Proyecto

Web App ToDoList LaLa

Alumna Laura Tatiana Gaona Castaño

Docente Mauricio Morales Meneses

Asignatura Arquitectura de software

Grupo A1

Universidad Manuela Beltrán

Año 2025

**ÍNDICE**

1. Definición del proyecto
   1. Planteamiento del problema
   2. Solución Ofrecida
   3. Justificación
   4. Usuario Final
   5. Utilidad
2. Comparativa de Ideas de proyecto
   1. Tabla1 (Ventajas/Desventajas/Justificación)
3. Producto Backlog
   1. Tabla2 (MoSCoW y Fibonacci)
4. Sprint Backlog
   1. Tabla3 (Criterios de Aceptación y Tareas)
5. Plan de prueba
   1. Casos de HU
6. Estrategias & Rol
7. Cronograma Sprint
   1. Tabla4 (Descripción de tareas a realizar)
8. Diagramas UML y Modelo 4+1
   1. Diagrama de clases
   2. Diagrama de objetos
   3. Diagrama de despliegue
   4. Diagrama de paquetes
   5. Diagrama de secuencia
   6. Diagrama de Casos de uso
   7. Esquema 4+1
9. Bibliografías

Texto

El contenido generado por IA puede ser incorrecto.**DEFINICIÓN DEL PROYECTO**

Nombre del proyecto:

ToDoList LaLa

Fecha de creación:

8 de Agosto del 2025

* 1. Planteamiento del Problema

Se escogió la edad actual, ya que muchas personas, incluyendo equipos de trabajo presentan dificultades para organizar sus actividades diarias y priorizar tareas de manera eficiente. El uso de agendas físicas o recordatorios dispersos reduce la productividad y puede ocasionar olvidos.

* 1. Solución Ofrecida

Desarrollar una aplicación Web que permita a los usuarios crear, editar, eliminar y marcar como completadas sus tareas. La aplicación permitirá organizar las tareas por categorías y fechas limite.

* 1. Justificación

ToDoList aporta valor al facilitar la organización personal y profesional de los usuarios, mejorando la gestión del tiempo y reduciendo olvidos, Es un sistema ligero, de bajo costo y fácil adopción.

* 1. Usuario Final

El sistema esta dirigido a estudiantes, profesionales independientes y equipos pequeños que necesitan gestionar sus tareas y proyectos de forma sencilla y accesible.

* 1. Utilidad

La utilidad del sistema radica en su capacidad de aumentar la productividad. El modelo podría incluir una versión gratuita y una versión premium con funciones adicionales como notificaciones avanzadas y colaboración en tiempo real.

**COMPARATIVA DE IDEAS DE PROYECTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Idea | Ventaja | Desventajas | Complejidad | Justificación |
| Gestión de lista de tareas | Fácil desarrollo, alta utilidad | Funciones básicas | Media | Elegí esta por su aplicabilidad en su día a día |
| Agenda Medica Online | Aporta a sector salud una manera eficaz de ordenar sus agendas diarias | Mayor regulación y complejidad | Media-Alta |  |
| Sistema de inventario comercial pequeño | Útil para negocios que requieran de tener inventarios al día | Mas funciones y reportes | Media |  |

Tabla 1.

**PRODUCTO BACKLOG**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID HU | Descripción | Prioridad (MoSCoW) | Valor Usuario (1-5) | Estimación (pts. Fibonacci) |
| HU1 | Crear tarea | Must Have | 5 | 3 |
| HU2 | Editar tarea | Must Have | 4 | 3 |
| HU3 | Eliminar tarea | Should Have | 4 | 2 |
| HU4 | Marcar tarea completada | Must Have | 5 | 2 |
| HU5 | Categorizar tareas | Could Have | 3 | 1 |

Tabla 2.

