**Universidad de las Ciencias Informáticas**

**Facultad 7**

# Título: Análisis y Diseño de un sistema para la aplicación de técnicas de Card Sorting en la obtención de Arquitecturas de información

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

**Autor:** Jeison Mario Orovio Pino

**Tutor:** Ing. Serguey González Garay

Ciudad de La Habana, junio de 2009
"Año del 50 Aniversario del Triunfo de la Revolución"

**Agradecimientos**

*A mis padres por entregar su alma en el empeño de que sea un hombre de éxito.*

*A Lisney por ser la razón que me mueve a ser cada día mejor.*

*A mis amigos Tan, Bati y Toledo por suplir a los hermanos que nunca tuve.*

*A los profesores que me han formado que sin su apoyo hoy no fuese quien soy.*

*A Nadier, Aley, Alejandro, Nino, Rega, Adnier, Ruben, Yojan, Reinel, Roberto, Adrian por dejarme estar cerca de ellos y aprender de cada uno lo bonito que es tener amigos.*

*A Serguey por ser tutor y amigo.*

*A Leslier por sus sabios consejos.*

**Dedicatoria**

*Dedico este trabajo:*

*En primer lugar a mi abuelo Argelio, que se le daría la alegría más grande de su vida si hubiese tenido la oportunidad de vivir este momento.*

*A mis padres por ser mi escudo, mi lanza y mi látigo a lo largo de mi vida de estudiante.*

*A mi novia Lisney por soportarme y sacar de mí siempre lo mejor.*

*A mis abuelas por ocupar un pedacito de mi corazón.*

*A mis tías Mayita y Yamila por estar ahí cuando las he necesitado.*

**DECLARACIÓN DE AUTORÍA**

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los _17_ días del mes de __junio__ del año __2009__.

Jeison Mario Orovio Pino

Ing. Serguey Gonzalez Garay

**Resumen**

La creciente necesidad que tienen los profesores, trabajadores y estudiantes de organizar la información dentro de las aplicaciones que se realizan en la Universidad de las Ciencias Informáticas y la mala gestión de la información existente en la técnica de Card Sorting es lo que ha llevado a la realización de este trabajo. En el mismo se propone una vía para automatizar el proceso de la aplicación de un Card Sorting que no son más que encuestas a usuarios sobre cómo puede quedar la organización final de la información dentro de una aplicación informática.

En el presente trabajo de diploma se desarrolla el flujo de trabajo de Análisis y Diseño de la aplicación Card Sort. Está concebido bajo la metodología RUP, con Java como lenguaje de programación y la herramienta CASE Visual Paradigm para documentar el software.

El diseño elaborado permitirá una mejor realización del software para la automatización del proceso de Card Sorting algo de vital importancia para la optimización de la organización de la información dentro de una aplicación informática. También facilitará el trabajo de los arquitectos de la información que estén encargados de este proceso pues lograrán en un menor tiempo tener un resultado real de cómo estructurar la aplicación.

**Palabra Clave:** Card Sorting, Aplicación Informática, Arquitecto de Información.

## Tabla de Contenido:

**Introducción**

La década del 90 del pasado siglo se convirtió en una década de explosión de la información, Internet provocó un caos informacional sin precedentes y como se había planteado mucho antes siguieron coexistiendo la información digital e impresa. Esta explosión llevó a muchas tendencias dentro del campo de la información, que aunque siendo un problema multidisciplinario a los especialistas en información les compete una buena parte de la búsqueda de métodos para enfrentarla. Por este motivo la gestión del recurso "información" desde la década del 90 del siglo XX se impone como una función esencial de la ciencia de la información. [1]

El término Arquitectura de la Información se puede definir como el estudio de la organización de la información para llevar al usuario a encontrar un camino hacia el conocimiento y hacia la comprensión de la misma, para referirse a la organización de la información dentro de un sitio web se define como la ciencia que estructura y clasifica los mismos con el fin de que los usuarios encuentren y manejen la información fácilmente.

