada**: el repositorio refleja fielmente la nueva arquitectura empresarial de EcoShop (que ahora pasa a ser el "baseline" actual).
- **Mecanismos de gobierno establecidos**: la estructura de gobernanza que se usó en este programa puede quedar como permanente para gobernar futuras iniciativas de arquitectura.
- **Confianza en la arquitectura**: los stakeholders ven los resultados positivos, aumentando la credibilidad de la función de arquitectura en la organización. Esto es clave para el ciclo de cambio continuo.

Ahora que la nueva plataforma está en marcha, la labor se orienta a **mantenerla relevante** ante futuros cambios: es decir, la Fase H de cambio arquitectónico continuo.

3.9 FASE H – ARCHITECTURE CHANGE MANAGEMENT



Es la etapa final (y continua) del ciclo ADM, enfocada en **mantener la arquitectura empresarial relevante ante el cambio**. Una arquitectura no es un producto estático; una vez implementada, vive en un entorno dinámico donde surgen nuevas necesidades de negocio, tecnologías emergentes, cambios regulatorios o lecciones aprendidas. La fase H establece el proceso para **gestionar cambios** en la arquitectura de forma controlada, asegurando que la arquitectura evolucionada siga alineada con la estrategia empresarial y evitando la degradación con el tiempo.

En términos prácticos, se trata de instituir un ciclo de mejora continua: monitorizar, detectar necesidades de cambio, y lanzar iteraciones del ADM (ya sea completas o parciales) cuando corresponda

3.9.1 Objetivo

Garantizar la **vigencia y efectividad** de la arquitectura a lo largo del tiempo. Esto implica:

- **Monitorear el entorno interno y externo** para identificar factores que requieran ajustes en la arquitectura. Por ejemplo, cambios en la estrategia de negocio, nuevas oportunidades tecnológicas, feedback de desempeño de la arquitectura actual, incidencias recurrentes, cambios de mercado o competidores, etc.

- **Gestionar el proceso de solicitud de cambios arquitectónicos**: formalizar cómo se proponen, evalúan y deciden cambios una vez la arquitectura está en producción.
- **Implementar cambios menores** de manera ágil sin perder el control (p. ej., cambios evolutivos pequeños pueden gestionarse directamente bajo governance ligera), y **planificar cambios mayores** disparando nuevos ciclos del ADM cuando sea necesario.
- **Evitar la deriva arquitectónica**: mantener disciplina para que las soluciones implementadas no se desvíen con parches o soluciones ad-hoc sin pasar por análisis (una tentación común post-proyecto).
- **Continuar la gobernanza** de arquitectura en el BAU (Business-as-Usual): integrando la arquitectura empresarial en los procesos operativos de la organización (por ejemplo, participando en la planificación estratégica anual, en el intake de proyectos de TI, etc.).

3.9.2 Actividades clave

- **Establecer monitoreo continuo:** Ahora que la plataforma está viva, se define cómo se medirá su rendimiento y éxito. Por ejemplo, EcoShop instrumenta dashboards de KPIs: tiempos de respuesta, tasa de conversión, uso de nuevas funcionalidades, etc., y también monitorea el entorno: seguimiento de tendencias e-commerce, feedback de clientes, movimientos de competencia. El **Architecture Board** o un equipo asignado revisa periódicamente estos indicadores y contexto.
- **Proceso de Request for Architecture Work:** En TOGAF, cuando se detecta una necesidad de cambio significativa, se genera un **Request for Architecture Work** que esencialmente inicia un nuevo ciclo ADM enfocado en esa necesidad. En fase H se gestiona esto: se recibe una petición (p.ej., "El negocio quiere incorporar una nueva línea de producto con características personalizadas, ¿qué cambios requiere en la arquitectura?"). El Architecture Board analiza si es un cambio de **menor alcance** (que se puede resolver con modificaciones aisladas) o **mayor alcance** (que justifica un nuevo ciclo ADM formal desde Fase A).
- **Clasificar y responder a cambios:** No todos los cambios ameritan reiniciar todo el ADM. Se suelen clasificar:
    - **Cambios menores**: Ej., ajustar un parámetro de capacidad, agregar un servidor, actualizar una versión de software, etc. Estos se manejan bajo procedimientos de cambio estándar de TI con arquitectura dando visto bueno rápido.
    - **Cambios moderados**: Ej., añadir una funcionalidad importante que estaba fuera de alcance original pero sin alterar principios fundamentales. Quizá se puede lanzar una "iteración parcial" del ADM centrada en esa funcionalidad (por ejemplo, un mini-ciclo que arranca en Fase A para esa iniciativa específica).
    - **Cambios mayores/estratégicos**: Ej., una nueva visión de negocio que requiere reorientar en buena medida la arquitectura (como entrar en un nuevo modelo de

negocio, integrar una empresa adquirida, o adoptar tecnología disruptiva). Esto podría detonar un nuevo macro-ciclo ADM para evolucionar la arquitectura global.
.

EcoShop por ejemplo podría enfrentar en un futuro: la dirección decide lanzar una **plataforma marketplace** para terceros vendedores – eso es un cambio sustancial que implicaría un nuevo ciclo completo de arquitectura partiendo otra vez de Preliminary (para ajustar principios si hace falta) y luego Fase A con nueva visión, etc.

