ARQUITECTURA DE SOFTWARE

arquitectura de referenciaSistema de GESTION DE PROYECTOS

vERSION 1.0

2022

# Índice

[Índice 2](#_Toc101886265)

[Índice de ilustraciones 3](#_Toc101886266)

[Índice de tablas 3](#_Toc101886267)

[1 Descripción 4](#_Toc101886268)

[2 Propósito 5](#_Toc101886269)

[3 Registro de cambios 6](#_Toc101886270)

[4 Introducción 7](#_Toc101886271)

[4 Decisiones de Diseño 8](#_Toc101886272)

[4.1 Base de Datos 8](#_Toc101886273)

[4.2 Lenguaje de Programación 8](#_Toc101886274)

[4.3 Patrón de Diseño REST 8](#_Toc101886275)

[4.4 Patrón MVC 8](#_Toc101886276)

[4.5 Patrón Repositorio (Repository Pattern) 9](#_Toc101886277)

[4.6 Patrón DTO (Data Transfer Object) 9](#_Toc101886278)

[4.7 Patrón Singleton 9](#_Toc101886279)

[5 Diagrama de contexto del sistema 10](#_Toc101886280)

[5.1 Contexto del sistema 11](#_Toc101886281)

[5.1.1 Descripción de los contextos del sistema 11](#_Toc101886282)

[5.2 Vista de Atención al cliente 12](#_Toc101886283)

[5.2.1 Descripción de componentes 12](#_Toc101886284)

[6 Diagrama de contenedor del sistema 13](#_Toc101886285)

[6.1 Contenedores del sistema 14](#_Toc101886286)

[6.2 Descripción de los contenedores del sistema 15](#_Toc101886287)

[7 Diagrama de componentes del sistema 16](#_Toc101886288)

[7.1 Componentes del sistema 17](#_Toc101886289)

[7.2 Descripción de los componentes del sistema 18](#_Toc101886290)

[7.3 Arquitectura de referencia 19](#_Toc101886291)

[8 Diagrama entidad-relación 21](#_Toc101886292)

[9 Bibliografía 23](#_Toc101886293)

# Índice de ilustraciones

[Ilustración 1. Explicación Modelo C4 (Brown, 2020) 4](#_Toc101886294)

[Ilustración 2. Diagrama C4 - Contexto 11](#_Toc101886295)

[Ilustración 3. Modelo Vista Controlador 12](#_Toc101886296)

[Ilustración 4. Modelo C4 - Diagrama de Contenedores 14](#_Toc101886297)

[Ilustración 5. Diagrama C4 - Componentes del Sistema 17](#_Toc101886298)

[Ilustración 6. Estilo arquitectónico N-Capas 19](#_Toc101886299)

[Ilustración 7. Arquitectura de referencia 20](#_Toc101886300)

# Índice de tablas

[Tabla 1. Contexto del Sistema 11](#_Toc101886301)

[Tabla 2. Contenedores del Sistema 15](#_Toc101886302)

[Tabla 3. Componentes del Sistema 18](#_Toc101886303)

# Descripción

Los modelos y anotaciones definidos por un estándar de descripción de arquitecturas, como el modelo C4, ayudan a capturar y comunicar los resultados del proceso de diseño. Como complemento a este documento, existe el documento de decisiones de arquitectura que soporta las definiciones establecidas para la arquitectura del Sistema de Gestión de Proyectos

Este documento describe la arquitectura de sistemas de software, basado en la comunicación de manera eficiente y efectiva la arquitectura de software, a diferentes niveles de detalle, contando diferentes historias a diferentes tipos de audiencia.