Para realizar una organización de categorías centrada en el usuario, el arquitecto de información dispone de técnicas de ayuda en la toma de decisiones, como es la denominada Card Sorting u Ordenación de Tarjetas.

Consiste en escribir en unas tarjetas el nombre y descripción de todas las secciones de un programa, y pedir a los usuarios que ordenen las tarjetas en grupos de una manera que les parezca lógica. Esto permite hacerse una idea de cómo los usuarios organizan mentalmente la información y da una base para organizar los menús del programa si no se estaba muy seguro. [2]

De esta forma, partiendo del comportamiento de los propios usuarios, es posible organizar y clasificar la información de una aplicación informática conforme a su modelo mental.

El proyecto no se apoya en la realización de esta técnica para la organización de la información dentro de una aplicación informática, pues actualmente se tiene que realizar de forma manual y para lograr una organización del contenido que se base realmente en lo que desee el usuario y lo que solicite el cliente hay que aplicarla a varias personas, esto genera un gran cúmulo de documentos y hace engorroso la aplicación de la misma.

Por lo anteriormente explicado se ha identificado como **problema de investigación**: ¿Cómo optimizar la gestión de la información en la realización de la técnica de ordenamiento por tarjetas o Card Sorting?

Se espera que con la automatización de esta técnica se obtenga una correcta documentación del empleo de técnicas Card Sorting que fundamente la aplicación a desarrollar. Facilitar la creación

de sistemas de etiquetado por parte de los Arquitectos de Información de la UCI a partir de la obtención de un modelo conceptual más cercano a los usuarios finales de las aplicaciones que en la universidad se desarrollan. Diseñar las funcionalidades que conforman los tres módulos de la aplicación informática, o sea, el establecimiento de bloques, el establecimiento de clases y el establecimiento de secuencia. Diseñar una funcionalidad que mediante un reporte brinde información referida a la aplicación de la técnica Card Sorting. Lograr el Análisis y Diseño del sistema a desarrollar para la utilización de la técnica Card Sorting en la obtención de Arquitecturas de Información.

El **objeto de estudio** lo constituye la Arquitectura de la Información.

El **campo de acción** de la investigación queda delimitado concretamente al proceso de aplicación de la técnica de Card Sorting en el proyecto Grupo de Investigación y Desarrollo de Internet.

Se ha identificado como **objetivo general** diseñar un sistema informático que viabilice la aplicación de la técnica de ordenamiento por tarjetas o “Card Sorting”.

Se han identificado como **objetivos específicos**:

- Profundizar la forma en que se aplica actualmente la técnica de Card Sorting.
- Nutrir al Arquitecto de Información de nuevos tipos de ejercicios que complementan los resultados de la técnica de Card Sorting.
- Hacer un minucioso levantamiento de requisitos para obtener claramente las funcionalidades del sistema.
- Identificar el flujo de procesos para que usuarios, clientes y desarrolladores tengan un entendimiento común.
- Realizar el Análisis y Diseño del sistema a desarrollar.

Para darle cumplimento a los objetivos específicos se han definido las siguientes **tareas**:

- Hacer una valoración sobre el estado del arte de la técnica Card Sorting, sistemas de etiquetado y la organización de información de un sitio web.
- Identificar los principales problemas de una ineficiente organización de la información dentro de un sitio web.
- Realizar el levantamiento de requisitos para la aplicación a desarrollar.

- Identificar las tecnologías que finalmente serán usadas en el desarrollo de la solución propuesta.
- Emplear el Visual Paradigm como herramienta de modelado a utilizar para realizar el análisis y diseño de la aplicación.
- Realizar el Análisis y Diseño de la aplicación a implementar.

El trabajo quedará finalmente estructurado en 3 capítulos:

El **Capítulo 1 Fundamentación teórica**: Incluye un estado del arte del tema tratado, a nivel internacional, nacional y de la Universidad, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad o en las que se apoya para la solución del problema que se enfrenta, sobre los que es necesario profundizar.