- **Actualizar la arquitectura continuamente:** Para cambios menores y moderados que se implementen, fase H asegura que la **documentación de arquitectura se actualiza** cada vez. Si en producción se decide añadir un microservicio extra, el repositorio se modifica. Se mantiene así la alineación entre la arquitectura "como está" y la documentación, evitando obsolescencia.
- **Re-evaluar principios y estándares:** Con el tiempo, algunos principios pueden quedar obsoletos o requerir ajustes. Fase H periódicamente (por ejemplo anual) revisa: ¿Siguen nuestros principios siendo adecuados? ¿Han surgido nuevas prioridades (ej., sostenibilidad ambiental de TI) que deban reflejarse en principios? Si se decide cambiar algún principio, se documenta y comunica formalmente (y aplicará en futuros ciclos).
- **Medir resultados y beneficios continuos:** Se compara el desempeño actual con el esperado. Si ciertos beneficios de la arquitectura no se están logrando, se investiga por qué: ¿fallo en la implementación? ¿cambiaron las condiciones? Esto podría generar mini-proyectos de optimización. Por ejemplo, si tras 1 año EcoShop ve que la personalización no incrementó ventas como se planeaba, quizás deciden ajustar el algoritmo de recomendaciones o hacer otro proyecto de mejora (eso es un cambio).
- **Comunicación de logros y cambios:** La función de arquitectura en fase H también difunde los éxitos: por ejemplo, presentar a dirección cómo la nueva plataforma incrementó X%. Esto refuerza la cultura de arquitectura. Asimismo, comunica a TI y negocio cualquier cambio planificado en arquitectura para alinear.
- **Integración con gobernanza corporativa:** Con la arquitectura en operación, fase H busca integrarse con el ciclo de planificación corporativo. Por ejemplo, asegurarse que en cada plan anual de TI, se considera la arquitectura (un representante del Architecture Board participa priorizando proyectos en portafolio TI, para alinearlos con la arquitectura y evitar esfuerzos desconectados).

3.9.3 Entradas típicas

- **Arquitectura implementada (nueva baseline):** La documentación actualizada tras fase G.
- **Monitoreo de desempeño**: KPI de negocio y TI, reportes de operaciones, tickets recurrentes de soporte (indicando áreas problemáticas), etc.

- **Estrategia de negocio actualizada:** Cualquier cambio en plan estratégico, nuevos objetivos, fusiones, expansiones, etc., que pudiera implicar cambios en TI.
- **Innovaciones tecnológicas:** Información sobre nuevas tecnologías que podrían ser relevantes (por ej., AI, IoT, etc.), proveniente de investigación o conferencias, que la función de arquitectura suele vigilar.
- **Feedback de usuarios/stakeholders:** Encuestas de satisfacción, retroalimentación de clientes, sugerencias de usuarios internos sobre mejoras al sistema.

3.9.4 Salidas clave (Entregables)

- **Registro de Solicitudes de Trabajo de Arquitectura:** Documentos que describen el caso de cambio, alcance y decisión (si se aprueba iniciar mini-proyecto de arquitectura).
- **Planes de actualización arquitectónica:** Si se aprueba un cambio moderado, puede resultar en un "plan de arquitectura incremental" que añade o modifica componentes (por ejemplo, documento de cómo integrar un nuevo módulo en la arquitectura existente).
- **Nueva iteración del ADM (si se lanza):** En caso de cambio mayor, la salida sería un nuevo ciclo completo para ejecutar, con su propia documentación (lo cual se convierte en un proyecto arquitectónico nuevo, volviendo a Preliminary para esa iteración).
- **Repositorio de arquitectura actualizado continuamente:** reflejando cualquier cambio realizado.
- **Reporte de estado de arquitectura periódicamente:** Podría ser un informe anual de Arquitectura Empresarial indicando: cambios realizados en el año, beneficios obtenidos, próximos retos (para informar a la alta dirección).
- **Plan de gestión de capacidad y mejoras:** A veces, salidas de fase H son planes proactivos, como un plan de capacidad (ej., "con el crecimiento actual, en 2 años necesitaremos escalar la base de datos o refactorizar tal módulo; proponer proyecto en el presupuesto del año siguiente").

3.9.5 Roles involucrados

- **Architecture Board permanente:** Sigue operando como guardián de la arquitectura durante la operación normal, reuniéndose con menor frecuencia (ej. trimestral) pero siempre disponible para evaluar cambios significativos.
- **Arquitectos Enterprise o de Solución:** Monitorean áreas específicas; por ejemplo, el arquitecto de infraestructura monitorea uso de servidores para anticipar escalamientos, el arquitecto de datos monitorea calidad de datos en el nuevo CRM.
- **Gestores de portafolio/proyectos:** Cualquier nuevo proyecto de TI que inicie, involucrar arquitectos para ver impacto en la arquitectura.

- **Stakeholders de negocio:** Pueden presentar nuevas demandas a la función de arquitectura (por ejemplo, marketing quiere una nueva herramienta – se discute en Architecture Board si encaja).
- **Operaciones de TI:** Proveen reportes de incidentes o problemas que puedan requerir cambios de arquitectura (ej., si cierto componente falla con frecuencia, quizá hay que rediseñarlo).
- **Equipo de innovación/tecnología emergente:** Si existe, colabora con arquitectura para introducir nuevas tecnologías de forma controlada cuando agregan valor.