**STRING BACKLOG**

|  |  |  |  |
| --- | --- | --- | --- |
| ID HU | Tareas | Criterios de Aceptación | Esfuerzo (pts. Fibonacci) |
| HU1 | Diseñar formulario de nueva tarea  Programar API creación | El usuario puede añadir título, descripción y fecha; La tarea aparece en la lista | 3 |
| HU2 | Diseñar Interfaz edición  Programar API actualización | El usuario puede modificar datos; Los cambios se guardan y reflejan inmediatamente | 3 |
| HU4 | Añadir checkbox para marcar completada  Programar Lógica | El usuario puede marcar/ desmarcar su tarea creada; El estado permanecerá en la base de datos. | 2 |

Tabla 3.

**PLAN DE PRUEBA**

Sprint Test Planning

Casos de prueba

**HU1 – Crear tarea**

1. Verificar que el formulario permite ingresar título.
2. Verificar que se puede agregar descripción.
3. Verificar que se puede asignar fecha límite.
4. Guardar la tarea y verificar que aparece en la lista.
5. Probar validación cuando el título está vacío.

**HU2 – Editar tarea**

1. Abrir una tarea existente y verificar que se cargan sus datos.
2. Cambiar el título y guardar.
3. Cambiar la fecha y guardar.
4. Verificar que los cambios se reflejen sin recargar la página.
5. Verificar que no se pueda guardar si faltan campos obligatorios.

**HU4 – Marcar tarea completada**

1. Marcar una tarea como completada y verificar el cambio visual.
2. Desmarcar una tarea completada y verificar el cambio.
3. Refrescar la página y comprobar que el estado se mantiene.
4. Verificar que no se pueden marcar tareas inexistentes.
5. Probar el marcado en dispositivos móviles.

