**DPTO. INFORMATICA - I.E.S. SAN SEBASTIÁN MÓDULO PROYECTO**

**C.F.G.S. ADMINISTRATICIÓN DE SISTEMAS INFORMÁTICOS Y EN RED**

Un dibujo de una cara feliz

El contenido generado por IA puede ser incorrecto.

**Aplicación web de alta disponibilidad y escalabilidad**

**Autor:** Sergio García Márquez

**Fecha:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tutor:** Manuel Lucero Sánchez

Índice

[1. Introducción 5](#_Toc198737503)

[1.1. Introducción a la memoria 8](#_Toc198737504)

[1.2. Descripción 9](#_Toc198737505)

[1.3. Objetivos generales 10](#_Toc198737506)

[1.4. Beneficios 11](#_Toc198737507)

[1.5. Motivaciones personales 12](#_Toc198737508)

[2. Estudio de viabilidad 13](#_Toc198737509)

[2.1. Introducción 13](#_Toc198737510)

[2.1.1. Tipología y palabras clave 14](#_Toc198737511)

[2.1.2. Descripción 15](#_Toc198737512)

[2.1.3. Objetivos del proyecto 17](#_Toc198737513)

[2.1.4. Clasificación de los objetivos 19](#_Toc198737514)

[2.1.5. Definiciones, acrónimos y abreviaciones 21](#_Toc198737515)

[2.1.6. Referencias 22](#_Toc198737516)

[2.1.7. Documentación del proyecto 23](#_Toc198737517)

[2.2. Estudio de la situación actual 24](#_Toc198737518)

[2.2.1. Contexto 24](#_Toc198737519)

[2.2.2. Lógica del sistema 25](#_Toc198737520)

[2.2.3. Descripción física 26](#_Toc198737521)

[2.2.4. Diagnóstico del sistema actual 28](#_Toc198737522)

[2.2.5. Normativa y legislación 29](#_Toc198737523)

[2.3. Requisitos del sistema 30](#_Toc198737524)

[2.3.1. Requisitos 30](#_Toc198737525)

[2.3.2. Restricciones del sistema 32](#_Toc198737526)

[2.3.3. Catalogación y priorización de los requisitos 34](#_Toc198737527)

[2.4. Alternativas y selección de la solución 35](#_Toc198737528)

[2.4.1. Alternativa 1: Arquitectura mínima funcional 35](#_Toc198737529)

[2.4.2. Alternativa 2: Arquitectura escalable y distribuida (elegida) 36](#_Toc198737530)

[2.4.3. Conclusiones 37](#_Toc198737531)

[2.5. Planificación del proyecto 38](#_Toc198737532)

[2.5.1. Recursos del proyecto 38](#_Toc198737533)

[2.5.2. Tareas del proyecto 39](#_Toc198737534)

[2.6. Evaluación de riesgos 40](#_Toc198737535)

[2.6.1. Lista de riesgos 40](#_Toc198737536)

[2.6.2. Catalogación de riesgos 40](#_Toc198737537)

[2.6.3. Plan de contingencia 41](#_Toc198737538)

[2.7. Presupuesto 42](#_Toc198737539)

[2.7.1. Estimación de costes materiales y personales 42](#_Toc198737540)

[2.7.2. Resumen y análisis coste-beneficio 43](#_Toc198737541)

[2.8. Conclusiones 44](#_Toc198737542)

[2.8.1. Beneficios 44](#_Toc198737543)

[2.8.2. Inconvenientes 45](#_Toc198737544)

[3. Análisis 46](#_Toc198737545)

[3.1. Introducción 46](#_Toc198737546)

[3.2. Requisitos funcionales de usuarios 46](#_Toc198737547)

[3.3. Requisitos no funcionales 47](#_Toc198737548)

[3.4. Diagrama del sistema 48](#_Toc198737549)

[3.5. Conclusión del análisis 48](#_Toc198737550)

[4. Diseño 49](#_Toc198737551)

[4.1. Introducción 49](#_Toc198737552)

[4.2. Selección de la arquitectura de red 49](#_Toc198737553)

[5. Implementación 50](#_Toc198737554)

[5.1. Fase 1: Creación de una aplicación web funcional básica 50](#_Toc198737555)

[5.2. Fase 2: desacoplamiento de los componentes de la aplicación 56](#_Toc198737556)

[5.3. Fase 3: implementar la alta disponibilidad y escalabilidad 62](#_Toc198737557)

[6. Implementación extra 66](#_Toc198737558)

[6.1. Introducción 66](#_Toc198737559)

[6.2. CloudFormation 67](#_Toc198737560)

# Introducción

La transformación digital ha impulsado de manera vertiginosa la adopción de infraestructuras en la nube como solución preferente para el despliegue de servicios y aplicaciones empresariales. En este contexto, la arquitectura de aplicaciones que garanticen alta disponibilidad, escalabilidad y resiliencia se ha convertido en una competencia esencial para los profesionales de las tecnologías de la información. Este Trabajo de Fin de Grado se enmarca en ese paradigma, y tiene como objetivo la planificación, construcción y despliegue de una aplicación web de arquitectura cloud basada en Amazon Web Services (AWS), cumpliendo con los requisitos de disponibilidad continua, escalabilidad automática, seguridad estructurada y eficiencia operativa.

El proyecto ha sido diseñado como una experiencia de aprendizaje completa y progresiva, en la que yo, como estudiante, asumo el rol de arquitecto de soluciones cloud. A través de varias fases, se aborda el ciclo completo de desarrollo e implementación de una solución realista, aplicando buenas prácticas de ingeniería de sistemas y utilizando servicios ampliamente demandados en el sector, como Amazon EC2, Amazon RDS, Auto Scaling, Elastic Load Balancer, VPC, Security Groups, Route Tables, entre otros.

Desde una perspectiva educativa, este proyecto permite evaluar de forma integral los conocimientos y habilidades adquiridos a lo largo del ciclo formativo, integrando áreas como:

* Administración y configuración de redes virtuales.
* Seguridad de la infraestructura cloud.
* Automatización del despliegue de servicios.
* Diseño de arquitecturas resilientes y eficientes.
* Estimación y optimización de costes en la nube.
* Comprensión del modelo de responsabilidad compartida y la gestión de servicios gestionados.

El desarrollo del proyecto se realiza en un entorno de laboratorio de larga duración, con persistencia de recursos en AWS, lo que permite al estudiante iterar, reflexionar y refinar su solución a lo largo de varios días, tal y como ocurre en un entorno de trabajo profesional. Además, se fomenta la autonomía del alumno mediante la toma de decisiones técnicas, el uso de documentación oficial, y la búsqueda de soluciones ante problemas inesperados, fortaleciendo así su capacidad de análisis y resolución de incidencias.

El proyecto se estructura en fases bien definidas, que permiten una evaluación continua:

* **Planificación y diseño arquitectónico**: definición de la infraestructura, diseño del diagrama de arquitectura y estimación de costes utilizando la herramienta oficial *AWS Pricing Calculator*.
* **Despliegue inicial**: creación de la red virtual privada (VPC), subredes, gateway de internet, tablas de rutas y una instancia EC2 con una aplicación web básica.
* **Mejora de la solución**: desacoplamiento de componentes, migración de la base de datos a *Amazon RDS*, implementación de balanceo de carga y escalado automático.
* **Pruebas, validación y presentación**: documentación de la solución, evaluación del rendimiento y fiabilidad, y presentación final de resultados.

Cabe destacar que este tipo de proyectos no solo permite validar competencias técnicas, sino que también facilita el desarrollo de competencias transversales como el trabajo autónomo, la capacidad de organización, la comunicación técnica y la presentación de resultados de forma clara y profesional.

Este Trabajo de Fin de Grado proporciona un contexto idóneo para consolidar conocimientos en administración de sistemas en la nube, fomentar la autonomía técnica del estudiante y prepararlo para un entorno laboral donde el *cloud computing* representa una competencia clave. Desde el punto de vista docente, supone una herramienta excelente para medir la madurez técnica y metodológica del alumno, ofreciendo una evaluación significativa y alineada con las necesidades actuales del sector TIC.

## Introducción a la memoria

Esta memoria documenta el proceso completo de diseño, despliegue y evaluación de una arquitectura cloud para una aplicación web de alta disponibilidad, implementada en *Amazon Web Services* (AWS). El proyecto se enmarca en el Trabajo de Fin de Grado del ciclo formativo de Administración de Sistemas Informáticos en Red, y tiene como objetivo principal la integración y aplicación de conocimientos técnicos adquiridos a lo largo del itinerario formativo.

A través de esta propuesta se ha querido simular un entorno profesional real, abordando problemáticas frecuentes en el ámbito empresarial actual, como la necesidad de garantizar la disponibilidad continua de servicios, la gestión eficiente de recursos, la seguridad de la infraestructura y la escalabilidad ante picos de demanda.