3.9.6 Decisiones estratégicas

- **¿Cuándo iniciar un nuevo ciclo ADM?:** Probablemente la decisión más importante en fase H. Se evalúa el panorama: ¿La arquitectura actual sigue satisfaciendo los objetivos? ¿O se ha identificado una brecha estratégica que requiere reformular una parte sustancial? Por ejemplo, si EcoShop en unos años ve que el mercado exige una experiencia de realidad aumentada en compras, o la empresa decide internacionalizar fuertemente, podría ser hora de un nuevo ciclo de arquitectura para esa nueva visión. Decidir ese momento es estratégico, equilibrando no caer en "parálisis por análisis" (reiniciar muy pronto sin necesidad) ni en "obsolescencia" (esperar demasiado hasta que la arquitectura quede rezagada).
- **Cómo gestionar la obsolescencia tecnológica:** Con el tiempo, componentes implementados pueden quedar obsoletos (software EOL, etc.). Arquitectura debe planear actualizaciones. Decidir la estrategia de actualización continua vs reemplazos mayores es clave.
- **Priorizar inversiones de mejora vs nuevas iniciativas:** A veces, tras un gran programa, la empresa se enfoca en otras cosas. La fase H conlleva argumentar por inversiones en perfeccionar la arquitectura existente (refactoring, pagar deuda técnica) vs. solo invertir en nuevas funciones. Es estratégico convencer a negocio de equilibrar ambas.
- **Mantener o disolver la estructura de arquitectura:** Algunas organizaciones tras un gran proyecto disuelven el equipo. Estrategicamente, EcoShop debería mantener al menos un núcleo de arquitectura para fase H. Decidir la permanencia y roles es crucial para sostenibilidad.
- **Scale-up de la función EA:** Si la transformación tuvo éxito, tal vez deciden extender la práctica de arquitectura a otras áreas de la empresa. Ej., ahora abordar arquitectura de logística, etc. Esa es una decisión estratégica de madurez EA.

3.9.7 Ejemplo práctico (EcoShop) – Fase H

- **Operación continua:** EcoShop lanzó su nueva plataforma y pasó la vorágine del cambio. Ahora, Juan (arquitecto enterprise) se reúne trimestralmente con el Architecture Board para revisar la salud de la arquitectura. Tienen dashboards: ven que con crecimiento, la

base de datos llega al 70% de uso de CPU en peaks – un indicador a vigilar (quizá en 1 año habrá que escalar o particionar).

- **Feedback de negocio:** 6 meses post-implementación, el equipo de ventas sugiere una funcionalidad nueva: un sistema de **loyalty/recompensas** para clientes frecuentes. Esto no estaba en la arquitectura original. Se envía una **solicitud de trabajo** a arquitectura para evaluar. Architecture Board decide que esto es un cambio moderado (añadir un módulo de fidelización). Pueden gestionarlo con una **iteración parcial**: encargan a un arquitecto de aplicaciones diseñar cómo integrarlo sin afectar lo demás (quizá un nuevo microservicio de Loyalty que interactúa con el CRM y la web). No hace falta rehacer todo el ADM; se inicia en una mini-Fase A para esa capacidad, se llega a un diseño (Fase B/C focalizadas), se pasa a implementación directamente. Esto se documenta como extensión de la arquitectura y se mete en plan de TI del próximo trimestre.
- **Cambio tecnológico externo:** Un año después, surge una tecnología nueva de IA que promete mejorar recomendaciones considerablemente. Arquitectura investiga (fase H también promueve innovar). Deciden probarla. Hacen un prototipo en paralelo. Si resulta valiosa, el Board puede lanzar un **nuevo proyecto de mejora** para reemplazar el motor de recomendaciones actual con esta nueva IA. De nuevo, siguiendo governance: evalúan impacto, compatibilidad con principios, etc., antes de darle luz verde.
- **Estrategia de evolución:** Cada año, la arquitectura de EcoShop es revisada en conjunto con la planificación estratégica. Imaginemos que a los 3 años, la empresa planea expandirse a nuevos países. El Board evalúa: la arquitectura actual ¿soporta multidioma, multimoneda, etc.? Identifican algunas brechas (no consideraron localizaciones múltiples en diseño original). Este es un cambio mayor. Se decide iniciar un **nuevo ciclo ADM** enfocado en "Expansión Internacional", partiendo desde Fase A con nueva visión y adaptando la arquitectura en todas las capas para ser global (negocio: nuevos procesos de localización, datos: soporte multi-divisa, aplicaciones: instancias regionales quizás, tecnología: CDNs globales, etc.). Así, fase H hace de trampolín hacia un **ADM iterativo**: se vuelve a Preliminary (chequear capacidad para internacionalización), etc. Este nuevo ciclo se llamaría, por ejemplo, "Arquitectura Empresarial EcoShop 2.0", usando como baseline la actual.
- **Mantener disciplina:** Mientras, durante BAU, el Architecture Board insta a TI a no hacer cambios chapuceros: por ejemplo, si un desarrollador propone una rápida integración que no sigue estándares, el Board interviene: "por favor, presenta un diseño, lo evaluamos", evitando degradación. Esto a veces causa fricciones de velocidad vs calidad, pero con comunicación (recordando los fiascos evitados con buena arquitectura), EcoShop logra un balance.
- **Documentación viva:** Juan se asegura de actualizar el repositorio con cada cambio aprobado (incluyendo el módulo de loyalty, la eventual nueva IA, etc.). Así, la

documentación de arquitectura de EcoShop siempre refleja la realidad + los planes a futuro, sirviendo de referencia confiable.