**ESTRATEGIAS Y ROL**

**CRONOGRAMA DE TRABAJO**

**Fechas Claves**

Inicio del Sprint: Viernes 08 de agosto

Revisión de avances: Todos los viernes

Finalización Sprint: Viernes 19 de septiembre

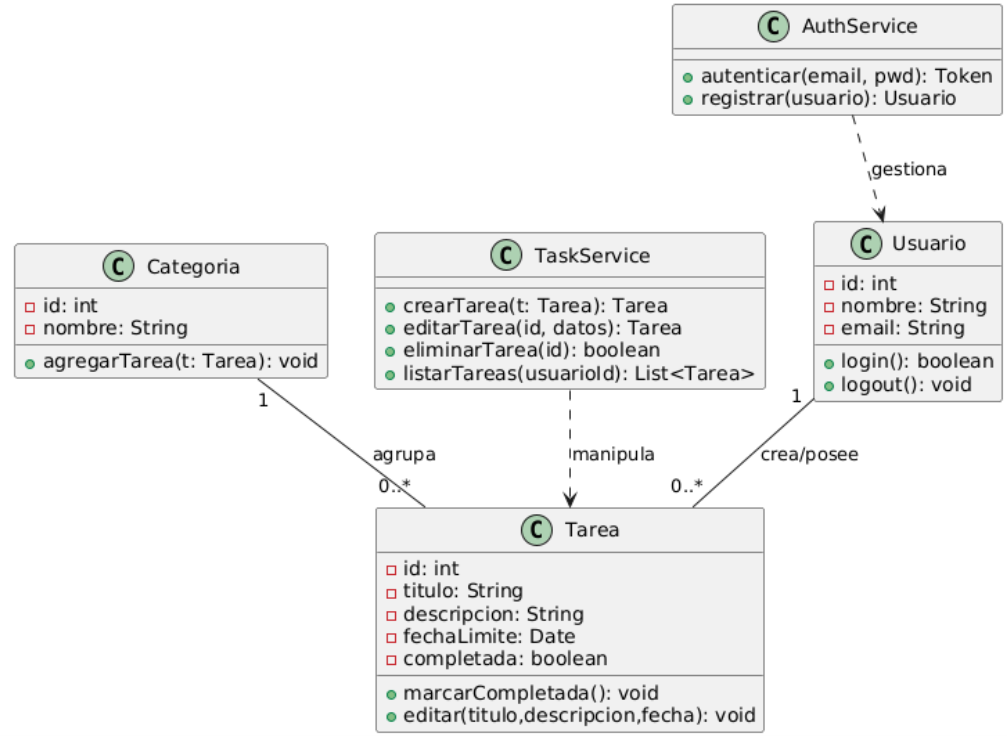
Sprint Review: Lunes 22 de septiembre

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tarea | Responsable | Fecha de Inicio | Fecha Fin | Estado |
| Diseño formulario nueva tarea | Laura Gaona Castaño | 08/08 | 15/08 | Pendiente |
| Programación API creación | Laura Gaona Castaño | 15/08 | 22/08 | Pendiente |
| Diseño de Interfaz Edición | Laura Gaona Castaño | 22/08 | 29/08 | Pendiente |
| Programación API actualización | Laura Gaona Castaño | 29/08 | 05/09 | Pendiente |
| Lógica marcado completada | Laura Gaona Castaño | 05/09 | 12/09 | Pendiente |
| Pruebas y correcciones | Laura Gaona Castaño | 12/09 | 19/09 | Pendiente |

**DIAGRAMAS UML Y MODELO 4+1**

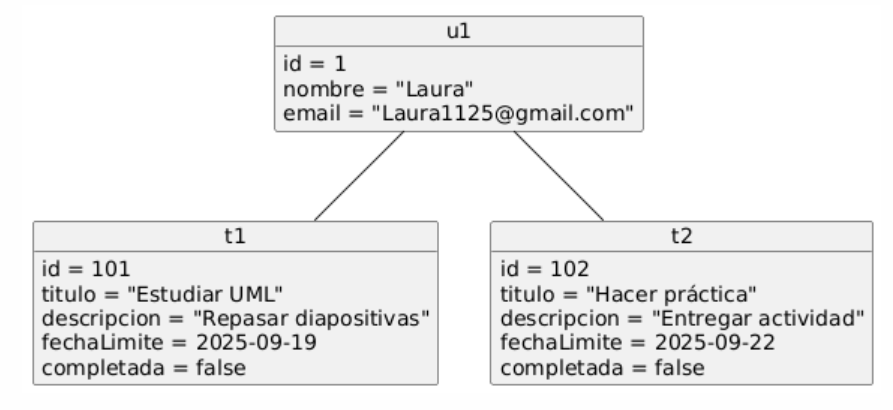
8.1. Diagrama de clases

Se muestra la estructura estática: Usuario, Tarea, Categoría y servicios (Auth y Task).