Este trabajo se desarrolla como parte de un laboratorio extensivo guiado por docentes, pero en el que yo, como estudiante, tendré autonomía para diseñar mi propia solución, tomar decisiones técnicas justificadas y reflexionar sobre las implicaciones prácticas y económicas de cada elección.

## Descripción

El proyecto consiste en la construcción de una solución completa en la nube, desplegando una aplicación web funcional que cumple con los principios fundamentales de las arquitecturas modernas: alta disponibilidad, escalabilidad horizontal, tolerancia a fallos, y separación de responsabilidades entre los distintos servicios.

La solución final incluye:

* El diseño y despliegue de una red virtual privada (VPC) con múltiples subredes distribuidas en distintas zonas de disponibilidad (AZ).
* El lanzamiento de instancias EC2 para la capa de aplicación, con autoscaling configurado para responder dinámicamente a la carga.
* La utilización de un balanceador de carga (Elastic Load Balancer) para distribuir tráfico y mejorar la tolerancia a fallos.
* La migración de la base de datos local a un servicio gestionado (Amazon RDS), con opciones de alta disponibilidad y copias de seguridad.
* El diseño de reglas de seguridad, NAT gateway, y configuraciones de routing para garantizar un tráfico seguro y controlado.
* Además, se realiza una planificación inicial de costes usando la herramienta AWS Pricing Calculator, permitiendo tener una estimación aproximada del gasto mensual que supondría mantener esta infraestructura en un entorno de producción real.

## Objetivos generales

Los principales objetivos del trabajo son los siguientes:

* **Diseñar una arquitectura cloud escalable y segura** para una aplicación web básica, empleando buenas prácticas de AWS.
* **Aplicar de forma práctica los conocimientos adquiridos** en administración de sistemas, redes, virtualización y servicios cloud.
* **Fomentar la toma de decisiones técnicas razonadas**, valorando aspectos como coste, rendimiento, resiliencia y mantenibilidad.
* **Automatizar parcialmente los despliegues**, para simular flujos de trabajo reales de DevOps.
* **Realizar una documentación clara y profesional**, que permita reproducir, mantener y escalar la solución diseñada.

## Beneficios

Este proyecto aporta una serie de beneficios claros tanto a nivel formativo como profesional:

* Consolidación de conocimientos: permite integrar múltiples módulos del ciclo en un solo proyecto coherente.
* Preparación para entornos reales: simula situaciones típicas en empresas tecnológicas que operan en la nube.
* Contacto directo con tecnologías de referencia: como Amazon EC2, RDS, Auto Scaling, y VPC, todas ellas ampliamente utilizadas en el sector.
* Potencial de mejora continua: la arquitectura diseñada puede servir como base para futuras ampliaciones como monitorización, seguridad avanzada, uso de contenedores, etc…
* Mejora de la empleabilidad: proporciona una experiencia tangible y demostrable en uno de los entornos más demandados por empresas del ámbito TIC.

## Motivaciones personales

Desde el inicio del ciclo formativo, uno de mis principales intereses ha sido comprender cómo se diseñan y gestionan infraestructuras a gran escala, especialmente aquellas que deben mantenerse disponibles y operativas las 24 horas del día, no es algo que use en las prácticas que estoy realizando, pero al ser Atlantic Copper una empresa de gran volumen de trabajadores, es útil saber cosas de escalabilidad. La computación en la nube representa un cambio de paradigma en la administración de sistemas, y me resulta especialmente atractiva por su flexibilidad, su potencia y su enfoque en la automatización.

Elegí este proyecto porque me permite aplicar todo lo aprendido de una manera práctica, tangible y orientada al mundo real. Además, trabajar con tecnologías como AWS, que son utilizadas por empresas de todos los tamaños y sectores, representa una oportunidad para adquirir competencias con alta demanda en el mercado laboral actual.

También me ha motivado la posibilidad de desarrollar un proyecto completo por fases, en el que puedo planificar, construir, probar y documentar una solución que no solo funcione técnicamente, sino que también sea defendible desde el punto de vista económico y estratégico.

# Estudio de viabilidad

## Introducción

El presente apartado tiene como objetivo analizar la viabilidad técnica, organizativa y funcional del proyecto, proporcionando una base sólida sobre la cual se justifica su desarrollo. Se examinan la naturaleza del trabajo, las tecnologías implicadas, los objetivos planteados, las partes interesadas, el estado actual del entorno, las posibles limitaciones y el marco normativo aplicable.

El proyecto consiste en el diseño, despliegue y validación de una infraestructura completa en la nube sobre la plataforma Amazon Web Services (AWS), cuyo fin es alojar una aplicación web funcional con características de alta disponibilidad y escalabilidad. Esta solución representa un entorno realista, similar al que podrían implementar empresas que requieren garantizar la continuidad del servicio y adaptarse a variaciones de carga, sin comprometer el rendimiento ni la seguridad.

Para garantizar la coherencia y el éxito del proyecto, se parte de una planificación inicial basada en buenas prácticas, que incluye la definición de componentes clave (instancias virtuales, redes privadas, balanceadores, bases de datos gestionadas), una estimación detallada de costes y una serie de pruebas funcionales que aseguren la correcta implementación de los servicios.

Este estudio no solo permite validar que el proyecto es técnicamente viable, sino que también ofrece una visión global del impacto que tendría su implantación en términos de costes, operatividad, mantenimiento y escalabilidad futura.

## Tipología y palabras clave

Este proyecto se enmarca en la tipología de despliegue de infraestructura como servicio (IaaS) y plataforma como servicio (PaaS) en entornos cloud. Específicamente, se orienta a la construcción y configuración de una arquitectura cloud segura, escalable y tolerante a fallos sobre AWS, con capacidad de alojar una aplicación web que pueda operar en un entorno productivo.

La propuesta se basa en los pilares fundamentales de la arquitectura moderna en la nube: alta disponibilidad, escalabilidad automática, segmentación lógica de redes, uso de servicios gestionados y control de costes. Para su implementación, se hace uso de una combinación de servicios que permiten automatizar tareas de despliegue, controlar el flujo del tráfico, proteger los datos y ofrecer una experiencia de usuario fluida y fiable.

Los conocimientos aplicados en este trabajo abarcan áreas clave de la administración de sistemas y redes, como la configuración de redes virtuales, gestión de instancias EC2, implementación de grupos de seguridad, administración de bases de datos con RDS, y configuración de Auto Scaling Groups y Load Balancers.

## Descripción

El presente trabajo tiene como finalidad diseñar, desplegar y validar una arquitectura completa en la nube utilizando los servicios que proporciona Amazon Web Services (AWS), con el objetivo de alojar una aplicación web funcional que cumpla con los principios de alta disponibilidad (HA), escalabilidad horizontal, seguridad robusta y eficiencia operativa.

El entorno por implementar debe ser capaz de garantizar el funcionamiento continuo del servicio ante posibles fallos de infraestructura o picos inesperados de tráfico. Para lograrlo, se construye una solución compuesta por diversos componentes desacoplados que se comunican entre sí a través de una red virtual privada (VPC), estructurada por zonas de disponibilidad (AZ) y protegida mediante reglas de enrutamiento y grupos de seguridad personalizados.

El sistema incluye:

* Una capa de cómputo basada en instancias EC2 distribuidas en distintas subredes públicas.
* Una capa de base de datos gestionada mediante Amazon RDS, ubicada en subredes privadas para reforzar la seguridad.
* Un balanceador de carga (Elastic Load Balancer) para repartir el tráfico entre las distintas instancias.
* Un grupo de escalado automático (Auto Scaling Group) para aumentar o reducir la capacidad de cómputo según la demanda.
* Una estimación de costes realista realizada con AWS Pricing Calculator, que permite conocer el impacto económico del despliegue en un entorno real.

## Objetivos del proyecto

El proyecto se orienta hacia una serie de objetivos definidos que guían su desarrollo y validación. Estos objetivos abarcan tanto aspectos técnicos como formativos, y se dividen en generales y específicos.

El objetivo general es diseñar y desplegar una arquitectura cloud segura, escalable y tolerante a fallos, utilizando los servicios nativos de AWS.

En cuanto a objetivos específicos, son los siguientes:

* **Construir una red virtual privada** (VPC) que permita segmentar recursos entre subredes públicas y privadas, estableciendo una topología segura y eficiente.
* **Desplegar una aplicación web funcional en instancias EC2**, asegurando su acceso mediante direccionamiento público controlado.
* **Migrar y desacoplar la base de datos**, ubicándola en un servicio gestionado (Amazon RDS) para facilitar su administración, respaldo y disponibilidad.
* **Implementar un sistema de escalado automático**, mediante Auto Scaling Groups, que permita añadir o eliminar instancias en función de la carga del sistema.
* **Añadir un balanceador de carga** (ELB) para repartir de forma equitativa las peticiones entrantes y garantizar tolerancia a fallos entre zonas de disponibilidad.
* **Realizar una estimación de costes detallada utilizando AWS Pricing Calculator**, evaluando el impacto económico de cada servicio involucrado.
* **Aplicar medidas de seguridad estructuradas**, como grupos de seguridad, control de tráfico entrante y saliente, y segregación de funciones.
* **Documentar el proceso completo de diseño e implementación**, con capturas, esquemas y justificaciones técnicas, para facilitar su comprensión y replicación.