- **Valor a largo plazo:** Gracias a fase H, EcoShop continúa cosechando beneficios: su arquitectura puede adaptarse con relativa facilidad a nuevas estrategias, porque tienen un proceso para ello. La empresa evita quedar atrapada en una plataforma rígida; en cambio, adopta una postura de mejora continua. Esto se traduce en ventajas competitivas sostenidas (pueden incorporar nuevos servicios más rápido que competidores con sistemas legados inflexibles).

3.9.8 Resultado esperado

La Fase H no tiene un fin definido como las anteriores; es un proceso continuo. Pero sus resultados se manifiestan en:

- **Longevidad de la arquitectura**: la plataforma e-commerce de EcoShop sigue vigente y eficiente varios años después, ya que se fue adaptando gradualmente en vez de volverse obsoleta.
- **Capacidad de respuesta al cambio**: la organización cuenta con un mecanismo establecido para evaluar y ejecutar cambios arquitectónicos, lo que reduce tiempos de reacción (no se improvisa, se sigue un camino claro).
- **Prevención de la entropía**: se evita que con el tiempo la arquitectura se "ensucie" con soluciones adhoc; en cambio, se mantiene estructurada.
- **Preparación para próximos ciclos**: eventualmente, cuando surja una transformación mayor, la organización y su equipo de arquitectura están experimentados y listos para iniciar un nuevo ciclo ADM con mejores prácticas aprendidas.

En conclusión, la Fase H asegura que la arquitectura empresarial de EcoShop permanece alineada con las necesidades del negocio en el tiempo, cerrando el ciclo actual del ADM y abriendo la puerta a futuros ciclos de mejora. **La arquitectura empresarial es un proceso continuo**, y con este enfoque, EcoShop institucionaliza la capacidad de reinventarse sin perder el control.

3.10 Conclusión de la Parte III

En esta tercera parte del libro hemos recorrido **todas las fases del ADM de TOGAF 10** aplicadas paso a paso a un caso práctico de una empresa de comercio electrónico. Desde preparar la organización y visualizar el futuro (Preliminary y Fase A), pasando por el diseño detallado de negocio, datos, aplicaciones y tecnología (Fases B, C, D), hasta la planificación, ejecución gobernada y gestión continua del cambio (Fases E, F, G, H), hemos visto cómo teoría y práctica convergen. El ejemplo de *EcoShop* nos permitió ilustrar cómo cada fase aporta valor tangible en una transformación real, y cómo el framework TOGAF guía de forma estructurada lo que de otro modo sería un proceso caótico. Esperamos que este recorrido práctico haya clarificado el **qué, cómo y por qué** de cada fase del ADM, preparando al lector para aplicar estos conceptos en sus propias organizaciones con una comprensión integral de la *arquitectura empresarial en acción*.

4 TOGAF E INTELIGENCIA ARTIFICAL

4.1.1 4.1. La IA como capability transversal del Architecture Development Method.

La Inteligencia Artificial (IA) se ha convertido en uno de los principales catalizadores de cambio en las organizaciones modernas. Su impacto no se limita a un dominio tecnológico concreto, ni puede reducirse a una simple herramienta de automatización. La IA afecta a la forma en que se diseñan procesos, se toman decisiones, se gobierna la información y se evolucionan las arquitecturas empresariales.

Ante este contexto, surge una pregunta inevitable para cualquier arquitecto empresarial:

¿Cómo encaja la Inteligencia Artificial dentro de TOGAF?

La respuesta corta es clara: TOGAF sigue siendo plenamente válido en la era de la IA.

La respuesta larga —y la que desarrollamos en este capítulo— es que la IA no sustituye al marco, sino que lo potencia, siempre que se integre de forma consciente, gobernada y alineada con sus principios fundamentales.

4.1.2 La IA no es un nuevo dominio arquitectónico

Con la aparición de tecnologías emergentes, es habitual caer en la tentación de crear nuevos dominios arquitectónicos: “Arquitectura de IA”, “Arquitectura de Automatización”, “Arquitectura Digital”, etc. Sin embargo, desde una perspectiva rigurosa de arquitectura empresarial, este enfoque suele generar más confusión que valor.

La Inteligencia Artificial **no rompe el modelo de dominios de TOGAF**:

- Se apoya en datos (entrenamiento, inferencia, calidad, gobierno).
- Se materializa mediante aplicaciones y servicios.
- Se ejecuta sobre infraestructura tecnológica.
- Habilita o transforma capacidades de negocio.

Por tanto, la IA no necesita un dominio propio para existir dentro de TOGAF.

En lugar de ello, debe entenderse como una capacidad transversal, que actúa sobre todos los dominios y fases del método ADM.

4.1.3 IA como capability transversal del ADM

El Architecture Development Method (ADM) es el corazón operativo de TOGAF. Su fortaleza reside en su carácter iterativo, estructurado y orientado a la toma de decisiones informadas. Precisamente por ello, el ADM es un terreno natural para la integración de la IA.