El **Capítulo 2 Características del sistema**: Incluye una descripción de los procesos que serán objeto de automatización, descripción general de la propuesta de sistema, como debe funcionar. Análisis comparativo de otras soluciones existentes con la propuesta, llegando a conclusiones sobre los aspectos en los que su propuesta se diferencia de las otras y las supera. Quedará plasmado el modelo del negocio así como los requisitos funcionales y no funcionales del sistema.

El **Capítulo 3 Análisis y Diseño del sistema**: Incluye la definición del modelo de análisis, del modelo de clases de análisis, se hace referencia al modelo de diseño; que incluye los diagramas de clases de diseño, los diagramas de interacción de los casos de uso del sistema más críticos y los subsistemas de diseño.

# Capitulo 1: Fundamentación Teórica

Actualmente uno de los grandes desafíos en los diseños de aplicaciones informáticas es estructurar la información de forma óptima, este se hace mediante varias técnicas para una categorización de contenidos centrada en el usuario, el Card Sorting es una de éstas técnicas que sirve como ayuda para la toma de decisiones en la etapa de diseño conceptual y para evaluar una organización concreta de categorías en etapas de evaluación de usabilidad. En este capítulo se abordan las tecnologías, metodologías y software existentes que de una forma u otra están relacionados con el tema que se aborda, además de una visión general de las mismas.

## 1.1 Que es Gestión

Coordinar todos los recursos disponibles para conseguir determinados objetivos, implica amplias y fuertes interacciones fundamentalmente entre el entorno, las estructuras, el proceso y los productos que se deseen obtener.

## 1.2 Que es Gestión de la Información

Para Gloria Ponjuán cuando se habla de gestión de información se refiere al proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico la gestión del ciclo de vida de este recurso y ocurre en cualquier organización. Es propia también de unidades especializadas que manejan este recurso en forma intensiva, llamadas unidades de información. [3]

Por lo que se puede definir a la gestión de información como el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización.

## 1.3 Arquitectura de la Información

La Arquitectura de la Información (AI) es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

En relación con la World Wide Web el Information Architecture Institute, define la Arquitectura de la Información como:

1. El diseño estructural en entornos de información compartida.

2. El arte y la ciencia de organizar y rotular sitios web, intranets, comunidades en línea y software para promover la usabilidad.

3. Una comunidad emergente orientada a aplicar los principios del diseño y la arquitectura en el entorno digital.

Su principal objetivo es facilitar al máximo los procesos de comprensión y asimilación de la información, así como las tareas que ejecutan los usuarios en un espacio de información definido.

La arquitectura de la información es un proceso iterativo, transversal, que se da a lo largo de todo el diseño del sitio y en cada una de sus fases, para asegurarse de que los objetivos de su producción y del desarrollo de la interfaz se cumplen de manera efectiva.

La Arquitectura de la Información como disciplina no busca definir una metodología de diseño universal sino articular un conjunto de técnicas para ayudar al desarrollo y producción de espacios de información como los sitios web.

Con el fin de que la asimilación de contenidos por parte del usuario sea eficiente y efectiva, y para que el sitio sea accesible y usable, la Arquitectura de la Información como proceso en general, se encarga, durante el desarrollo de definir:

- El objeto, propósito y fines del sistema de información o sitio
- La definición del público objetivo y los estudios de la audiencia.
- La realización de análisis competitivos.
- El diseño de la interacción.
- El diseño de la navegación y esquemas de facetas.
- El etiquetado o rotulado de los contenidos para acceder a la información.
- La planificación, gestión y desarrollo de contenidos.
- La facilidad de búsqueda y el diseño de la interfaz de búsqueda.
- La usabilidad. [4]

### 1.4 Sistema de Etiquetado

Las cosas reciben un nombre, el sistema de etiquetado es, por lo tanto, un sistema de representación que utiliza términos o expresiones para identificar, de la forma más inequívoca posible, el contenido informativo. Las etiquetas serán entonces términos que representen sin lugar a error, bloques significativos de información. Palabras como “Contactar”, “Documentos” o “Índice”, usadas como representantes de bloques de información más complejos, son etiquetas.