Estos objetivos no solo buscan la construcción de un sistema funcional, sino también el desarrollo de habilidades de análisis, planificación, gestión de recursos cloud y documentación técnica.

## Clasificación de los objetivos

|  |  |
| --- | --- |
| Plazo | Objetivos |
| Corto plazo | Crear la red virtual privada (VPC) con su correspondiente puerta de enlace a Internet.  Comprobar la accesibilidad pública de la aplicación y su correcto funcionamiento. |
| Medio Plazo | Separar la base de datos de la instancia de aplicación, migrándola a Amazon RDS.  Desacoplar la arquitectura para facilitar el mantenimiento, la recuperación y el escalado. |
| Largo plazo | Implementar Auto Scaling para garantizar la escalabilidad horizontal.  Añadir un balanceador de carga (ELB) para distribuir tráfico entre instancias.  Optimizar costes, reducir puntos únicos de fallo y mejorar el rendimiento general del sistema. |

La ejecución del proyecto se estructura en base a distintos horizontes temporales y niveles de prioridad, lo que permite establecer una ruta lógica de desarrollo y asegurar la consecución de metas intermedias antes de alcanzar el objetivo final.

* Según el plazo temporal:
* Según la prioridad funcional:

|  |  |
| --- | --- |
| Tipo de objetivos | Objetivos |
| Prioritarios | Garantizar la alta disponibilidad de la aplicación mediante redundancia y distribución.  Asegurar la funcionalidad mínima viable (MVP): una aplicación accesible y estable.  Establecer mecanismos de escalabilidad para adaptarse a variaciones de carga. |
| Secundarios | Optimizar la experiencia de usuario mediante una configuración de red más fluida.  Explorar ampliaciones futuras, como la monitorización, alertas o integración con otros servicios (S3, CloudWatch, etc.). |

## Definiciones, acrónimos y abreviaciones

| **Término** | **Definición** |
| --- | --- |
| AWS | **Amazon Web Services**: plataforma de servicios en la nube. |
| EC2 | **Elastic Compute Cloud**: servicio de instancias virtuales. |
| VPC | **Virtual Private Cloud**: red virtual privada en AWS. |
| RDS | **Relational Database Service**: servicio de bases de datos gestionadas. |
| ELB | **Elastic Load Balancer**: balanceador de carga para distribuir tráfico. |
| AZ | **Availability Zone**: ubicación geográfica independiente dentro de una región de AWS. |
| Auto Scaling | Servicio que ajusta automáticamente el número de instancias según la demanda. |
| CIDR | **Classless Inter-Domain Routing**: notación para definir rangos de direcciones IP. |
| SG | **Security Group:** firewall virtual para controlar el tráfico en AWS. |
| IAM | **Identity and Access Management**: gestión de usuarios y permisos en AWS. |

## Documentación del proyecto

A lo largo del proyecto se genera diversa documentación técnica, destinada a describir y justificar cada uno de los procesos llevados a cabo:

* **Memoria técnica**: Documento principal que recoge la planificación, implementación, análisis de riesgos, estimación de costes y conclusiones finales.
* **Diagrama de arquitectura**: Representación gráfica de los componentes desplegados y sus interconexiones (VPC, instancias EC2, RDS, ELB, etc.).
* **Capturas de pantalla**: Evidencias visuales del estado de la infraestructura, configuraciones y pruebas funcionales.
* **Presupuesto estimado**: Documento generado mediante *AWS Pricing Calculator*, detallando el coste previsto para cada recurso.
* **Scripts de configuración**: Scripts de automatización o configuraciones utilizadas en el User Data de EC2 o en la creación de recursos.
* **Plan de riesgos**: Documento donde se identifican y catalogan posibles riesgos asociados a la implantación de la solución.

## Estudio de la situación actual

## Contexto

Antes de la ejecución de este proyecto, los entornos tradicionales de despliegue de aplicaciones en instituciones educativas o pequeñas empresas solían basarse en servidores locales o en soluciones de hosting compartido. Estos sistemas, aunque válidos en escenarios de baja carga, presentaban múltiples limitaciones:

* **Baja tolerancia a fallos**. La caída de un servidor implicaba la caída total del servicio.
* **Escalabilidad limitada**. Para soportar más usuarios, era necesario intervenir manualmente añadiendo recursos físicos o migrando a nuevas infraestructuras.
* Mantenimiento costoso tanto en tiempo como en recursos humanos.
* Falta de segmentación de redes, lo que aumentaba los riesgos de seguridad.

El crecimiento de la demanda de accesibilidad global, la necesidad de responder a variaciones de tráfico y la importancia crítica de garantizar disponibilidad continua han impulsado la migración hacia arquitecturas basadas en la nube.

La situación actual, previa al despliegue de la solución, es la de una infraestructura inexistente en la nube y una dependencia de métodos tradicionales que no se ajustan a los estándares modernos de alta disponibilidad y escalabilidad.

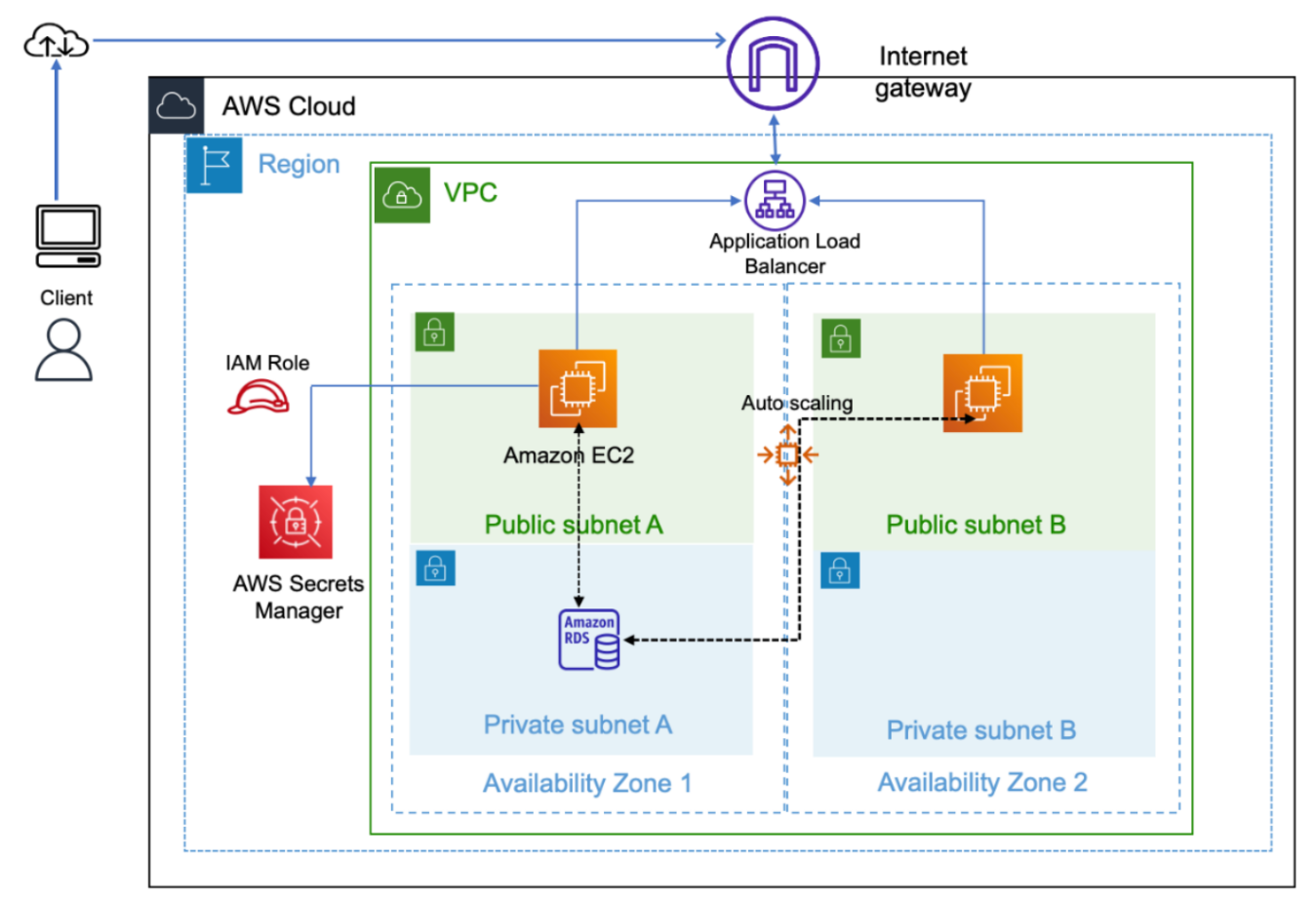
## Lógica del sistema

La lógica que guiará la arquitectura de la solución se basa en una separación de capas funcionales, siguiendo un modelo de arquitectura distribuida:

* **Capa de presentación** (Frontend). Instancias EC2 donde se aloja la aplicación web. Esta capa es accesible desde Internet mediante direcciones públicas o un balanceador de carga.
* **Capa de procesamiento** (Aplicación). Las instancias EC2 procesan las peticiones recibidas, acceden a la base de datos y devuelven los resultados al usuario.
* **Capa de datos** (Backend). Base de datos relacional alojada en Amazon RDS, aislada en subredes privadas para reforzar la seguridad.
* **Capa de balanceo de carga**. ELB que distribuye el tráfico entrante de forma automática entre las instancias disponibles.
* **Capa de escalado**. Auto Scaling Group que añade o elimina instancias en función de métricas de carga (por ejemplo, uso de CPU).

Toda la infraestructura estará encapsulada dentro de una VPC, diseñada específicamente para garantizar el aislamiento de recursos, la segmentación por AZ y la aplicación de políticas de seguridad específicas.

## Descripción física

La infraestructura propuesta será desplegada completamente en AWS, aprovechando los recursos virtualizados ofrecidos por su plataforma. A nivel físico, aunque no existe un hardware gestionado directamente por el usuario, la solución debería verse de la siguiente manera:

Mientras que, a nivel físico de AWS, llevará lo siguiente:

* **Instancias EC2**
* **Tipo**: t3.micro.
* **Ubicación**: subredes públicas distribuidas entre varias zonas de disponibilidad.
* **Sistema operativo**: Ubuntu Server 22.04 LTS.
* **Base de datos RDS**
* **Motor:** MySQL
* **Tipo de instancia**: db.t3.micro.
* **Ubicación**: subredes privadas.
* **Red VPC**
* **CIDR:** 10.0.0.0/16.
* Subredes públicas y privadas en dos zonas de disponibilidad distintas.
* Gateway de Internet para acceso público.
* NAT Gateway para acceso saliente de instancias privadas.
* **Elastic Load Balancer (ELB)**
* Balanceo entre instancias EC2 distribuidas geográficamente.
* **Auto Scaling Group**
* Mínimo 1 instancia, máximo escalado dinámico según la demanda.
* **Security Groups y reglas de red**
* Control de acceso SSH, HTTP/HTTPS, y conexión a la base de datos exclusivamente desde las instancias de aplicación.

## Diagnóstico del sistema actual

El diagnóstico previo a la implementación revela que el modelo tradicional utilizado presenta numerosas debilidades:

* **Falta de redundancia**: un único punto de fallo puede colapsar todo el servicio.
* **Dificultad de escalado**: la ampliación de recursos requiere intervención manual y no puede adaptarse automáticamente a cambios de carga.
* **Problemas de seguridad**: sin segmentación adecuada de redes ni control de accesos, la exposición a ataques es elevada.
* **Costes de mantenimiento elevados**: necesidad de hardware propio, licencias, energía, y personal técnico especializado.

La transición a un modelo cloud sobre AWS busca corregir estos problemas mediante:

* Distribución geográfica de instancias en distintas zonas de disponibilidad.
* Activación de escalado automático basado en métricas objetivas.
* Separación de funciones entre capas de aplicación y base de datos.
* Implementación de políticas de seguridad rigurosas.
* Pago por uso real de los recursos consumidos.

## Normativa y legislación

Durante el desarrollo del proyecto se han tenido en cuenta las siguientes normativas y buenas prácticas legales y de seguridad:

* **Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD) y Reglamento General de Protección de Datos (RGPD)**

Aunque el proyecto no maneja datos personales reales, se configuran las instancias siguiendo buenas prácticas de minimización de datos y seguridad.

* **AWS Shared Responsibility Model**

Se asume que AWS es responsable de la seguridad de la infraestructura física, mientras que el alumno es responsable de la seguridad de los recursos virtuales desplegados (configuración de instancias, seguridad de redes, control de acceso, etc.).

* **Licencias de software**

El sistema operativo Ubuntu Server es de licencia libre y gratuita.

Los servicios utilizados son parte del Free Tier de AWS o cubiertos por créditos educativos, cumpliendo las condiciones de uso establecidas.

* **Normativa interna del centro educativo**

Cumplimiento de los requisitos de documentación, presentación y ejecución del módulo de proyecto establecidos por el Departamento de Informática del C.F.G.S. de Administración de Sistemas Informáticos en Red.

## Requisitos del sistema

## Requisitos

Los requisitos del sistema se definen como todas aquellas condiciones técnicas, funcionales y operativas necesarias para que la infraestructura desplegada cumpla los objetivos planteados. Podemos distinguir tres tipos de requisitos indispensables:

* **Requisitos funcionales**
* El sistema debe permitir el despliegue de una aplicación web accesible desde Internet.
* La base de datos debe estar separada de la capa de aplicación y alojada en un servicio gestionado (Amazon RDS).
* El acceso a la aplicación debe estar distribuido mediante un balanceador de carga.
* El sistema debe ser capaz de escalar automáticamente el número de instancias EC2 en función de la carga.
* La aplicación debe ser capaz de permanecer operativa aunque una zona de disponibilidad (AZ) quede inactiva.
* **Requisitos no funcionales**
* El sistema debe desplegarse en la región de AWS más cercana para minimizar la latencia.
* El tiempo de inactividad durante las pruebas no debe superar los 5 minutos (simulación de alta disponibilidad).
* El acceso debe estar restringido y protegido mediante grupos de seguridad y uso de claves SSH.
* La solución debe ser documentada de forma clara y replicable.
* **Requisitos técnicos**
* Se debe utilizar al menos una instancia EC2 (Ubuntu Server).
* Debe configurarse una VPC con al menos dos subredes públicas y dos privadas.
* La base de datos debe alojarse en Amazon RDS MySQL.
* Debe habilitarse Auto Scaling y un Load Balancer (ELB).
* Todos los recursos deben estar contemplados en la capa gratuita de AWS o dentro del crédito educativo asignado.

## Restricciones del sistema

Al tratarse de un entorno educativo, el proyecto debe ajustarse a una serie de limitaciones derivadas tanto del modelo académico como de las características del entorno AWS gratuito o con créditos limitados:

* **Uso de cuenta de estudiante**
* Limitación en el número y tipo de instancias EC2 disponibles (por ejemplo, solo t2.micro o t3.micro). Por este motivo no se usa más espacio.
* Algunas regiones pueden no ofrecer todos los servicios gratuitos o no estar disponibles.
* Créditos con duración limitada o entornos cerrados (en el caso de AWS Academy).
* **Restricción de costes**
* El sistema debe diseñarse para funcionar sin generar costes adicionales fuera del crédito gratuito.
* Recursos como el ELB, RDS o NAT Gateway pueden consumir crédito rápidamente si no se controlan.
* **Tiempo de uso limitado**
* En entornos de laboratorio, el entorno tiene un tiempo de vida predefinido de 4 horas por sesión.
* **Recursos de cómputo reducidos**
* Las instancias gratuitas (t2.micro) tienen rendimiento limitado: 1 vCPU y 1 GB RAM.
* El rendimiento puede verse afectado si se exceden las cuotas o se comparten recursos con otros usuarios.
* **Seguridad educativa controlada**
* Algunas funcionalidades de IAM (usuarios y políticas) pueden estar bloqueadas en entornos de AWS Academy.
* No se permite el uso de datos reales ni configuraciones que impliquen exposición a Internet sin control.

## Catalogación y priorización de los requisitos

A continuación, se presenta una tabla con la clasificación de los requisitos según su tipo y nivel de prioridad:

| **Requisito** | **Tipo** | **Prioridad** |
| --- | --- | --- |
| Desplegar una aplicación web accesible desde Internet | Funcional | Alta |
| Separar la base de datos en RDS | Funcional | Alta |
| Configurar Auto Scaling con al menos 2 instancias | Técnica / Funcional | Media |
| Implementar ELB como balanceador de carga | Técnica / Funcional | Alta |
| Utilizar red VPC con subredes públicas y privadas | Técnica | Alta |
| Aplicar reglas de seguridad personalizadas (grupos de seguridad) | Técnica | Alta |
| Usar instancias EC2 t2.micro (capas gratuitas o con crédito) | Técnica | Alta |
| Hacer que la arquitectura tolere la caída de una zona de disponibilidad | Funcional | Alta |
| Documentar todas las fases del despliegue | No funcional | Alta |
| Optimizar el coste para no exceder el crédito disponible | No funcional | Alta |
| Mejorar estética y experiencia del frontend | No funcional | Baja |
| Simular monitorización y alertas (CloudWatch, opcional) | Técnica / Ampliación | Media/Baja |

## Alternativas y selección de la solución

## Alternativa 1: Arquitectura mínima funcional

Esta alternativa consiste en desplegar una solución básica pero funcional, centrada en minimizar el uso de recursos de AWS para no superar el crédito educativo o el Free Tier. Está especialmente pensada para entornos de prácticas o laboratorios con recursos limitados.