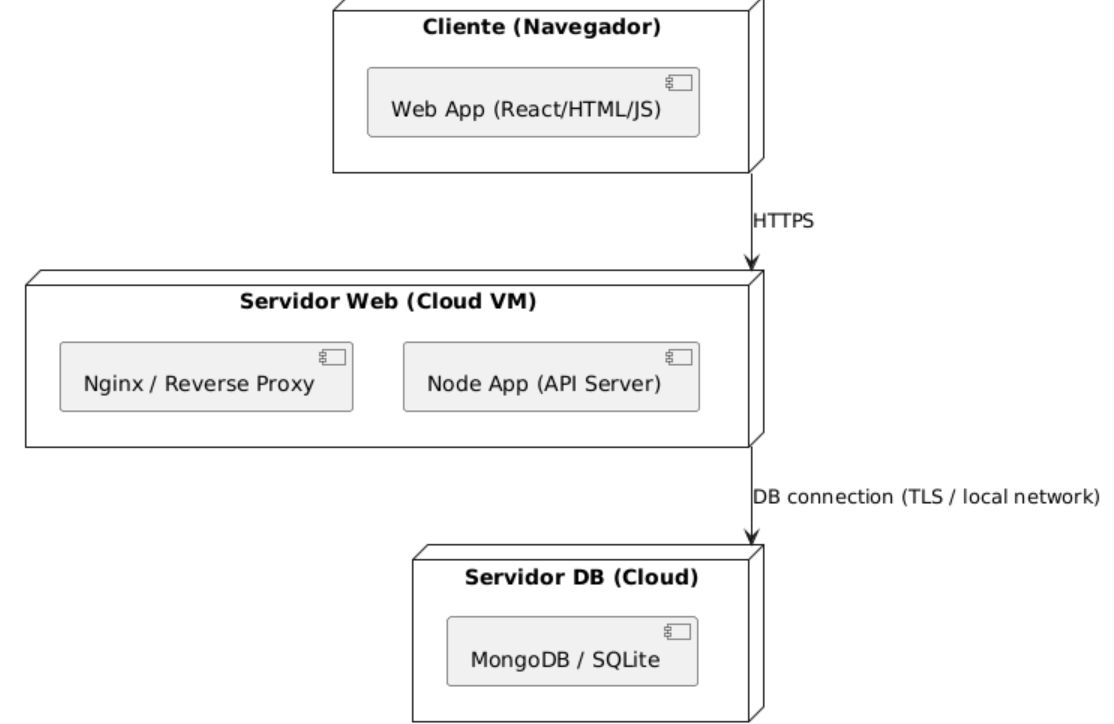
* 1. Diagrama de objetos

Instancias concretas (Usuario Laura y sus tareas).



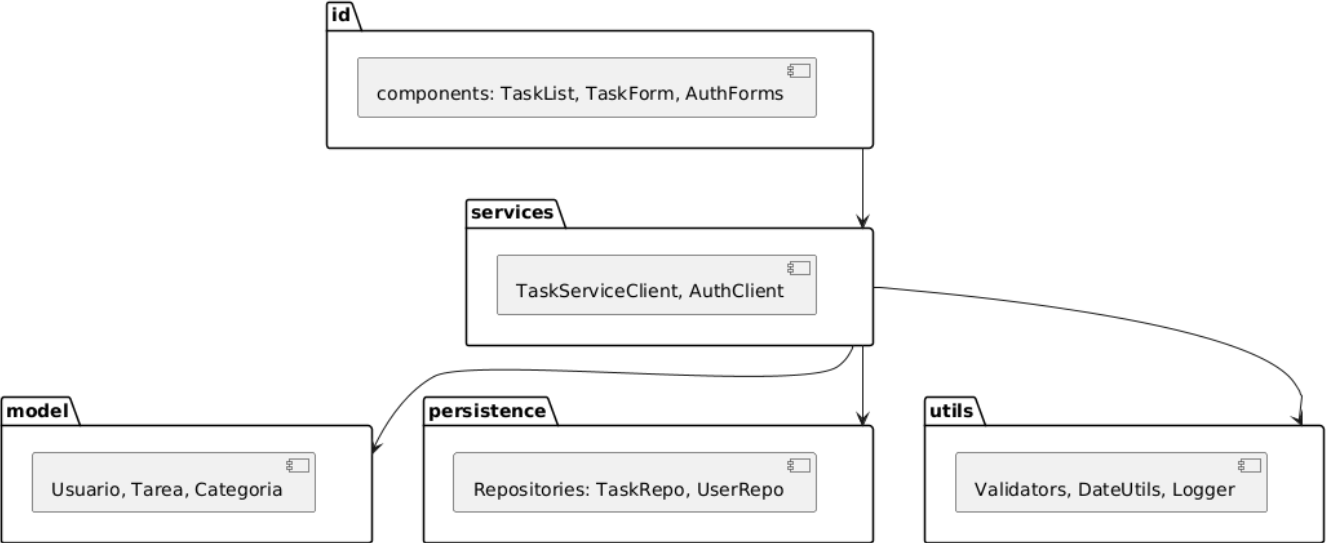
* 1. Diagrama de despliegue

Cliente → servidor web (Nginx/Node) → base de datos.