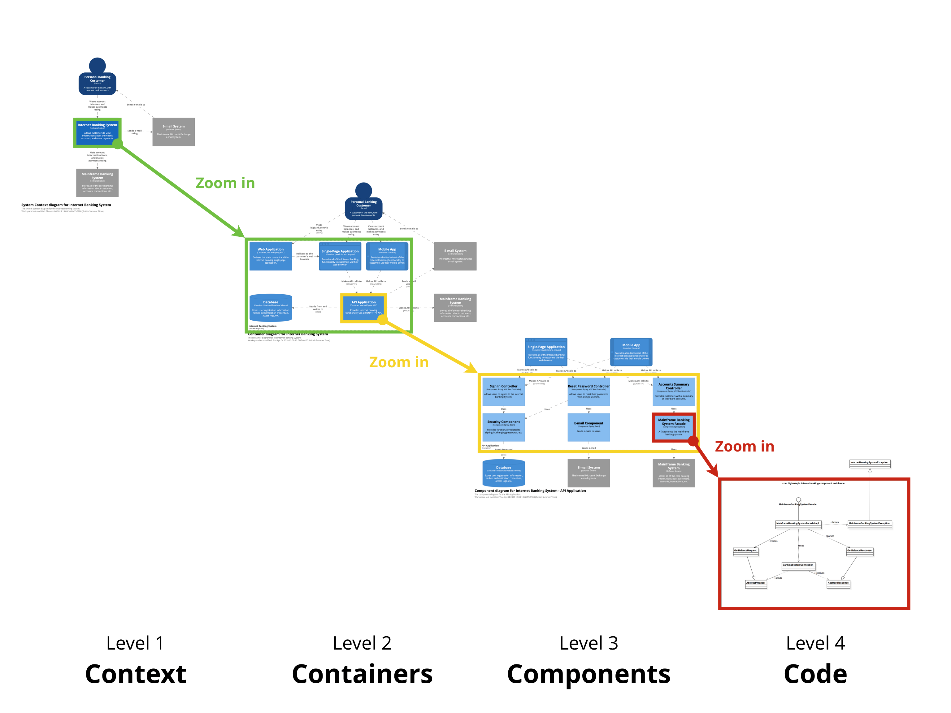


Ilustración 1. Explicación Modelo C4 (Brown, 2020)

# Propósito

Brindar una vista global de la arquitectura presentando aspectos relevantes de los componentes de software y hardware que conforman el Sistema de Gestión de Proyectos, teniendo en cuenta aspectos referentes a requerimientos funcionales, requerimientos no funcionales, restricciones técnicas y factores del negocio asociado al sistema.

Exponer un diseño funcional con alto nivel de abstracción, asociando los componentes, relaciones, responsabilidades y alcance, para asegurar el cumplimento de los requerimientos funcionales y la completitud de las funcionalidades definidas para el Sistema de Gestión de Proyectos. También analizan los atributos de calidad y la implicación de estos en las decisiones de la arquitectura para solucionar los requerimientos no funcionales del proyecto, esto se realiza a través de diferentes vistas que representan la arquitectura desde ópticas particulares responder a los intereses de cada una de las personas involucradas con la construcción del mismo.

# Registro de cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Motivo | Realizado Por | Rol | Fecha |
| 1.0 | Elaboración Inicial |  | Arquitecto |  |

# Introducción

Partiendo de las restricciones, requisitos y necesidades identificadas tempranamente, se busca contribuir a la documentación y a la toma de decisiones arquitectónicas de la solución. Este documento busca reflejar de una manera general, una visual a los distintos stakeholders.

# Decisiones de Diseño

A continuación, se presenta una serie de decisiones tomadas respecto a la arquitectura, los cuales se ven reflejados en el producto final; estas decisiones partieron de la experiencia y los conocimientos adquiridos; responden en gran parte de los casos a dos atributos de calidad (disponibilidad e interoperabilidad)

## Base de Datos

Se define para el sistema transaccional o Core, una base de datos **MSSQL**, para darle un soporte optimo y performance al proyecto.

## Lenguaje de Programación

Se utiliza **ASP.NET Framework** como base del proyecto, porque es uno de los lenguajes más utilizados, del cual se pueden encontrar recursos que fácilmente pueden dar **mantenimiento** a los sistemas escritos en este lenguaje, además de **estandarizar** el lenguaje para cada uno de los sistemas propuestos.