**Características:**

* Red VPC con una sola zona de disponibilidad (AZ).
* Subred pública donde se ubica una instancia EC2 que aloja tanto la aplicación web como la base de datos (MySQL o PostgreSQL).
* Reglas de seguridad para permitir tráfico HTTP y SSH.
* Instancia EC2 del tipo t2.micro o t3.micro, dentro del Free Tier.
* Arquitectura monolítica, sin balanceador de carga ni Auto Scaling.
* Documentación clara del proceso, justificación técnica y pruebas de acceso desde el exterior.

**Ventajas:**

* Muy bajo consumo de recursos, ideal para trabajar con cuenta de estudiante.
* Rápida de configurar y desplegar.
* No requiere servicios adicionales que puedan generar coste (como ELB o RDS).
* Perfecta para demostrar los conceptos básicos de redes, seguridad y despliegue en la nube.

**Inconvenientes:**

* No garantiza alta disponibilidad: si la instancia falla, el servicio se interrumpe.
* Sin escalabilidad automática ante carga.
* La base de datos comparte recursos con la aplicación, lo que puede afectar el rendimiento.

## Alternativa 2: Arquitectura escalable y distribuida (elegida)

Esta opción contempla una infraestructura más completa, siguiendo buenas prácticas de AWS orientadas a alta disponibilidad, desacoplamiento de servicios y escalado automático. Esta es la alternativa escogida por lo siguiente:

**Características:**

* VPC con varias subredes en diferentes zonas de disponibilidad.
* Instancias EC2 para la aplicación distribuidas en varias AZs.
* Base de datos gestionada en Amazon RDS (subred privada).
* Auto Scaling y Load Balancer para garantizar resiliencia y adaptabilidad.
* Arquitectura orientada a producción real.

**Ventajas:**

* Alta disponibilidad y escalabilidad.
* Separación clara de servicios (aplicación y base de datos).
* Mejores prácticas profesionales.

**Inconvenientes:**

* Consumo elevado de crédito, especialmente por servicios como ELB, NAT Gateway o RDS.
* Mayor complejidad técnica.
* Requiere más tiempo de configuración y pruebas.

## Conclusiones

Se ha optado por desarrollar la **Alternativa 2**, priorizando la calidad técnica, la seguridad y la escalabilidad del sistema, incluso si esto implica un mayor consumo de crédito en la cuenta educativa. Esta decisión permite implementar una arquitectura más cercana a un entorno de producción real, aplicando conceptos clave de VPC, subredes públicas y privadas, grupos de seguridad, instancias EC2, NAT Gateway y conexión segura a la base de datos.

Aunque implica un esfuerzo adicional de configuración y un mayor coste, esta opción refuerza el aprendizaje práctico y la adquisición de competencias profesionales en el uso de servicios avanzados de AWS.

## Planificación del proyecto

Este apartado describe los recursos necesarios, las tareas a realizar y la planificación temporal estimada para llevar a cabo el proyecto. La planificación se ha realizado considerando la sencillez de la arquitectura elegida, así como las limitaciones de tiempo y crédito disponibles.

## Recursos del proyecto

Recursos materiales (en la nube de AWS)

| **Recurso** | **Descripción** | **Tipo** |
| --- | --- | --- |
| **Instancia EC2** | Servidor virtual (t2.micro) con Ubuntu Server para alojar la aplicación web. | Free Tier |
| **VPC** | Red virtual privada con subred pública. | Gratuito |
| **Grupo de seguridad** | Reglas de firewall personalizadas (HTTP, SSH). | Gratuito |
| **Almacenamiento EBS** | Volumen asociado a la instancia EC2. | 30 GB Free Tier |
| **Dirección IP elástica (opcional)** | Para asignar una IP fija a la instancia. | Monetizado |
| **Servicio web (nginx, Apache, etc.)** | Servidor web y stack de aplicación instalado manualmente. | Software libre |
| **Base de datos local** | Motor de base de datos (MySQL o PostgreSQL) instalado en la misma instancia. | Software libre |

## Tareas del proyecto

| **Tarea** | **Descripción** |
| --- | --- |
| **Estudio de viabilidad y análisis inicial** | Identificación de necesidades, alternativas y elección de la arquitectura. |
| **Diseño de red en AWS (VPC + subred)** | Creación de la infraestructura básica de red. |
| **Creación y configuración de la instancia EC2** | Despliegue de servidor, instalación del sistema operativo. |
| **Instalación del servidor web y base de datos** | Configuración de Apache/Nginx + MySQL/PostgreSQL en la misma instancia. |
| **Implementación de medidas de seguridad** | Configuración de grupos de seguridad, acceso SSH y puertos. |
| **Pruebas de conectividad y acceso externo** | Verificación de que la aplicación es accesible vía navegador desde Internet. |
| **Capturas, validación y evidencias** | Documentación de las pruebas, capturas de consola, panel de AWS, etc. |
| **Redacción de la memoria técnica** | Explicación detallada de cada fase del proyecto. |
| **Estimación de costes** | Evaluación con AWS Pricing Calculator y control del uso del crédito. |
| **Presentación final del proyecto** | Preparación y exposición de los resultados ante el profesor o tribunal. |

## Evaluación de riesgos

## Lista de riesgos

* R1: Agotamiento del crédito gratuito de AWS.
* R2: Eliminación accidental de recursos (instancias, red, etc.).
* R3: Configuración incorrecta de seguridad (puertos abiertos, acceso global).
* R4: Caída de la instancia EC2 (por error, actualización o fallo).
* R5: Pérdida de datos de la base de datos al no estar desacoplada ni respaldada.
* R6: Fallos en el acceso a la cuenta o problemas con credenciales.

## Catalogación de riesgos

| **Código** | **Descripción** | **Probabilidad** | **Impacto** | **Nivel** |
| --- | --- | --- | --- | --- |
| **R1** | Créditos educativos agotados | Alta | Media | Alto |
| **R2** | Borrado accidental de recursos | Media | Alta | Alto |
| **R3** | Errores de configuración de seguridad | Media | Media | Medio |
| **R4** | Fallo en instancia EC2 | Media | Alta | Alto |
| **R5** | Pérdida de datos sin backup | Baja | Alta | Medio |
| **R6** | Problemas de acceso a la cuenta | Baja | Alta | Medio |

## Plan de contingencia

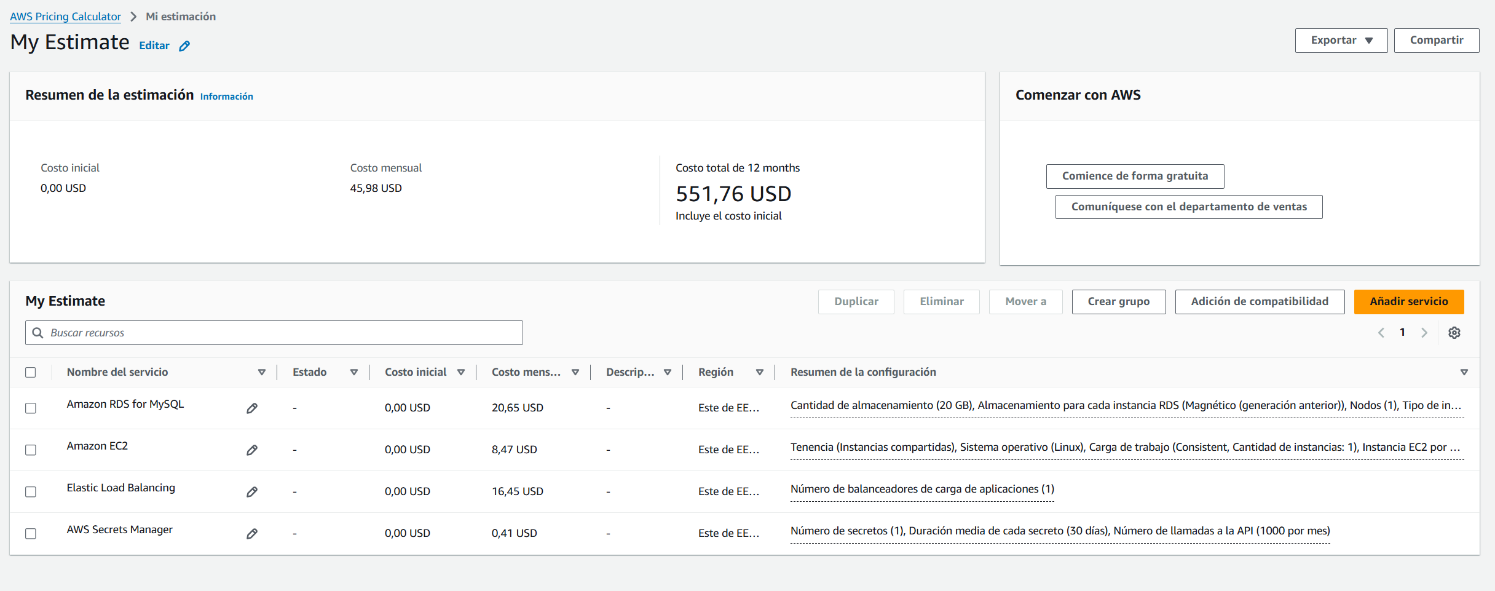
* **R1**: Crear una nueva cuenta de estudiante y reconfigurar el proyecto mediante CloudFormation.
* **R2**: Realizar capturas y guardar una plantilla del despliegue para poder recrearlo.
* **R3**: Usar solo los puertos necesarios (22, 80) y restringir accesos por IP
* **R4**: Monitorizar el estado de la instancia y tener un script de instalación guardado (User Data).
* **R5**: Exportar la base de datos periódicamente (si se usa contenido real).
* **R6**: Guardar credenciales en un gestor seguro y usar autenticación de múltiples factores si está disponible.