En este enfoque, la IA no actúa como un decisor autónomo, sino como un **asistente arquitectónico** que:

- Acelera el análisis.
- Reduce el trabajo mecánico.
- Detecta inconsistencias.
- Mejora la trazabilidad.
- Apoya la evolución continua de la arquitectura.

Principio clave

> La IA propone, analiza y asiste; la responsabilidad arquitectónica sigue siendo humana.

Este principio es esencial para mantener la gobernanza, la trazabilidad y la confianza en la práctica de arquitectura empresarial. Y completamente alineado con el concepto "Human In The Loop".

4.1.4 Aplicación de la IA en cada fase del ADM

4.1.4.1 Fase Preliminar – Preparar la organización

En esta fase, la IA puede utilizarse para:

- Analizar documentación existente (estrategia, procesos, sistemas).
- Detectar patrones de madurez arquitectónica.
- Identificar principios implícitos ya presentes en la organización.
- Apoyar la definición inicial del repositorio de arquitectura.

Ejemplo de uso:

- Análisis automático de documentos corporativos para identificar redundancias, contradicciones o áreas sin definición arquitectónica clara.

Resultado:

- Una preparación más rápida y objetiva de la capacidad de arquitectura empresarial.

4.1.4.2 Fase A – Architecture Vision

La IA puede asistir en:

- Generación de escenarios de arquitectura target.
- Evaluación comparativa de alternativas arquitectónicas.
- Redacción inicial de la visión arquitectónica y sus drivers.
- Identificación de riesgos y supuestos.

Ejemplo de uso:

- Simulación de distintos escenarios de transformación (cloud-first, data-driven, AI-enabled) y análisis de impacto.

Resultado:

- Visiones más informadas, coherentes y alineadas con los objetivos estratégicos.

4.1.4.3 Fase B – Business Architecture

En la arquitectura de negocio, la IA puede aportar un gran valor en:

- Descubrimiento de capacidades de negocio a partir de procesos existentes.
- Análisis de brechas entre estado actual y estado objetivo.
- Detección de ineficiencias operativas.
- Apoyo a la priorización de capacidades.

Ejemplo de uso:

- Extracción automática de mapas de capacidades a partir de documentación de procesos y entrevistas.

Resultado:

- Modelos de negocio más completos, coherentes y trazables.

4.1.4.4 Fase C – Arquitectura de Datos y Aplicaciones

En esta fase, la IA se integra de forma natural:

- **Arquitectura de Datos (C1):**
  - Análisis de calidad y redundancia de datos.
  - Propuesta de modelos conceptuales.
  - Identificación de flujos de datos críticos.
  - Apoyo a políticas de gobierno del dato.
- **Arquitectura de Aplicaciones (C2):**
  - Detección de solapamientos funcionales entre sistemas.
  - Propuesta de consolidación o modernización.
  - Generación de diagramas lógicos iniciales.

Resultado:

Diseños más consistentes y alineados con las necesidades reales del negocio.

4.1.4.5 Fase D – Technology Architecture

En la arquitectura tecnológica, la IA puede:

- Comparar tecnologías, plataformas y patrones.
- Analizar trade-offs técnicos (coste, escalabilidad, seguridad).
- Proponer arquitecturas de referencia.
- Detectar riesgos tecnológicos tempranos.

Ejemplo de uso:

- Evaluación asistida de alternativas cloud, plataformas de datos o servicios de IA gestionados.

Resultado:

- Decisiones técnicas mejor fundamentadas y documentadas.

4.1.4.6 Fases E y F – Opportunities, Solutions y Migration Planning

En estas fases, la IA apoya principalmente en:

- Identificación de building blocks reutilizables.
- Optimización de roadmaps de migración.
- Simulación de escenarios de entrega.
- Análisis de dependencias y riesgos.

Resultado:

- Planes de transición más realistas, eficientes y alineados con las capacidades organizativas.

4.1.4.7 Fase G – Implementation Governance

La IA puede utilizarse como mecanismo de verificación continua:

- Comprobación automática del cumplimiento de principios arquitectónicos.
- Validación de alineación entre diseño y ejecución.
- Soporte a revisiones de arquitectura.

Resultado:

- Gobernanza más objetiva, consistente y escalable

4.1.4.8 Fase H – Architecture Change Management

En la fase de gestión del cambio, la IA permite:

- Detección de desviaciones entre arquitectura definida y arquitectura real (architecture drift).
- Propuesta de evoluciones incrementales.
- Apoyo a la mejora continua del repositorio arquitectónico.

Resultado:

- Una arquitectura verdaderamente viva y evolutiva.

4.1.5 IA y el Architecture Content Framework (ACF)

La integración de la IA no altera el ACF, sino que lo potencia:

- Los entregables siguen existiendo.
- Los artefactos mantienen su estructura.
- Los building blocks siguen siendo reutilizables.

La diferencia es que ahora:

- Los artefactos pueden generarse, actualizarse y validarse de forma asistida.
- El repositorio de arquitectura se convierte en una fuente activa de conocimiento.
- La trazabilidad se refuerza automáticamente.

