Modelo de Diseño

**Testify**

OSLO

Ojeda Valeria – Sly Eduardo

Levipichun Emilio – Oyarzo Malena



El propósito del Modelo de Diseño es empezar a realizar los casos de uso desarrollados durante las etapas anteriores en especial la etapa de captura de requerimientos. Es decir, tomar el Modelo de Casos de Uso y las Especificaciones Suplementarias creadas con anterioridad entre otros insumos y generar un modelo de diseño que pueda ser usado por los desarrolladores durante la etapa de implementación.

Tabla de contenido

[Introducción 4](#_Toc257623469)

[Propósito 4](#_Toc257623470)

[Alcance 4](#_Toc257623471)

[Definiciones, siglas y abreviaturas. 4](#_Toc257623472)

[Referencias 4](#_Toc257623473)

[Visión general 4](#_Toc257623474)

[Diseño de Casos de Uso 5](#_Toc257623475)

[Diseño del Caso de Uso [nombre del caso de uso 1] 5](#_Toc257623476)

[Diagrama de paquetes 5](#_Toc257623477)

[Diagrama de Interacción 5](#_Toc257623478)

[Diseño de Flujo de eventos 5](#_Toc257623479)

[Requerimientos especiales o de implementación 5](#_Toc257623480)

[Diseño del Caso de Uso [nombre del caso de uso 2] 6](#_Toc257623481)

[Diagrama de paquetes 6](#_Toc257623482)

[Diagrama de Interacción 6](#_Toc257623483)

[Diseño de Flujo de eventos 6](#_Toc257623484)

[Requerimientos especiales o de implementación 6](#_Toc257623485)

[Diseño de Objetos 6](#_Toc257623486)

[[Objeto 1] 6](#_Toc257623487)

[[Objeto 2] 6](#_Toc257623488)

[Diseño de Subsistemas 7](#_Toc257623489)

[Subsistemas Específicos 7](#_Toc257623490)

[[Nombre del Subsistema Específico 1] 7](#_Toc257623491)

[Propósito 7](#_Toc257623492)

[Función 7](#_Toc257623493)

[Subordinados 7](#_Toc257623494)

[Dependencias 7](#_Toc257623495)

[Recursos 8](#_Toc257623496)

[Diagramas 9](#_Toc257623497)

[Diagrama de componentes 9](#_Toc257623498)

[Diagrama de Clases 9](#_Toc257623499)

[Diagrama de Secuencia 10](#_Toc257623500)

[Diagramas de Paquetes 11](#_Toc257623501)

[Diagrama de Colaboración 12](#_Toc257623502)

Modelo de Diseño

Introducción

[La introducción debe proporcionar una visión general del documento Modelo de Diseño.]

El presente documento es el Modelo de Diseño para el sistema Testify, que se encarga de la gestión de casos de prueba en proyectos de software. Aquí se detallan los aspectos técnicos y funcionales que guían la implementación de las diferentes funcionalidades descritas en las etapas anteriores de análisis y captura de requerimientos. Este documento es clave para alinear el desarrollo y la arquitectura del sistema con los requerimientos funcionales y no funcionales, asegurando que cada componente interactúe correctamente en el entorno del sistema.

Propósito

[Esta sección debe indicar el propósito del documento Modelo de Diseño y la audiencia esperada para este documento.]

El propósito del Modelo de Diseño es proporcionar una descripción detallada de cómo se implementarán los componentes del sistema Testify. Este documento está dirigido principalmente a los desarrolladores, diseñadores de software y arquitectos de sistemas que son responsables de la implementación. Además, sirve como una guía técnica para los interesados en comprender cómo el sistema ejecutará las funcionalidades requeridas. Los diagramas, especificaciones de interacción y descripciones incluidas en este documento son cruciales para la fase de desarrollo.

Alcance

[Esta sección es una breve descripción de a que se aplica el documento Modelo de Diseño, que es afectado o influenciado por este documento.]

El Modelo de Diseño se aplica a todo el sistema Testify, afectando directamente las áreas de gestión de usuarios, asignación de roles, administración de proyectos, iteraciones y escenarios de prueba. Este modelo cubre todas las capas de software involucradas, desde la interacción del usuario hasta la persistencia de datos en la base de datos. Abarca tanto la implementación del frontend como del backend del sistema, proporcionando una visión integral de las interacciones entre sus componentes.

Definiciones, siglas y abreviaturas.

[Esta sección debe proporcionar las definiciones de todos los términos, las siglas, y abreviaturas requeridas para interpretar apropiadamente el documento Modelo de Diseño. Esta información puede proporcionarse por la referencia al Glosario del proyecto.]