## Presupuesto

## Estimación de costes materiales y personales

La estimación económica del proyecto se ha realizado teniendo en cuenta dos componentes: el coste de los recursos en la nube y el valor aproximado del trabajo técnico realizado por el estudiante (la luz está muy cara).

Se ha utilizado la herramienta oficial ***AWS Pricing Calculator*** para estimar el consumo de los servicios seleccionados, utilizando instancias y configuraciones incluidas dentro del Free Tier o cubiertas por el crédito educativo. A continuación, se muestra un resumen:



## Resumen y análisis coste-beneficio

El proyecto ha sido ejecutado sin generar ningún coste económico real gracias al uso responsable de la capa gratuita de *AWS* y los créditos educativos disponibles para el estudiante. Esto demuestra que es viable desarrollar e implementar una solución en la nube con características reales de producción sin necesidad de inversión económica inicial.

Desde el punto de vista formativo y profesional, el beneficio obtenido es elevado, ya que, como estudiante, he podido trabajar con tecnologías reales y actuales, enfrentándome a un proceso completo de despliegue, análisis de costes, documentación técnica y planificación.

Además, esta experiencia permite escalar el proyecto en el futuro si se dispone de más recursos o si se desea migrar a una arquitectura más compleja (con *RDS, Auto Scaling y ELB*).

El punto a tener en cuenta es que, si es un proyecto que seguirá en el tiempo, podemos ver que mantener una aplicación que generará beneficios sin necesitar aumentar el hardware que se necesitará.

## Conclusiones

## Beneficios

El desarrollo de este proyecto ha aportado múltiples beneficios tanto desde el punto de vista formativo como técnico y personal:

* **Aprendizaje práctico de la nube.** Se ha trabajado directamente con AWS, una de las plataformas cloud líderes, adquiriendo experiencia en despliegue de infraestructura real.
* **Diseño completo de una solución funcional**. Aunque se ha optado por una arquitectura mínima, el sistema resultante cumple los objetivos básicos de disponibilidad y funcionalidad.
* **Coste cero**. Gracias al uso de la cuenta de estudiante, se ha conseguido desplegar toda la solución sin incurrir en gastos reales.
* **Desarrollo de competencias clave**. Se han puesto en práctica conocimientos de redes, servidores, seguridad, automatización y documentación técnica.
* **Documentación profesional**. La estructura de la memoria y el uso de herramientas como AWS Pricing Calculator o CloudFormation aportan valor añadido y profesionalismo.
* **Base para evolución futura**. Aunque se ha implementado una versión básica, la solución es escalable y puede convertirse en una arquitectura más avanzada si se requiere.

## Inconvenientes

A pesar del éxito general del proyecto, también se han identificado algunas limitaciones e inconvenientes:

* **Falta de alta disponibilidad real**. Al tratarse de una única instancia EC2, no se garantiza la continuidad del servicio ante fallos de hardware o red.
* **Dependencia de la cuenta gratuita**. La viabilidad del proyecto depende del uso responsable del crédito educativo; si se agota, el sistema deja de estar operativo.
* **Configuración manual.** Al no utilizar herramientas como CloudFormation para desplegar todo desde el inicio, la creación es más lenta y propensa a errores si se repite. Por ello se optará por esta automatización más adelante.

# Análisis

## Introducción

El objetivo del análisis es definir las funcionalidades del sistema, las necesidades de los usuarios y los requisitos técnicos y no técnicos del entorno de despliegue. Este proyecto consiste en el despliegue de una aplicación web en la nube utilizando AWS, con una arquitectura distribuida que separa la lógica de aplicación y la base de datos entre subredes pública y privada respectivamente. Se aplican principios de seguridad, escalabilidad y buenas prácticas en infraestructura.

## Requisitos funcionales de usuarios

* El usuario debe poder acceder a la aplicación web desde Internet mediante un navegador.
* La aplicación debe permitir operaciones CRUD sobre los datos almacenados.
* El acceso a la base de datos debe estar restringido solo a la aplicación, no directamente accesible desde el exterior.
* Debe garantizarse que, en caso de reinicio de la máquina, los datos no se pierdan (almacenamiento persistente).
* El administrador debe tener acceso mediante SSH a la instancia EC2 para tareas de mantenimiento o monitorización.

## Requisitos no funcionales

* **Disponibilidad.** La aplicación debe estar disponible 24/7 durante el periodo de pruebas.
* **Seguridad**. La base de datos se encuentra en una subred privada; solo la aplicación puede comunicarse con ella. Se usan grupos de seguridad para limitar el tráfico entrante y saliente.
* **Escalabilidad**. La arquitectura puede evolucionar para incluir balanceadores de carga y múltiples instancias.
* **Rendimiento**. La latencia entre la aplicación y la base de datos debe mantenerse baja mediante el uso de zonas de disponibilidad adecuadas.
* **Facilidad de uso**. El acceso a la aplicación debe ser intuitivo para el usuario final.

## Diagrama del sistema

Diagrama

El contenido generado por IA puede ser incorrecto.

## Conclusión del análisis

El análisis ha permitido definir con claridad las necesidades funcionales del sistema, así como las características técnicas que garantizarán su correcto funcionamiento en la nube. La arquitectura distribuida elegida proporciona una solución segura, escalable y profesional, adecuada para un entorno educativo avanzado. Este enfoque sienta las bases para el diseño e implementación de un sistema que refleje prácticas reales de despliegue en AWS.

# Diseño

## Introducción

El diseño del sistema define cómo se desplegará y estructurará la aplicación web en AWS. Se ha optado por una arquitectura distribuida que separa la lógica de aplicación y la base de datos en diferentes subredes, aplicando buenas prácticas de seguridad, aislamiento de recursos y escalabilidad. A continuación, se detallan las decisiones de diseño tomadas en cada componente del sistema.

## Selección de la arquitectura de red

Se ha seleccionado una arquitectura de red basada en una VPC personalizada que incluye:

* Una subred pública, donde se aloja una instancia EC2 que actúa como servidor web (frontend y backend).
* Una subred privada, donde se aloja una base de datos gestionada en Amazon RDS.
* Una NAT Gateway, para permitir que la base de datos acceda a actualizaciones sin estar expuesta directamente a Internet.
* Grupos de seguridad, configurados para limitar el acceso: solo el servidor web puede acceder a la base de datos, y solo ciertos puertos están abiertos al público (por ejemplo, 80 y 443).

# Implementación

Para implementar este proyecto, necesitaremos varias fases, cada una con un objetivo concreto y una relación directa con la siguiente. Estas fases permiten estructurar el trabajo de forma progresiva, asegurando que el diseño, despliegue y validación de la infraestructura se realicen de manera ordenada y eficaz.

## Fase 1: Creación de una aplicación web funcional básica

En esta fase, comenzamos a construir la solución. Configuramos la red virtual y creamos el servidor web donde alojaremos la aplicación.