## Patrón de Diseño REST

El **patrón de diseño REST** Se implementa para dar solución a una de las necesidades más claras, que la comunicación sea abierta y se pueda acceder desde cualquier sitio, de manera que, aplicando este estilo permita un primer nivel de organización para acceder a cada uno de los recursos de forma independientemente, favoreciendo la **reutilización** y aumentando la **flexibilidad** para poder clasificar el tipo de operación y el método de envío a utilizar (GET, POST, PUT, DELETE)

El objetivo de implementar este patrón de arquitectura es **establecer un marco** (protocolos, diseño y metodología) sobre el que se deben integrar los nuevos canales de las organizaciones externas para **evitar al máximo las dependencias estructurales** entre ellos. para distribuir todo ese tráfico en dos o más nodos.

## Patrón MVC

Se implementa MVC porque facilita el escalamiento vertical y horizontal ya que la aplicación se hospeda en varias instancias, se usa un equilibrador de carga para asignar solicitudes a instancias individuales de la aplicación. Cuando aumenten las necesidades de la aplicación, se pueden necesitar soluciones de implementación más sólidas y complejas. Esta unidad se puede escalar vertical u horizontalmente para aprovechar la escalabilidad a petición basada en la nube. El escalado vertical significa agregar más CPU, memoria, espacio en disco u otros recursos al servidor en el que se hospeda la aplicación. El escalado horizontal significa agregar instancias adicionales de estos servidores, con independencia de que sean servidores físicos, máquinas virtuales o contenedores.

## Patrón Repositorio (Repository Pattern)

El patrón repositorio consiste en separar la lógica que recupera los datos y los asigna a un modelo de entidad de la lógica de negocios que actúa sobre el modelo, esto permite que la lógica de negocios sea independiente del tipo de dato que comprende la capa de origen de datos, en pocas palabras un repositorio, media entre el dominio y las capas de mapeo de datos, actuando como una colección de objetos de dominio en memoria (M. Fowler).

## Patrón DTO (Data Transfer Object)

El patrón DTO tiene como finalidad la creación de objetos planos con una serie de atributos que puedan ser enviados o recuperados del servidor en una sola invocación, de tal forma que un DTO puede contener información de múltiples fuentes o tablas y concentrarlas en una única clase simple.

* Solo lectura: Dado que el objetivo de un DTO es utilizarlo como un objeto de transferencia entre el cliente y el servidor, es importante evitar tener operaciones de negocio o métodos que realicen cálculos sobre los datos, es por ello que solo deberemos de tener los métodos GET y SET de los respectivos atributos del DTO.
* Serializable: Es claro que, si los objetos tendrán que viajar por la red, deberán de poder ser serializables, pero no hablamos solamente de la clase en sí, sino que también todos los atributos que contenga el DTO deberán ser fácilmente serializables.

## Patrón Singleton

Singleton es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

# Diagrama de contexto del sistema

Esta vista representa los componentes de software, relaciones entre ellos y responsabilidades de cada componente con el objetivo de presentar la arquitectura del sistema con un alto nivel de abstracción.

El enfoque debe estar en las personas (actores, roles, personas, etc.) y los sistemas de software en lugar de las tecnologías, los protocolos y otros detalles de bajo nivel. Es el tipo de diagrama que puede mostrar a personas no técnicas.