CRUD: Create, Read, Update, Delete. Operaciones básicas de manejo de datos en el sistema.

API: Application Programming Interface. Conjunto de funciones y procedimientos para interactuar con los componentes del sistema.

UML: Unified Modeling Language. Lenguaje estándar utilizado para representar el diseño y la arquitectura del sistema.

GUI: Graphical User Interface. Interfaz gráfica de usuario para la interacción visual con el sistema.

Testify: Sistema de gestión de casos de prueba para proyectos de desarrollo de software.

Gestor de Prueba: Usuario encargado de gestionar las pruebas de software y su progreso.

Referencias

[Esta sección debe proporcionar una lista completa de todos los documentos a los que se hace referencia en el documento Modelo de Diseño. Cada documento debe identificarse por el título, número del informe (si se aplica), fecha, y organización que lo publica. Especifique las fuentes de las que pueden obtenerse las referencias. Esta información puede proporcionarse por la referencia a un apéndice o a otro documento.]

Visión general

[Esta sección describe que contiene el resto del documento Modelo de Diseño y explica cómo se organiza este documento.]

El documento está estructurado en varias secciones que detallan el diseño de los casos de uso del sistema, la arquitectura del software y las interacciones entre los diferentes componentes. Las principales secciones incluyen:

Diseño de Casos de Uso: Describe la implementación técnica de los casos de uso definidos en la etapa de análisis.

Diseño de Subsistemas: Detalla los subsistemas clave y sus interacciones.

Diagramas de Paquetes: Representa las divisiones de responsabilidad y los componentes del sistema.

Diagramas de Interacción: Muestra cómo interactúan los distintos componentes en la ejecución de casos de uso.

Requerimientos Especiales: Consideraciones técnicas adicionales que influyen en la implementación, como limitaciones de rendimiento y consideraciones de seguridad.

Diseño de Casos de Uso

[En esta sección se especifica el comportamiento de los casos de uso, mediante subsistemas u objetos de diseño que interactúan, y se determinan las operaciones e interfaces de los distintos subsistemas u objetos de diseño.

Un objeto de diseño es una abstracción de un objeto X o varios objetos X en la implementación del sistema. Se define en base a parámetros, reglas, operaciones que realiza, requerimientos de implementación, interfaz de usuario (si corresponde) y relaciones con otros objetos.

Esta especificación se realiza mediante Diagrama de Paquetes, Diseño de flujo de eventos, Diagramas de interacción y requerimientos especiales o de implementación.]

En esta sección se especifica el comportamiento de los casos de uso mediante la definición de los subsistemas y objetos de diseño que interactúan para ejecutar cada caso de uso. A continuación, se definen los componentes involucrados en el sistema:

Objetos de Diseño: Cada objeto de diseño es una abstracción que encapsula una parte de la lógica del sistema. Estos incluyen clases como Usuario, Proyecto, Escenario, Iteración, Checklist, entre otros.

Operaciones: Se especifican las operaciones que cada objeto puede realizar. Por ejemplo, el objeto Escenario tiene operaciones para crear, actualizar, eliminar, y consultar escenarios. Estas operaciones interactúan con otros objetos como Usuario y Proyecto para asegurar una correcta gestión de pruebas.

Interfaces: Se describen las interfaces de usuario (UI) y de programación (API) que permitirán la interacción entre los componentes. La interfaz gráfica (GUI) facilita la interacción con los actores del sistema, mientras que las API permiten la integración con otros servicios.

Los diagramas de paquetes, de flujo de eventos, de interacción y los requerimientos especiales son utilizados para detallar cómo se implementan estos objetos en el sistema. A continuación, se proporciona un diagrama de paquetes para entender las relaciones entre los subsistemas, seguido de un diagrama de interacción que detalla el flujo de mensajes en un caso de uso específico (CRUD Escenario).

Diseño del Caso de Uso CRUD Escenario

Diagrama de paquetes

Este diagrama divide el sistema en paquetes que representan los diferentes subsistemas o áreas de responsabilidad que interactúan en el caso de uso CRUD Escenario.

Paquetes involucrados:

UserManagement: Contiene clases relacionadas con los usuarios, como Usuario y RolesAsignados. Proporciona acceso a los roles y permisos del usuario que ejecuta la operación.

ScenarioManagement: Contiene clases y servicios relacionados con la creación, modificación y eliminación de escenarios, como Escenario, Iteracion, Tipo, Subtipo, y Categoria.