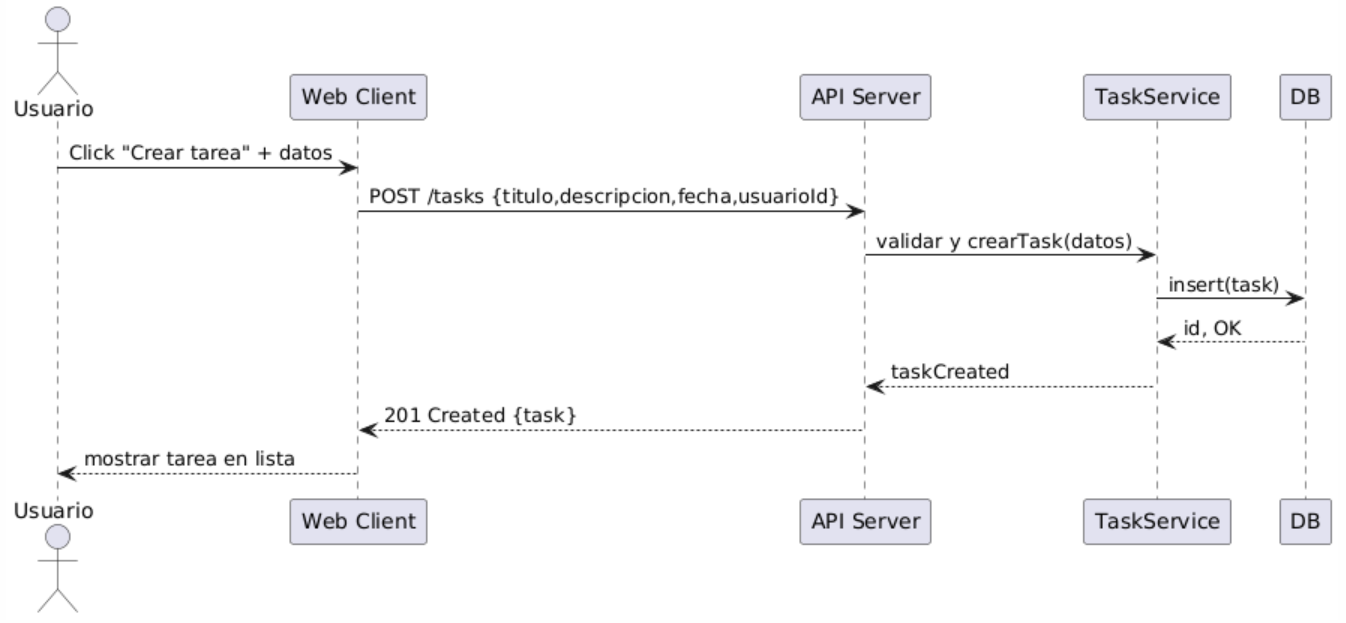
* 1. Diagrama de paquetes

Organización del código por módulos (id, services, model, persistence, utils).