4.1.6 La IA como Architecture Co-Pilot

En este modelo, la IA actúa como un Architecture Co-Pilot, con responsabilidades claras:

- Generar borradores iniciales.
- Sugerir mejoras y alternativas.
- Detectar incoherencias.
- Mantener la consistencia documental.
- Acelerar el trabajo operativo del arquitecto.

La responsabilidad final, la toma de decisiones y la aprobación siguen siendo humanas y gobernadas por los mecanismos definidos en TOGAF (Architecture Board, principios, revisiones formales).

4.1.7 TOGAF y la Inteligencia Artificial no compiten; se complementan.

TOGAF aporta:

- Método.
- Gobernanza.
- Trazabilidad.
- Disciplina arquitectónica.

La IA aporta:

- Velocidad.
- Capacidad de análisis.
- Automatización inteligente.

Soporte a la evolución continua.

Cuando la IA se integra como una capability transversal del ADM, el resultado no es una arquitectura descontrolada, sino una arquitectura más viva, más consciente y más alineada con el negocio.

En este sentido, TOGAF no solo sigue siendo relevante en la era de la IA: se convierte en uno de los marcos más adecuados para gobernarla.

5 ANEXOS

5.1 CONTINUUM EMPRESARIAL Y RECURSOS TOGAF

TOGAF no solo se enfoca en el cómo desarrollar arquitectura (a través del ADM), sino también en el desde dónde construirla. Ahí es donde entra el concepto de Enterprise Continuum, un eje conceptual que guía la evolución arquitectónica desde modelos abstractos y reutilizables hasta soluciones específicas y concretas.

5.1.1 ¿Qué es el Enterprise Continuum?

El Enterprise Continuum es una herramienta mental que permite categorizar y organizar los activos arquitectónicos en un espectro que va desde lo más genérico hasta lo más particular.

Se compone de dos grandes subconjuntos:

Architecture Continuum

Enfocado en la arquitectura como visión y diseño abstracto.

Nivel	Ejemplo
Foundation Architecture	Estándares de interoperabilidad, arquitectura técnica base (como IaaS o protocolos REST)
Common Systems Architecture	Patrones comunes como microservicios, APIs compartidas, DevOps pipeline
Industry Architecture	Modelos típicos del sector (financiero, sanitario, energético)
Organization-Specific	Arquitectura concreta de una empresa o unidad de negocio

Solutions Continuum

Describe soluciones implementadas que derivan de la arquitectura.

Nivel	Ejemplo
Foundation Solutions	Productos base como Linux, Azure, PostgreSQL
Common Solutions	Plataformas compartidas (IAM corporativo, integración)
Industry Solutions	CRM para banca, ERP para energía, verticales sectoriales
Organization-Specific	CRM customizado, portales internos, soluciones a medida

El Architecture Continuum responde al "qué y por qué", mientras que el Solutions Continuum responde al "cómo y con qué"

5.1.2 ¿Cómo se usan en la práctica?

Cuando diseñamos una arquitectura:

1. Partimos de modelos abstractos (Foundation o Industry).
2. Adaptamos building blocks reutilizables.
3. Integramos soluciones propias y específicas.

Esto asegura:

- Coherencia con principios arquitectónicos comunes.
- Aceleración mediante reutilización de patrones.
- Facilidad de mantenimiento y evolución.

5.1.3 Building Blocks reutilizables

En el centro del Continuum están los Architecture Building Blocks (ABB) y los Solution Building Blocks (SBB), ya tratados en capítulos anteriores.

Tipo de bloque	Contenido	Nivel en el Continuum
ABB	Definición lógica, funcional	Architecture Continuum
SBB	Solución tecnológica implementada	Solutions Continuum

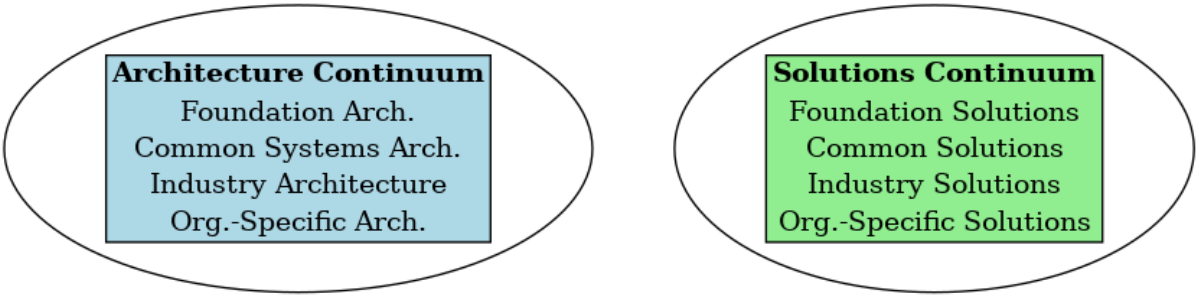
5.1.4 TOGAF Resource Base

TOGAF incluye una Resource Base o biblioteca que contiene recursos para implementar mejor el framework:

- Plantillas de deliverables (por ejemplo, Architecture Vision, Roadmap)
- Guías y checklists por fase del ADM
- Catálogos de artefactos y building blocks
- Técnicas de modelado y buenas prácticas
- Referencias cruzadas con otros marcos (ITIL, COBIT, SAFe)

Estos recursos no son obligatorios, pero son altamente recomendables para agilizar el trabajo y mantener coherencia.