ProjectManagement: Gestiona los proyectos en los que se crean y almacenan los escenarios, como Proyecto e Iteracion.

DocumentManagement: Gestiona los documentos que pueden ser adjuntados a los escenarios.

Database: Paquete que gestiona el acceso a la base de datos, asegurando la persistencia de los escenarios y sus cambios.

El diagrama de paquetes puede representarse de esta manera:

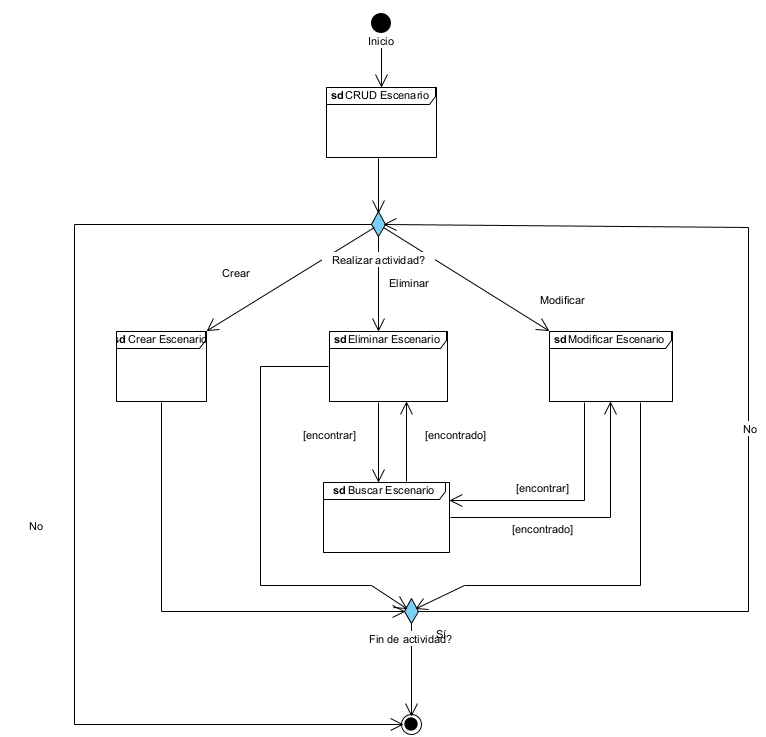
UserManagement ↔ ScenarioManagement ↔ ProjectManagement

ScenarioManagement ↔ DocumentManagement

ScenarioManagement ↔ Database

Diagrama de Interacción

El diagrama de interacción se utiliza para representar la secuencia de mensajes entre los objetos involucrados en el CRUD Escenario. Aquí se muestra cómo interactúan los objetos dentro del sistema para cumplir con las acciones del caso de uso.



**Diagrama de Secuencia:**

Usuario solicita realizar una acción (crear, modificar, eliminar o consultar) sobre un escenario.

Sistema verifica los permisos del usuario a través de UserManagement.

ScenarioService se encarga de la lógica del negocio, verificando si los detalles del escenario y la iteración son válidos.

Para crear, ScenarioService crea un nuevo objeto Escenario, lo asocia a una iteración y lo guarda.

Para modificar, ScenarioService obtiene el escenario existente, lo actualiza y lo guarda.

Para eliminar, ScenarioService elimina el escenario de la base de datos.

Para consultar, ScenarioService obtiene y devuelve la lista de escenarios asociados a la iteración.

ProjectManagement confirma que el escenario está asociado a un proyecto válido.

DocumentManagement puede ser invocado si se adjuntan documentos al escenario.

Database persiste los cambios realizados (creación, modificación, eliminación).

Diseño de Flujo de eventos

[Descripción escrita en términos de objetos y subsistemas de diseño, que explica y complementa el diagrama de interacción y sus niveles.]

Requerimientos especiales o de implementación

[Descripción que recoge los requerimientos (no funcionales) en la realización de un caso de uso.]

Diseño del Caso de Uso [nombre del caso de uso 2]

Diagrama de paquetes

[En esta sección se identifican los objetos y subsistemas de diseño que intervienen en el caso de uso y sus relaciones.]

Diagrama de Interacción

[Se sugiere realizar un diagrama de interacción para representar la realización de cada Caso de Uso.]

Diseño de Flujo de eventos

[Descripción escrita en términos de objetos y subsistemas de diseño, que explica y complementa el diagrama de interacción y sus niveles.]

Requerimientos especiales o de implementación

[Descripción que recoge los requerimientos (no funcionales) en la realización de un caso de uso.]