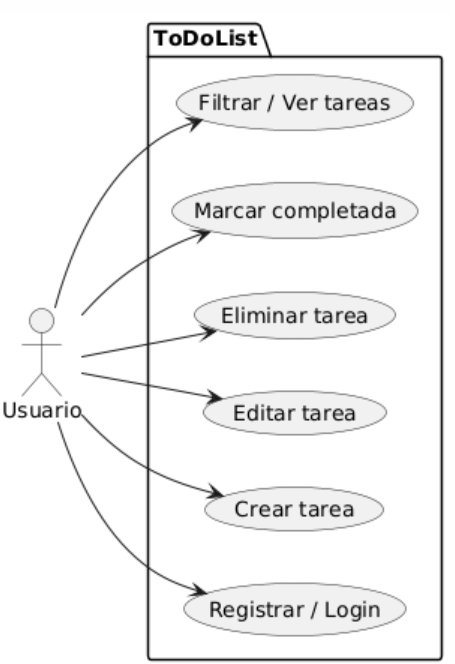
* 1. Diagrama de secuencia

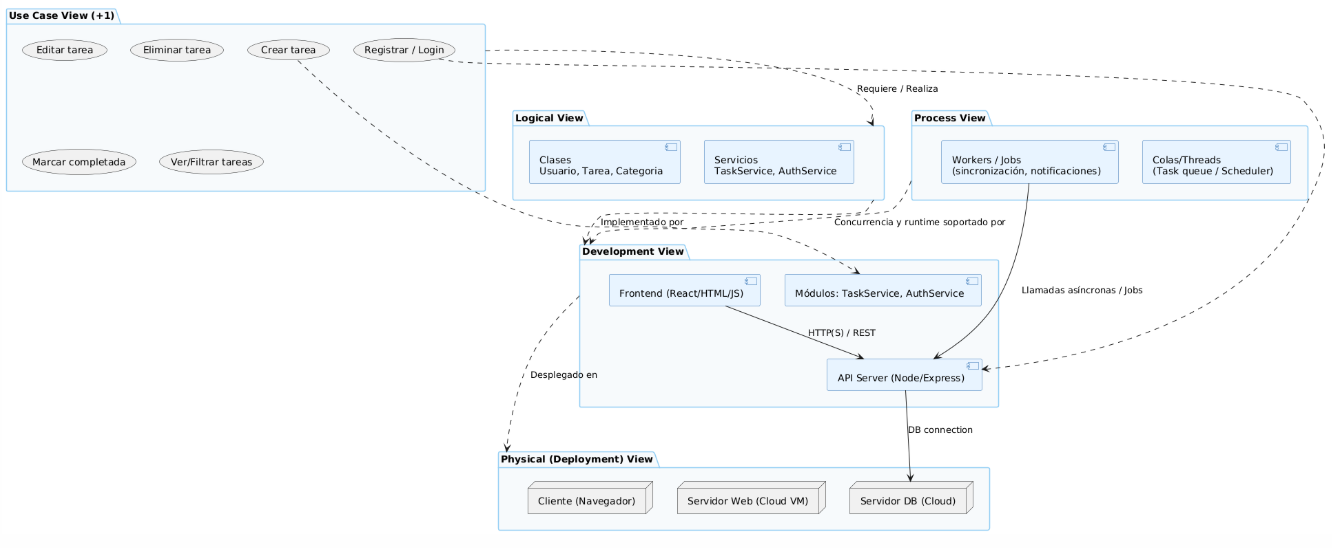
Flujo para crear tarea (Frontend→API→TaskService→DB).



* 1. Diagrama de casos de uso

Lista de interacciones principales del usuario.



* 1. Diagrama modelo 4+1
* Vista lógica (Logical): diagramas de clases, objetos; muestra la estructura estática.
* Vista de procesos (Process / Run-time): concurrencia, aspectos dinámicos.
* Vista de desarrollo (Development): Organización del software en componentes, módulos.
* Vista física (Physical / Deployment): Nodos de hardware / despliegue.
* +1 Vista de casos de uso (Use-Case): Escenarios de usuario que dirigen las otras vistas.

**BIBLIOGRAFIAS**

Ramos Martín, A., & Ramos Martín, M. J. (2014). *Aplicaciones Web 2*. Ediciones Paraninfo, SA.

Lerma-Blasco, R. V., Murcia, J. A., & Talón, A. E. M. (2013). *Aplicaciones web*. McGraw-Hill/Interamericana de España.

Cobo, Á. (2005). *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. Ediciones Díaz de Santos.

Boronat Fornés, P. J. (2012). *Gestión de tareas basada en GTD: aplicación Web* (Doctoral dissertation, Universitat Politècnica de València).

Vidal, C. L., Rivero, S. E., López, L. P., & Pereira, C. A. (2014). Propuesta y aplicación de diagramas de clases UML JPI. *Información tecnológica*, *25*(5), 113-120.

Vega Fajardo, A. (2019). Método para complementar la generación de códigos de aplicaciones web desde el diagrama de clases UML. *Revista de Ciencia y Tecnología*, (19).