* **Alcance:** un único sistema de software.
* **Elementos primarios:** El sistema de software en alcance.
* **Elementos de soporte:** personas (por ejemplo, usuarios, actores, roles o personas) y sistemas de software (dependencias externas) que están directamente conectados al sistema de software en su alcance. Por lo general, estos otros sistemas de software se encuentran fuera del alcance o los límites de su propio sistema de software, y usted no tiene responsabilidad ni propiedad de ellos.
* **Audiencia prevista:** Todos, tanto técnicos como no técnicos, dentro y fuera del equipo de desarrollo de software. (Brown, 2020)

## Contexto del sistema

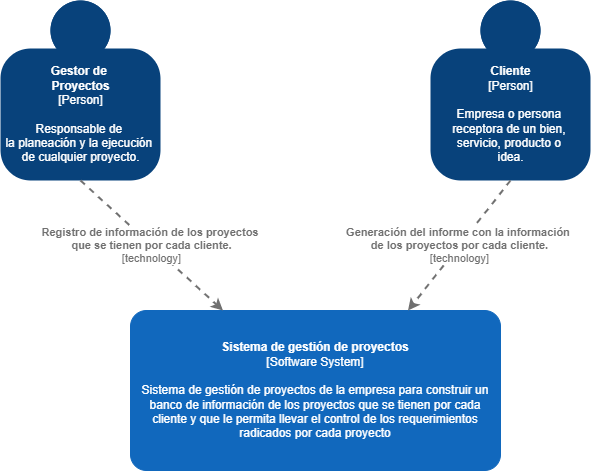


Ilustración 2. Diagrama C4 - Contexto

### Descripción de los contextos del sistema

|  |  |  |
| --- | --- | --- |
| Sistema | Descripción | Naturaleza |
| Sistema de gestión de proyectos | Sistema de gestión de proyectos de la empresa para construir un banco de información de los proyectos que se tienen por cada cliente y que le permita llevar el control de los requerimientos radicados por cada proyecto. | Sistema propio |

Tabla 1. Contexto del Sistema

## Vista de Atención al cliente

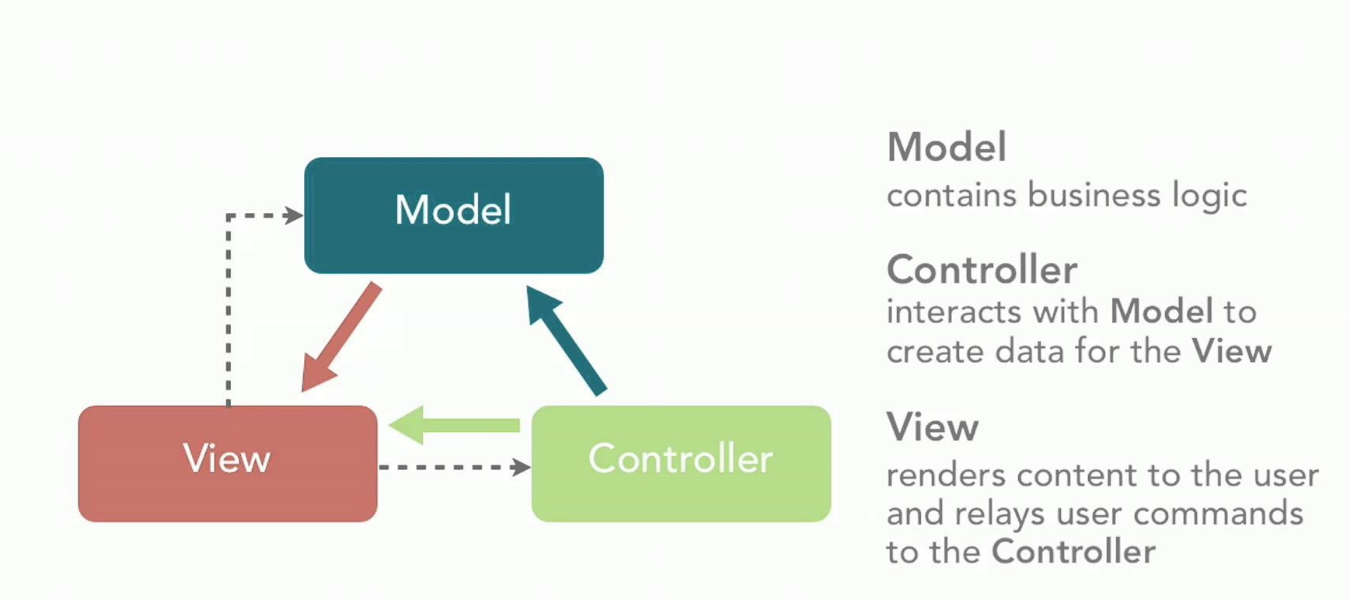


Ilustración 3. Modelo Vista Controlador

### Descripción de componentes

A continuación, se describe las responsabilidades de cada uno de los componentes identificados en la solución hasta el momento:

**El modelo**

Es la capa encargada de los datos, es decir, la que se encarga de hacer peticiones a las bases de datos para enviar o recibir información. Estas bases de datos pueden estar alojadas de forma local en nuestra app o de forma remota en un servidor externo.

**La vista**

Se trata del código que nos permitirá presentar los datos que el modelo nos proporciona, como ejemplo podríamos decir que en una aplicación es el código HTML que nos permite mostrar la salida de los datos procesados.

**Controlador**

Es la capa que sirve de enlace entre la vista y el modelo. Envía comandos al modelo para actualizar su estado, y a la vista correspondiente para cambiar su presentación.

# Diagrama de contenedor del sistema

El diagrama de Contenedores muestra la forma de alto nivel de la arquitectura de software y cómo se distribuyen las responsabilidades en ella. También muestra las principales opciones tecnológicas y cómo los contenedores se comunican entre sí.

* **Alcance:** un único sistema de software.
* **Elementos primarios:** Contenedores dentro del sistema de software en alcance.
* **Elementos de soporte:** personas y sistemas de software directamente conectados a los contenedores.
* **Audiencia prevista:** personas técnicas dentro y fuera del equipo de desarrollo de software; incluyendo arquitectos de software, desarrolladores y personal de operaciones / soporte.
* **Notas:** Este diagrama no dice nada sobre escenarios de implementación, agrupación, replicación, conmutación por error, etc. (Brown, 2020)