Diseño de Objetos

[Objeto 1]

[Se especifican los parámetros, reglas, condiciones usando la misma sintaxis del lenguaje y el código y métodos son especificados en pseudocódigo.]

* Descripción:
* Objetivo:
* Atributos:
* Métodos
* Interacciones:
* Diagrama de estado
* Interfaces que implementa

[Objeto 2]

...

Diseño de Subsistemas

Subsistemas Específicos

[Nombre del Subsistema Específico 1]

Propósito

[Descripción de por qué el subsistema existe. Este atributo debe dar la razón de la creación del subsistema. Como ser la funcionalidad específica y los requerimientos de performance por los cuales fue creado. También describe requerimientos especiales que se deben lograr con él que no están incluidos en la especificación de requerimientos del software.]

Función

[Expresa qué realiza el subsistema. Establece la transformación aplicada a las entradas del subsistema para producir la salida deseada.]

Subordinados

[Se identifican los objetos de diseño y subsistemas de diseño que componen el subsistema que se describe. Se propone representar esta información con un diagrama de paquetes.]

Dependencias

[Descripción de la relación de este subsistema con otros subsistemas de diseño. Describir la naturaleza de cada interacción incluyendo características como tiempo y condiciones de la interacción. Estas, pueden involucrar la iniciación, orden de ejecución, datos compartidos, creación, duplicación, uso o almacenamiento.

Se propone representar esta información con una tabla de dependencias.]

|  |  |  |
| --- | --- | --- |
| Subsistema del que depende | Naturaleza de interacción | Características |
| [Identificación del subsistema del que depende] | [Condiciones para que se realice la interacción] | [Características de la interacción, como ser, pasaje de parámetros, mensajes, datos compartidos, etc.] |

Recursos

[Se deben Identificar y describir todos los recursos externos al diseño que necesita el subsistema para realizar su función. Especificar las reglas de interacción y métodos para usar el recurso.

Este atributo brinda información sobre elementos como dispositivos físicos (impresoras, particiones de disco, bancos de memoria), servicios de software (librerías, servicios del sistema operativo), y recursos de procesamiento (ciclos de CPU, ubicación de memoria, buffers).

Se deben describir características de uso como el tiempo de proceso al cual se debe adquirir el recurso e incluir la cantidad de tiempo de uso. Debe incluir también la identificación de capacidad potencial y facilidades de manejo del recurso.]

# Diagramas

Diagrama de componentes

[Lo que distingue el diagrama de componentes de otro tipo de diagramas es sin duda su contenido. Normalmente contiene componentes, interfaces y relaciones entre ellos.

Los componentes pertenecen a un mundo físico, es decir, representan a un bloque de construcción al modelar aspectos físicos de un sistema.

Cada componente debe tener un nombre que lo distinga de los demás. Al igual que las clases los componentes pueden enriquecerse con compartimientos adicionales que muestran sus detalles. ]



Diagrama de Clases

[En UML el diagrama de clases es uno de los tipos de diagramas o símbolo estático y tiene como fin describir la estructura de un sistema mostrando sus clases, atributos y relaciones entre ellos. Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, en donde se intentan conformar el diagrama conceptual de la información que se manejará en el sistema. Como ya sabemos UML es un modelado de sistema Orientados a Objetos, por ende los conceptos de este paradigma se incorporan a este lenguaje de modelado.

Los diagramas de clases tienen las siguientes características:

* Las clases define el ámbito de definición de un conjunto de objetos.
* Cada objeto pertenece a una clase.
* Los objetos se crean por instanciación de las clases.

En su representación gráfica contamos con:

* Nombre de la Clase.
* Atributos de la Clase.
* Operaciones con las Clases. ]