5.1.5 Ejemplo



Idealmente, tu arquitectura concreta se nutre de elementos de ambos lados.

5.1.6 Resultado esperado

Con un uso efectivo del Enterprise Continuum:

- Se evita reinventar la rueda.
- Se potencia la interoperabilidad.
- Se acelera la generación de arquitectura aplicando patrones validados.
- Se establece una arquitectura sostenible, escalable y alineada con el negocio.

5.2 MÉTODO ARCDOC+

ArcDoc+

by jmfloreszazo.com

ArcDoc+ es un método práctico de documentación arquitectónica diseñado para complementar TOGAF, integrando distintos modelos de representación de forma coherente y ligera. No pretende sustituir al Architecture Content Framework, sino **operacionalizarlo** en el día a día de los proyectos.

El objetivo principal de ArcDoc+ es reducir la fricción entre arquitectura, diseño y ejecución, proporcionando una estructura clara de documentación que sea:

- Fácil de mantener.
- Comprensible para distintos perfiles.

- Directamente utilizable en proyectos reales.

Principios de ArcDoc+

ArcDoc+ se apoya en los siguientes principios:

- Documentar lo necesario, no todo.
- Un artefacto, un propósito claro.
- Consistencia visual y semántica.
- Trazabilidad entre negocio, sistemas y tecnología.

Componentes principales

ArcDoc+ combina distintos enfoques de documentación según la fase del ADM:

- Markdown estructurado: Para documentos de visión, principios, decisiones y descripciones textuales.
- Modelo C4: Para representar arquitectura de sistemas a distintos niveles (contexto, contenedores, componentes).
- UML (cuando aplica): Para secuencias, estados o casos donde el comportamiento es relevante.
- Modelos de procesos (BPMN / Capability Maps): Para arquitectura de negocio.
- Requisitos con Gherkins: Para describir features como comportamientos verificables y trazables, usando un lenguaje estructurado (Given/When/Then) que conecta negocio, arquitectura, desarrollo y tests, y permite convertir requisitos funcionales en contratos claros a lo largo de todo el ADM.

Relación con TOGAF

ArcDoc+ no introduce nuevos entregables fuera de TOGAF. Simplemente:

- Da formato práctico a los artefactos definidos en el ACF.
- Facilita su almacenamiento en el repositorio de arquitectura.
- Permite su evolución a lo largo del ADM.

En este sentido, ArcDoc+ actúa como un puente entre el marco teórico y la práctica real, ayudando a que la arquitectura no se convierta en un ejercicio puramente documental, sino en un activo vivo y reutilizable.

5.2.1 Qué NO es ArcDoc+

Ahora ArcDoc+ queda muy bien definido por lo que es, pero no por lo que no es.

Una sola frase evita malentendidos.

> *ArcDoc+ no es una metodología de desarrollo ni un framework de gobierno adicional, sino una forma pragmática de materializar los artefactos de TOGAF en formatos vivos y accionables.*

5.2.2 Arquitectura viva

> *Los artefactos generados con ArcDoc+ están pensados para versionarse y evolucionar junto al sistema, manteniendo la trazabilidad entre decisiones arquitectónicas, requisitos y soluciones implementadas.*

5.2.3 ArcDoc+ ↔ TOGAF

Elemento ArcDoc+	Relación con TOGAF
Markdown	Deliverables y artefactos del ACF
C4 Model	Arquitecturas de aplicaciones y tecnología
BPMN / Capability Maps	Arquitectura de negocio
UML	Comportamiento y flujos
Gherkin	Requirements Management + validación

5.2.4 Documentación TOGAF + C4 Model + UML

Fase TOGAF	Objetivo	C4 Model	UML de Secuencia	Notas
Preliminary	Definición de principios, estructura EA y gobernanza	-	-	Organigramas o matrices de responsabilidad
Fase A – Architecture Vision	Visión de alto nivel de la arquitectura target	C4-1 (Context)	-	Mostrar actores externos e interfaces clave
Fase B – Business Architecture	Modelado de capacidades, procesos y actores	-	-	Usar BPMN o Capability Maps
Fase C1 – Data Architecture	Diseño de entidades, flujos e integridad de datos	-	-	ERD o modelos conceptuales si es necesario
Fase C2 – Application Architecture	Mapa de aplicaciones, componentes e integraciones	C4-2 (Container), C4-3 (Component)	✓	Representa microservicios, APIs, etc.
Fase D – Technology Architecture	Infraestructura física/lógica para soportar apps	Infraestructura extendida tipo C4	-	Opcional si se necesita detalle de deployment

Fase E – Opportunities and Solutions	Identificación y priorización de soluciones	C4-3 refinado	✓	Para ver cómo encajan soluciones o building blocks
Fase F – Migration Planning	Planificación de la transición de estados actuales a futuros	-	-	Usar timelines o diagramas de transición
Fase G – Implementation Governance	Gobierno de implementación y conformidad arquitectónica	Validación de C4-2 y C4-3	✓	UML útil para ver comportamiento y conformidad
Fase H – Architecture Change Management	Gestión del cambio y evolución de la arquitectura	Comparativa entre C4 actuales y futuros	-	Visualizar evolución de módulos
Requirements Management	Trazabilidad entre requisitos y componentes arquitectónicos	Vincular C4-3 con requisitos	-	Notas sobre qué módulo satisface qué necesidad

5.3 Guía para Responder RFPs Usando TOGAF

De los Requisitos a la Arquitectura Propuesta.