## Contenedores del sistema

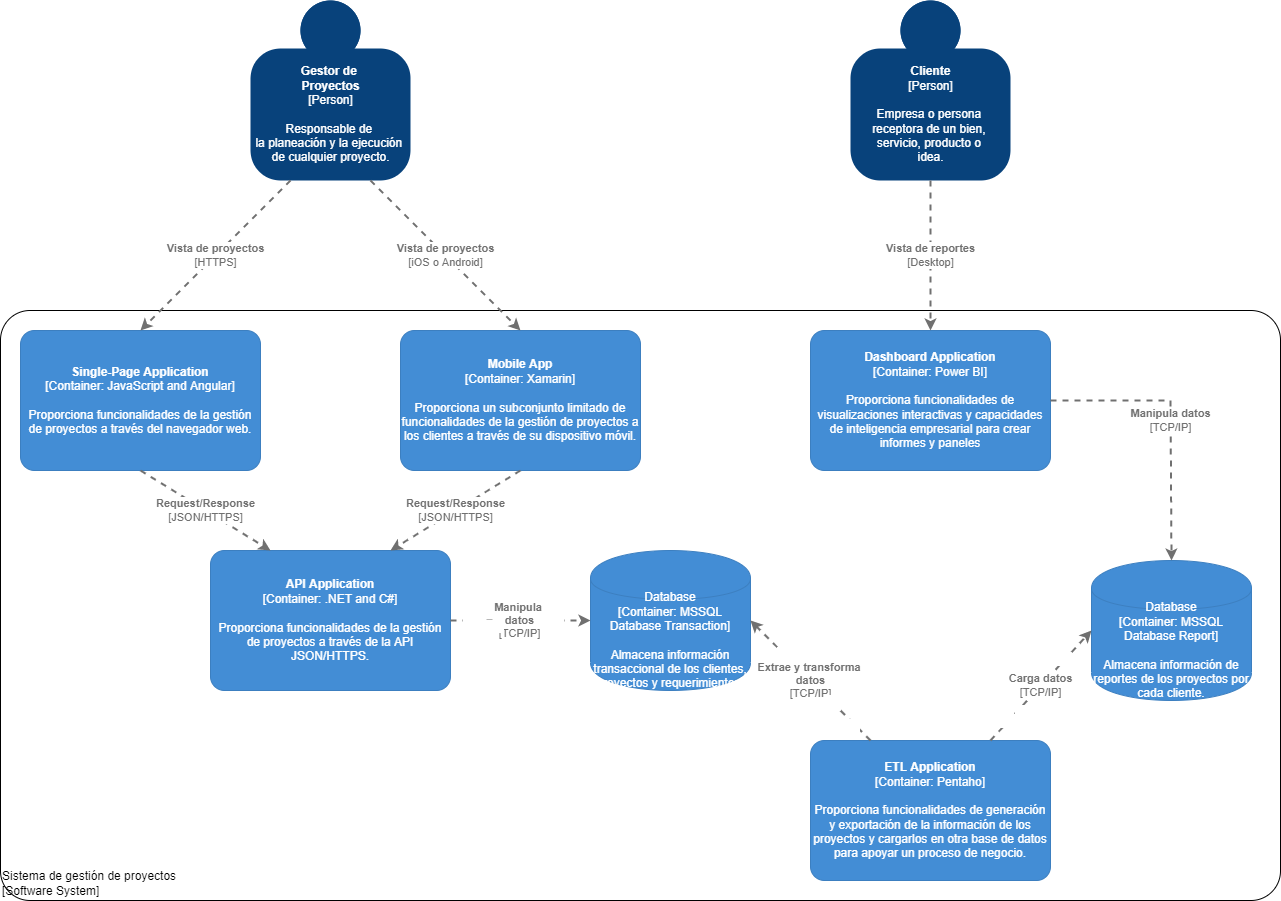


Ilustración 4. Modelo C4 - Diagrama de Contenedores

## Descripción de los contenedores del sistema

|  |  |  |  |
| --- | --- | --- | --- |
| Contenedor | Descripción | Tecnología | Sistema |
| MSSQL Database Transaction | Almacena información transaccional de los clientes, proyectos y requerimientos. | MSSQL | Sistema de gestión de proyectos |
| MSSQL Database Report | Almacena información de reportes de los proyectos por cada cliente. | MSSQL | Sistema de gestión de proyectos |
| ETL Application | Proporciona funcionalidades de generación y exportación de la información de los proyectos y cargarlos en otra base de datos para apoyar un proceso de negocio. | Pentaho | Sistema de gestión de proyectos |
| Dashboard Application | Proporciona funcionalidades de visualizaciones interactivas y capacidades de inteligencia empresarial para crear informes y paneles | Power BI | Sistema de gestión de proyectos |
| API Application | Proporciona funcionalidades de la gestión de proyectos a través de la API JSON/HTTPS. | .NET and C# | Sistema de gestión de proyectos |
| Mobile App | Proporciona un subconjunto limitado de funcionalidades de la gestión de proyectos a los clientes a través de su dispositivo móvil. | Xamarin | Sistema de gestión de proyectos |
| Single-Page Application | Proporciona funcionalidades de la gestión de proyectos a través del navegador web. | JavaScript and Angular | Sistema de gestión de proyectos |

Tabla 2. Contenedores del Sistema

# Diagrama de componentes del sistema

El diagrama de componentes muestra cómo un contenedor está compuesto por una serie de "componentes", cuáles son cada uno de esos componentes, sus responsabilidades y los detalles de tecnología / implementación.