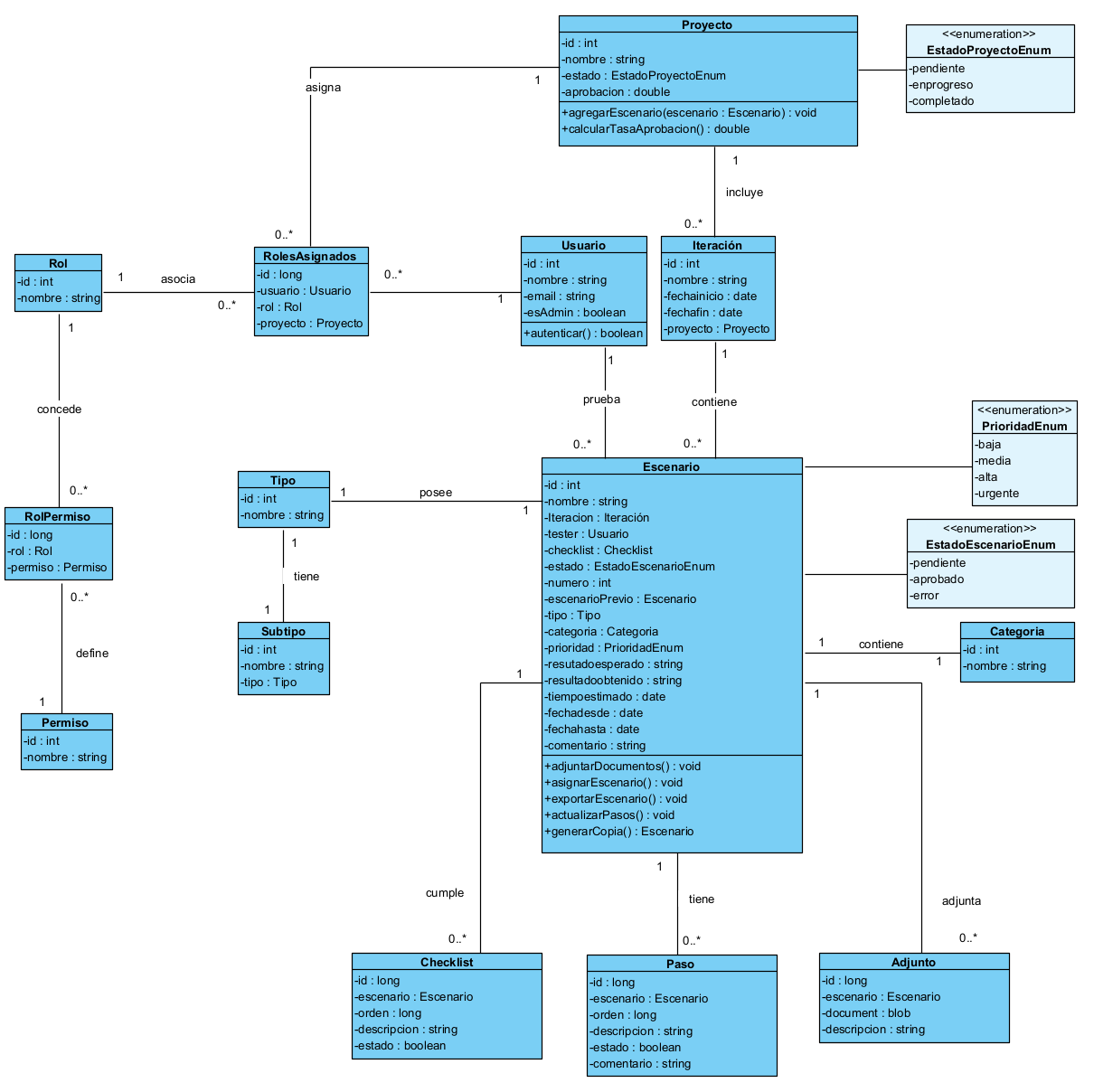


Diagrama de Secuencia

[En el diagrama de secuencia se muestra la interacción de los objetos que componen un sistema de forma temporal.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase]



Diagramas de Paquetes

[Los diagramas de Paquetes se usan para reflejar la organización de paquetes y sus elementos. Los usos más comunes de para los diagrama de paquete son para organizar diagramas de casos de uso y diagramas de clases, estos paquetes son como grandes contenedores de clases.

Los elementos contenidos en un paquete comparten el mismo espacio de nombres, esto significa que los elementos contenidos en un mismo espacio de nombres específico deben tener nombres únicos. Como otra característica de estos diagramas, cada paquete se debe identificar con un nombre único y opcionalmente mostrar todos los elementos dentro del mismo. ]



Diagrama de Colaboración

[Un diagrama de colaboración, se puede decir que es una forma alternativa al diagrama de secuencias a la hora de mostrar un escenario.

Este tipo de diagrama muestra las interacciones que ocurren entre los objetos que participan en una situación determinada. A diferencia del diagrama de secuencia, el diagrama de colaboración se enfoca en la relación entre los objetos y su topología de comunicación. En estos diagramas los mensajes enviados de un objeto a otro se representa mediante flechas, acompañado del nombre del mensaje, los parámetros y la secuencia del mensaje.

Estos diagramas están indicados para mostrar una situación o flujo de programa específico y son considerados uno de los mejores diagramas para mostrar o explicar rápidamente un proceso dentro de la lógica del programa. ]