Si nos pidieran responder a una RFP como la de **EcoShop**, esta sería una excelente oportunidad para aplicar todo lo aprendido y estructurar la respuesta de forma estratégica, alineada con TOGAF. Desde la definición de objetivos hasta la propuesta técnica y el roadmap de entrega, cada sección de la respuesta puede vincularse con una fase del ADM de TOGAF, ofreciendo claridad, trazabilidad y coherencia.

Este enfoque no es casual: en el documento "Request for Proposal - Una guía desde el otro lado V1.0.0", exploré precisamente cómo se construyen las RFP desde el lado del cliente. Y es que cuando entiendes cómo se formula una RFP, también sabes cómo debes responderla. Responder bien no es rellenar una plantilla: es demostrar que dominas el contexto, las necesidades y sabes traducirlas en una solución viable y sostenible.

Por eso, este anexo de TOGAF aplicado a RFPs busca ser algo más que un checklist: es un puente entre el pensamiento arquitectónico y la acción consultiva. Una herramienta para responder no solo con estructura, sino con visión.

5.3.1 Entender el punto de partida

Antes de proponer una arquitectura, es fundamental analizar el contexto del cliente. Esto incluye:

- Identificar objetivos estratégicos (TOGAF: Fase Preliminary / A)
- Analizar restricciones tecnológicas o de arquitectura
- Validar supuestos implícitos en la RFP

Consejo: Redacta una visión técnica que muestre cómo la propuesta se alinea con los objetivos del cliente.

5.3.2 Traducir requisitos en capacidades y arquitectura

El siguiente paso es convertir los requerimientos en una arquitectura lógica y técnica, basada en TOGAF:

- Clasifica los requerimientos por capacidades (Fase B)
- Define la arquitectura de aplicaciones y datos (Fase C1 y C2)
- Propón tecnologías y diseño técnico base (Fase D)

Consejo: Usa diagramas C4 (niveles 1 a 3) para explicar visualmente la solución propuesta.

5.3.3 Abordar ejecución, riesgos y transición

Más allá del diseño, una buena propuesta debe mostrar cómo se implementará y gobernará la solución:

- Identifica building blocks reutilizables o SaaS (Fase E)
- Define roadmap, MVP y fases de entrega (Fase F)
- Incluye enfoque de gobernanza técnica (Fase G)

Consejo: Define entregables por fase y cómo evitarás desviaciones de alcance y coste.

5.3.4 Trazabilidad de requisitos y cobertura

Una propuesta sólida debe dejar claro cómo cada requerimiento será cubierto y cómo evolucionará la solución:

- Crea una tabla de trazabilidad Req ↔ Componente (Requirements Mgmt)
- Explica cómo escalará o evolucionará sin rehacer desde cero (Fase H)

5.3.5 Plantilla sugerida de respuesta a RFP (alineada con TOGAF)

Sección	TOGAF	¿Obligatoria?	Notas
Visión y Objetivos del Proyecto	Fase A	Obligatoria	Necesario para demostrar alineación con el negocio del cliente.
Contexto Actual / Arquitectura As-Is	Fase B	Obligatoria	Ayuda a mostrar comprensión de la situación inicial del cliente.
Arquitectura Target y Diseño Lógico	Fase C	Obligatoria	Debe contener el diseño lógico propuesto. Usa diagramas C4.
Infraestructura y Herramientas	Fase D	Obligatoria	Describe tecnología, cloud, patrones técnicos.
Oportunidades y Soluciones	Fase E	Obligatoria	Muestra decisiones de diseño: qué se reutiliza, compra o construye.
Roadmap de Migración y Entregables	Fase F	Obligatoria	Imprescindible para planificar entregas, fases o MVP.
Gobernanza y Roles	Fase G	Recomendado	Útil si se incluyen tareas de ejecución o soporte.
Gestión del Cambio e Iteración Post Go-Live	Fase H	Recomendado	Relevante si se menciona mantenimiento o evolución futura.

Mínimo obligatorio para una respuesta sólida:

- Fase A (Visión)
- Fase B (Contexto actual)
- Fase C (Arquitectura lógica)
- Fase D (Tecnología)
- Fase E (Justificación de solución)
- Fase F (Plan de entrega)

Puedes omitir si no lo piden expresamente:

- Fase G (Gobierno del proyecto) → Si el cliente ya tiene gobierno interno definido.
- Fase H (Gestión del cambio) → Si no hay ciclo post-proyecto o soporte incluido.

5.3.6 Resumen

Sección de la respuesta	Qué incluir	Relación con TOGAF
Visión y objetivos	Lo que busca el cliente y cómo lo abordarás	Fase A
Alcance funcional	Capacidades cubiertas, límites, casos de uso	Fase B
Arquitectura propuesta	Diagramas lógicos y técnicos (C4)	Fase C y D
Enfoque de entrega	Fases, MVPs, roadmap, gestión de cambios	Fase E, F, H
Gestión de riesgos	Técnicos, organizativos, dependencias	Fase G
Trazabilidad	Matriz Req ↔ Solución	Requirements Management
Valor añadido	Optimización, TCO, escalabilidad	Fase E