* **Alcance:** un solo contenedor.
* **Elementos primarios:** Componentes dentro del contenedor en alcance.
* **Elementos de soporte:** Contenedores (dentro del alcance del sistema de software) más personas y sistemas de software directamente conectados a los componentes.
* **Destinatarios:** arquitectos y desarrolladores de software. (Brown, 2020)

## Componentes del sistema

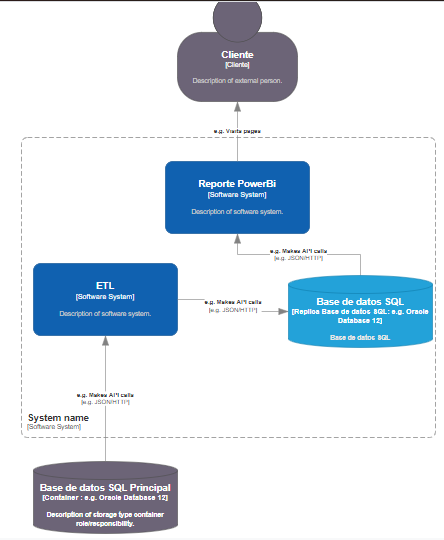


Ilustración 5. Diagrama C4 - Componentes del Sistema

## Descripción de los componentes del sistema

|  |  |
| --- | --- |
| Componente | Descripción |
| Base de datos Transaccional | Base de datos fuente, que proporciona los datos para la visualización de los reportes. |
| Base de datos de Reportes | Base de datos de reportes, sobre la cual se transfiere los datos de la Base de Datos Transaccional (SQL Server) |
| ETL | Proceso de Extracción, Transformación y Carga de los datos desde la BD transaccional hasta la BD de Reportes. (Visual Studio 2019) |
| Dashboard | Tablero de informes que conecta con la BD de reportes para no cargar la BD transaccional (Power BI) |
| Cliente | Usuario que le dará uso al dashboard para la respectiva toma de decisiones. |

Tabla 3. Componentes del Sistema

## Arquitectura de referencia

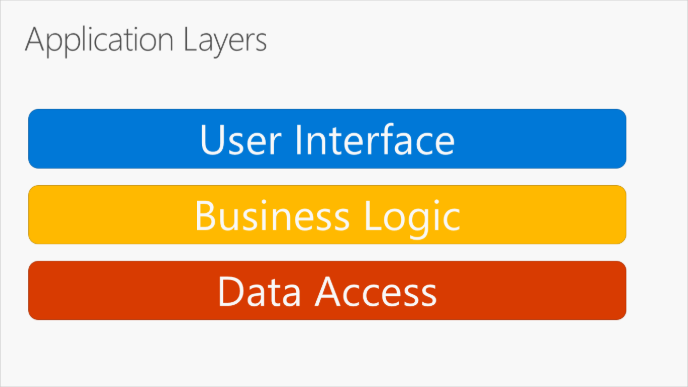


Ilustración 6. Estilo arquitectónico N-Capas

Estilo arquitectónico N-Capas, es un estilo arquitectónico donde se busca la separación en niveles de los diferentes componentes de una aplicación empresarial. Algunos niveles:

* Presentación
* Lógica de negocio
* Acceso a datos
* Integración

Estas capas se suelen abreviar como UI (interfaz de usuario), BLL (capa de lógica de negocios) y DAL (capa de acceso a datos). Con esta arquitectura, los usuarios realizan solicitudes a través de la capa de interfaz de usuario, que interactúa con la capa BLL. BLL, a su vez, puede llamar a DAL para las solicitudes de acceso de datos. La capa de interfaz de usuario no debe realizar solicitudes directamente a DAL, ni debe interactuar con la persistencia de forma directa a través de otros medios. Del mismo modo, BLL solo debe interactuar con la persistencia a través de DAL. De este modo, cada capa tiene su propia responsabilidad conocida.