La configuración de la red incluye la creación de una nube virtual privada (VPC), una puerta de enlace de internet, una subred y las rutas necesarias para permitir la conectividad. La solución que implementamos es la siguiente:

En la parte superior de la Consola de administración de AWS, utilizamos la barra de búsqueda para buscar y seleccionar VPC. Elegimos Create VPC (Crear VPC) y configuramos los siguientes ajustes:

* **Nombre de la etiqueta**: introducimos CapstoneVPC.
* **IPv4 CIDR**: establecemos 10.0.0.0/16 como bloque de direcciones.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Luego, en el panel de navegación, seleccionamos Internet gateways y procedimos a:

* Elegimos Create internet gateway.
* Asignamos el nombre de etiqueta CapstoneIGW.
* Creamos la puerta de enlace de internet.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Después, la adjuntamos a nuestra VPC:

* Seleccionamos Actions > Attach to VPC.
* Elegimos CapstoneVPC como VPC disponible.
* Hicimos clic en Attach internet gateway.

Seguidamente, en el panel de navegación, accedimos a Subnets y configuramos lo siguiente:

* Elegimos Create subnet.
* Seleccionamos CapstoneVPC como VPC.
* Nombramos la subred como Subred pública 1.
* Escogimos la primera Availability Zone disponible.
* Introdujimos el CIDR 10.0.1.0/24.
* Creamos la subred.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

A continuación, definimos la tabla de rutas:

* En Route tables, buscamos la asociada a CapstoneVPC y accedimos a su Route table ID.
* En la pestaña Routes, hicimos clic en Edit routes.
* Añadimos una ruta:
* Destination: 0.0.0.0/0
* Target: Internet Gateway → CapstoneIGW

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Con esto, dejamos lista la infraestructura básica de red virtual. El siguiente paso fue desplegar una máquina virtual (instancia EC2) dentro de esta red para alojar la aplicación web.

El siguiente paso será crear una instancia EC2 (máquina virtual) donde instalamos tanto la aplicación web como el cliente que se conectará a la base de datos ubicada en Amazon RDS. La solución que implementamos fue la siguiente:

En la parte superior de la Consola de administración de AWS, utilizamos la barra de búsqueda para localizar y seleccionar EC2.

* A continuación, seleccionamos Launch instance > Launch instance, y configuramos los siguientes parámetros:
* En la sección Name and tags, para el campo Name, escribimos CapstonePOC.
* En Application and OS Images, dentro de Quick Start, seleccionamos Ubuntu como sistema operativo base.
* En Key pair, elegimos un par de claves existente llamado vockey, que previamente habíamos generado para acceder por SSH a la instancia.
* En la sección Network settings, procedimos con la configuración de red:
  + Para el campo VPC, seleccionamos CapstoneVPC.
  + En Auto-assign Public IP, marcamos la opción Enable, para que la instancia tenga una IP pública y sea accesible desde Internet.
* En Firewall (grupos de seguridad), optamos por Create security group y configuramos añadimos CapstoneAPPSG.
* Luego, agregamos reglas de seguridad:
* Mantenemos la regla SSH predeterminada para acceso administrativo.
* Añadimos una regla HTTP, permitiendo acceso web desde cualquier parte (Anywhere).
* Añadimos una regla MySQL/Aurora, restringiendo el acceso solo al rango IP interno de la VPC: 10.0.0.0/16. Esta regla permite que nuestra instancia web se comunique con la base de datos RDS ubicada en la subred privada.
* Añadimos en user-data la siguiente configuración para instalar la app y la base de datos:

Texto

El contenido generado por IA puede ser incorrecto.

Así debería quedar la instancia:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Y sólo quedaría probar la aplicación que funciona insertando algunos datos:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

## Fase 2: desacoplamiento de los componentes de la aplicación

Después de construir la POC en una única máquina virtual y de que funcione como se espera, pasaremos al siguiente nivel de diseño para desacoplar las capas de la aplicación y de la base de datos. Separar las capas proporciona flexibilidad a la arquitectura y la capacidad de escalar cada capa de forma independiente. También se puede implementar la seguridad en ambas capas para mejorar la posición de seguridad.

El siguiente diagrama muestra el aspecto que podría tener la arquitectura una vez desacoplados los componentes:

Diagrama

El contenido generado por IA puede ser incorrecto.

Ahora crearemos la subred privada donde alojaremos la base de datos Amazon RDS:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ahora crearemos una tabla de enrutamiento privada, sin la internet Gateway, cuyo destino es sólo local:

Texto

El contenido generado por IA puede ser incorrecto.

Haremos otra subred privada con los mismos pasos:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

El siguiente paso será crear la base de datos con las siguientes especificaciones:

Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Ya tenemos la base de datos de Amazon RDS lista, ahora nos faltaría poder enlazarla con la app del proyecto, para ello configuraremos un entorno de desarrollo. Debe configurarse así:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Una vez en el entorno, configuraremos el Secrets Manager para almacenar las credenciales de la base de datos. La aplicación web utilizará estas credenciales para comunicarse con la instancia de base de datos RDS. Para hacer esto, iniciaremos el entorno de desarrollo y por comando enlazaremos la app de la siguiente forma:

Texto

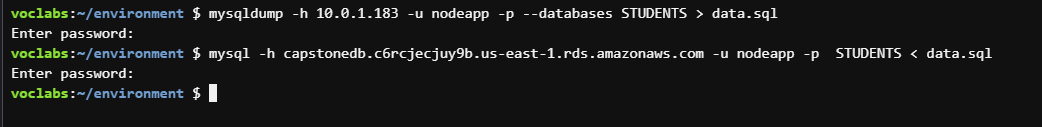
El contenido generado por IA puede ser incorrecto.

Ahora toca aprovisionar una nueva instancia para el servidor web, dicha instancia servirá como template más adelante.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

El siguiente paso de la fase es migrar la base de datos a Amazon RDS. Con el secreto listo, volcaremos la base de datos en un .sql y mediante el rdsendpoint, le daremos los datos de la base.



Con estos pasos veremos los datos que teníamos antes en nuestra base de datos:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

## Fase 3: implementar la alta disponibilidad y escalabilidad

El primer paso de esta fase será crear un Application Load Balancer, no sin antes crear otra subred pública dentro de la segunda AZ. La vpc debe quedar así:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ahora crearemos el Balanceador de carga para distribuir el tráfico de aplicaciones entre dos zonas de disponibilidad. Una vez creado debe verse así:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Para continuar, crearemos una AMI para el grupo de Auto Scaling que crearemos. Esta AMI será la que se lanzará cada vez que el Auto Scaling necesite una nueva instancia por peticiones o fallos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

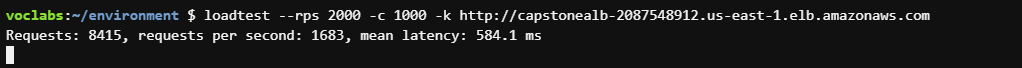
El contenido generado por IA puede ser incorrecto.

Ahora queda crear el grupo de AutoScaling y usar la AMI creada como base para escalar la aplicación siempre y cuando se necesite. Al crearse, debe verse así:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Ya tenemos lista la estructura final, sólo quedaría comprobar mediante test de solicitudes si la aplicación carga las peticiones correctamente, para ello, usaremos el comando que envíe peticiones para cargarla.



Ahora podremos comprobar las peticiones y que se creó una instancia más a parte:Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# Implementación extra

## Introducción

Como ampliación del proyecto, se propone el uso de herramientas de Infraestructura como Código (IaC) para automatizar la provisión y gestión de los recursos necesarios en la nube. Esta mejora busca aumentar la escalabilidad, repetibilidad y mantenibilidad del entorno desplegado.

Para ello, se presentan dos enfoques alternativos:

* ***AWS CloudFormation***, el servicio nativo de Amazon Web Services, que permite describir y aprovisionar toda la infraestructura mediante plantillas en formato YAML o JSON. Esta herramienta se integra perfectamente con el ecosistema de AWS y permite gestionar el ciclo de vida completo de los recursos de forma declarativa.
* ***Terraform***, una herramienta de código abierto desarrollada por HashiCorp, que permite definir infraestructura en múltiples proveedores (incluyendo AWS, Azure, Google Cloud, etc.) mediante archivos de configuración en HCL (HashiCorp Configuration Language). Terraform destaca por su enfoque multiplataforma, su planificación de cambios (plan) antes de aplicar, y su comunidad activa.

Ambas soluciones ofrecen ventajas frente al enfoque manual o semi-automatizado utilizado en el desarrollo principal del proyecto, y permiten implementar entornos completos con solo ejecutar unas pocas instrucciones, garantizando consistencia y control de versiones de la infraestructura.

## CloudFormation

AWS CloudFormation es un servicio que permite provisionar y administrar recursos de AWS mediante archivos de plantilla en formato YAML o JSON. En lugar de crear recursos manualmente desde la consola, utilicé CloudFormation en otro laboratorio para:

* Automatizar el despliegue de la infraestructura.
* Garantizar que el entorno sea replicable y consistente.
* Versionar los cambios en la infraestructura como si fuera código (Infrastructure as Code - IaC).

La plantilla se diseñó para definir todos los recursos necesarios para el funcionamiento de la aplicación. Los principales bloques que contiene son:

* **AWSTemplateFormatVersion**: Especifica la versión de la plantilla.
* **Description**: Breve descripción del stack.
* **Parámetros**: Para definir valores personalizables (como tipo de instancia EC2).
* **Resources**: Define todos los recursos que se desplegarán (EC2, RDS, Security Groups, etc.).
* **Outputs**: Muestra información útil al final del despliegue (como la IP pública del servidor).

Recursos desplegados con CloudFormation

* En mi proyecto, CloudFormation se encargó de crear y configurar automáticamente:
* Una instancia EC2 con un script de user-data para instalar y configurar la aplicación.
* Un grupo de seguridad (Security Group) con puertos abiertos para HTTP, HTTPS y SSH.
* Una base de datos RDS (MySQL), si se requería almacenamiento persistente.
* Un bucket S3, en caso de requerir almacenamiento de archivos estáticos.
* Roles de IAM, si la instancia necesita permisos para acceder a otros servicios.

Ventajas obtenidas con CloudFormation

🚀 Automatización del despliegue completo con un solo archivo.

🔄 Repetibilidad: puedo destruir y volver a crear la infraestructura fácilmente.

🧪 Entornos consistentes entre desarrollo, pruebas y producción.

🗂️ Control de versiones de la infraestructura junto con el código del proyecto.

Para implementar esta solución, lo primero será tener un archivo que contenga todo lo implementado en el laboratorio. Se usó la siguiente plantilla <CloudFormation.yaml>.

Una vez tenemos lista la plantilla, nos iremos a cloudformation y crearemos una pila:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Continuaremos y dejaremos que se ejecute la plantilla:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Cuando acabe, tendríamos desplegada la misma estructura que en el otro laboratorio.

## Terraform

Terraform es una herramienta de Infrastructure as Code (IaC) desarrollada por HashiCorp que permite provisionar y gestionar infraestructura en múltiples proveedores cloud, entre ellos AWS.

Utilicé Terraform en mi proyecto para:

* Automatizar la creación y configuración de los recursos de AWS.
* Versionar la infraestructura como si fuera código fuente.
* Asegurar reproducibilidad y facilidad para desplegar múltiples entornos (desarrollo, producción, etc.).

La infraestructura fue descrita en archivos .tf, que organizan los recursos y configuraciones de la siguiente manera:

* **main.tf**: Define los recursos principales (instancia EC2, grupos de seguridad, etc.).
* **variables.tf**: Declara las variables personalizables (tipo de instancia, regiones, etc.).
* **outputs.tf**: Define salidas útiles como IP pública, DNS, etc.
* **provider.tf**: Configura el proveedor AWS y región.

Ventajas obtenidas con Terraform:

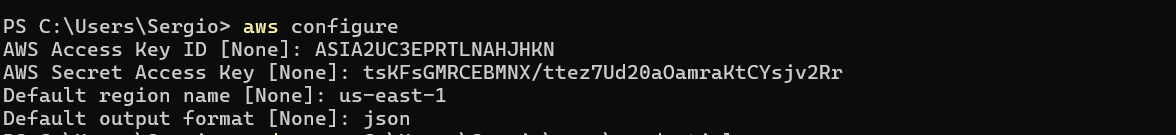
✅ Automatización total del despliegue: todos los recursos se crean desde código.

🔁 Entornos reproducibles: desarrollo, pruebas y producción idénticos.

🧩 Modularidad: la infraestructura puede dividirse en módulos reutilizables.

🌍 Multi-cloud: aunque este proyecto es en AWS, Terraform permite escalar a otros proveedores.

Para poder hacer esto, primero debemos descargar AWS cli para Windows y meter la carpeta como variable de entorno. Entonces, entramos a Powershell y configuramos las credenciales del laboratorio:



Una vez tenemos el archivo con parámetros listo <terraform.tf>, iniciaremos el proyecto para que descargue lo necesario y compruebe que el archivo no tiene fallos:

Texto

El contenido generado por IA puede ser incorrecto.

Una vez compruebe sin fallos, ejecutamos:

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Y, una vez acabe, tendremos desplegado el proyecto en el laboratorio: Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

# Conclusiones

La finalización de este proyecto ha representado una experiencia integral que permitió aplicar de forma práctica los conocimientos adquiridos sobre infraestructura en la nube y el diseño de sistemas resilientes, escalables y automatizados utilizando los servicios de Amazon Web Services (AWS). A lo largo de sus distintas fases, se construyó progresivamente una arquitectura moderna para alojar una aplicación web, desde su versión más básica hasta una solución altamente disponible y automatizada con herramientas profesionales de infraestructura como código.

En las primeras etapas, se desarrolló y desplegó una aplicación web funcional sobre una única instancia de Amazon EC2, lo cual permitió familiarizarse con el entorno y comprender los requisitos de conectividad, seguridad (grupos de seguridad y VPC), y acceso remoto. Posteriormente, se abordaron desafíos más avanzados como la desacoplación de servicios, migrando el backend a una base de datos relacional gestionada mediante Amazon RDS, en una subred aislada, respetando las buenas prácticas de segmentación y protección de recursos.

Uno de los momentos clave del proyecto fue la incorporación de un Elastic Load Balancer (ELB) y la configuración de Auto Scaling Groups (ASG), lo que permitió lograr una infraestructura escalable horizontalmente, capaz de adaptarse a la demanda de forma automática. Este paso fue fundamental para garantizar la alta disponibilidad y la tolerancia a fallos, objetivos centrales del proyecto. La capacidad de distribuir tráfico entre múltiples instancias y lanzar nuevas instancias de forma dinámica refuerza los principios de elasticidad y resiliencia que definen los entornos cloud nativos.

Además, se integraron herramientas de Infrastructure as Code (IaC) como AWS CloudFormation y Terraform, lo que permitió automatizar completamente el despliegue de la arquitectura creada manualmente. Esto no solo facilitó la reproducibilidad del entorno, sino que aportó valor añadido en términos de mantenimiento, versionado y eficiencia. Ambas herramientas mostraron fortalezas diferentes: CloudFormation como solución nativa e integrada con AWS, y Terraform como herramienta más flexible y modular, ideal para implementaciones complejas o multi-cloud.

El proyecto ha exigido una combinación de habilidades técnicas (configuración de redes, administración de instancias, seguridad en la nube, gestión de bases de datos, automatización con código) y organizativas (planificación, estimación de costes, elaboración de diagramas de arquitectura, documentación). Esta experiencia también ha fomentado el pensamiento crítico, la resolución de problemas, y la capacidad de adaptación ante errores y cambios de enfoque.

Finalmente, el enfoque práctico, dividido en fases progresivas, ha sido esencial para consolidar los conocimientos y ofrecer una visión realista del trabajo que se realiza en entornos empresariales modernos. La posibilidad de trabajar con recursos reales de AWS, junto con la oportunidad de retomar el laboratorio en distintas sesiones, permitió a los estudiantes profundizar sin la presión de un entorno efímero. Esta metodología ha demostrado ser efectiva para formar perfiles preparados para el mundo laboral, donde la automatización, la escalabilidad y la disponibilidad ya no son extras, sino requisitos fundamentales.

# Bibliografía

Amazon Web Services, Inc. (s.f.). *What is Amazon EC2?* Recuperado de <https://docs.aws.amazon.com/ec2/>

Amazon Web Services, Inc. (s.f.). *Amazon RDS Documentation*. Recuperado de <https://docs.aws.amazon.com/rds/>

Amazon Web Services, Inc. (s.f.). *Elastic Load Balancing Documentation*. Recuperado de <https://docs.aws.amazon.com/elasticloadbalancing/>

Amazon Web Services, Inc. (s.f.). *Auto Scaling Documentation*. Recuperado de <https://docs.aws.amazon.com/autoscaling/>

Amazon Web Services, Inc. (s.f.). *AWS CloudFormation User Guide*. Recuperado de <https://docs.aws.amazon.com/cloudformation/>

Amazon Web Services, Inc. (s.f.). *AWS Well-Architected Framework*. Recuperado de <https://aws.amazon.com/architecture/well-architected/>

HashiCorp. (s.f.). *Terraform by HashiCorp – Infrastructure as Code*. Recuperado de <https://www.terraform.io/>

HashiCorp. (s.f.). *Terraform AWS Provider Documentation*. Recuperado de https://registry.terraform.io/providers/hashicorp/aws/latest/docs

Microsoft. (s.f.). *Infrastructure as Code (IaC) Overview*. Recuperado de <https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code>

RedHat. (s.f.). *What is high availability?* Recuperado de https://www.redhat.com/en/topics/high-availability