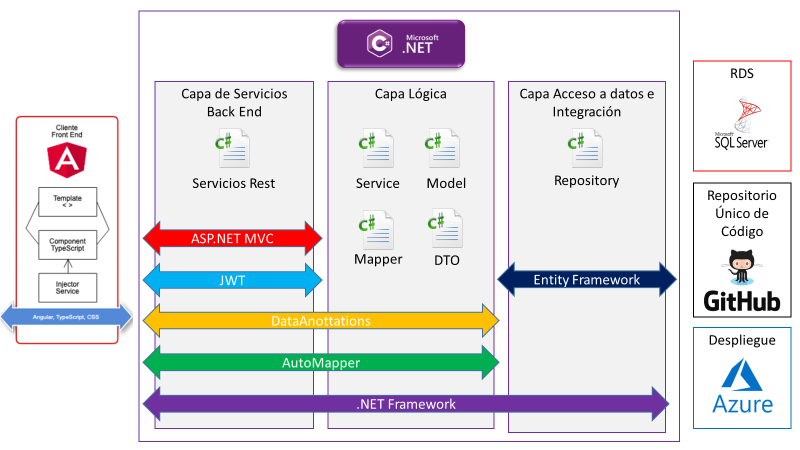


Ilustración 7. Arquitectura de referencia

# Diagrama entidad-relación

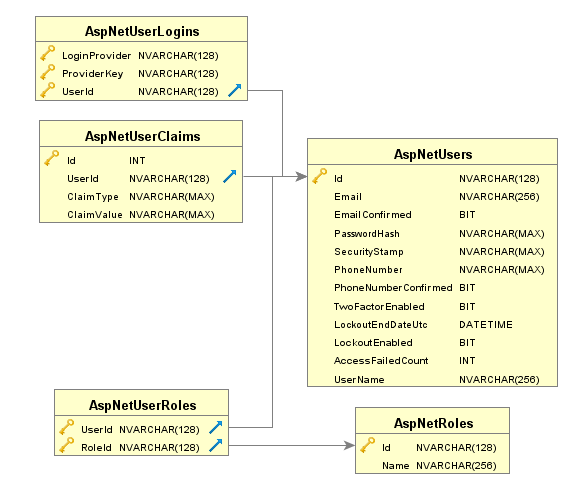


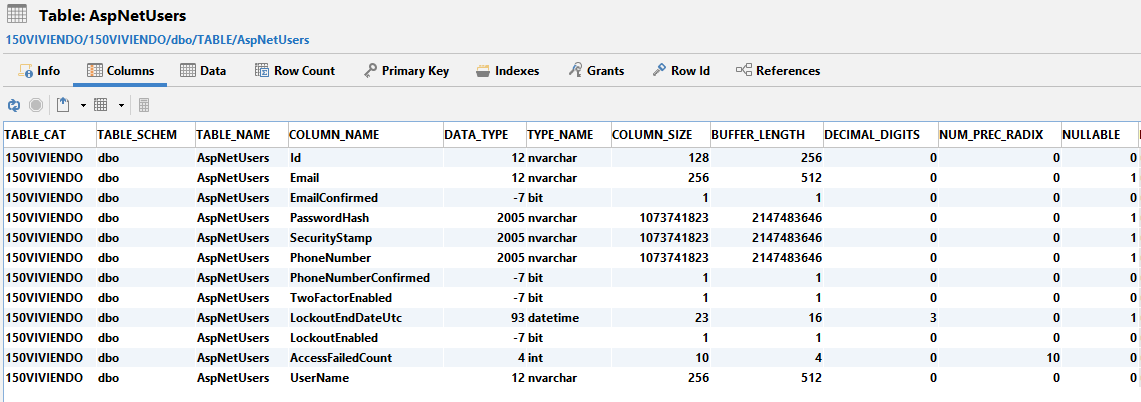
Ilustración7. Modelo entidad relación del proyecto

**Tabla:** [dbo]. [AspNetUsers]

**Descripción:** Tabla encargada de almacenar la información del usuario de acuerdo a la autenticación basada en Identity. Con Identity se crean las siguientes tablas:

* AspNetRoles
* AspNetUserClaims
* AspNetUserLogins
* AspNetUserRoles
* AspNetUsers

En estas tablas se guarda toda la información relacionada con los usuarios.



# Bibliografía

Brown, S. (07 de 02 de 2020). *C4 MODEL*. Obtenido de C4 MODEL: https://c4model.com/