Las etiquetas resultan ser la forma más clara y evidente de mostrar la organización y navegación en aplicaciones informáticas. Los sistemas de etiquetas más comunes son de tres tipos: textuales, icónicos y combinaciones de los anteriores.

### 1.5 La Técnica de Card Sorting

Al principio de cualquier diseño de la arquitectura de un sistema de información, es frecuente enfrentarse a una lista muy larga de temas susceptibles de ser incluidos en el diseño.

El reto es organizar esta información de una manera que sea útil y significativa para los usuarios del sistema. La investigación y el análisis cuidadoso de la información pueden revelar algunas pistas, sin embargo, si se hace colaborar a los usuarios, el organizar estos temas puede ser más rentable. Así, el Card Sorting consiste en pedir que los usuarios categoricen una lista de términos. Es útil cuando ya existe la lista de temas.

Se puede diferenciar entre dos tipos de Card Sorting: abierto y cerrado.

En el Card Sorting abierto el usuario puede agrupar las categorías libremente en el número de conjuntos que crea necesario; mientras que en el cerrado, los grupos o conjuntos están predefinidos y etiquetados y el usuario únicamente deberá colocar cada categoría en el grupo que crea corresponda. Este segundo tipo de Card Sorting está recomendado para verificar si una clasificación de información es familiar y comprensible para el usuario, mientras que el "abierto" tiene el objetivo de descubrir qué tipo de clasificación de categorías sería más correcto utilizar. [5]

### 1.6 Aplicaciones Existentes

Análisis de algunas de las aplicaciones existentes que han implementado la técnica de Ordenamiento de Tarjetas.

#### CardCluster

CardCluster es otra aplicación que hace el grupo de análisis sobre los datos obtenidos a partir de alguna aplicación que brinde la realización y organización de tarjetas, hace análisis de etiqueta para visualizar las preferencias de los usuarios sobre los posibles nombres de los grupos resultantes. Esta propuesta no resuelve nuestro problema pues necesita de una aplicación precedente para su funcionamiento.

**EZSort**

EZSort es un software desarrollado por IBM, pero desde hace tiempo ya no es actualizado ni se puede descargar desde el sitio web de IBM.

**CardSword**

CardSword es una herramienta open source que permite llevar a cabo pruebas de Card Sorting, así como el análisis cuantitativo de los resultados. Permite realizar estudios del Card Sorting de forma virtual, y además puede realizar dos tipos de análisis cuantitativos de los resultados: clustering (dendograma) y redes. Actualmente se encuentra en su versión Beta 0.9, lo que la hace poco efectiva pues le quedan bastantes funcionalidades por implementar. [6]

**WEBCAT**

WEBCAT es un producto interno de ALBA Software [7] que se utiliza como plataforma base para el desarrollo de la mayoría de las aplicaciones Web que desarrollan. En estos momentos WEBCAT no se mantiene fuera de los productos de ALBA Software por lo que no representa una opción a tener en cuenta por el momento. [8]

**WebSort**

Es un instrumento de software a base de web que permite a investigadores realizar estudios de clase de tarjeta remotos. Web Sort no es gratis, lo que priva de su utilización. De todas formas permite exportar los resultados para ser analizados con otra herramienta. [9]

**CardSort**

El Card Sort se trata de qué miran primero los usuarios. Es una buena herramienta cuando los usuarios nunca han visitado el sitio o cuando se ha rediseñado profundamente. El objetivo de este test es ver si la arquitectura del sitio tiene sentido para el usuario. Una regla es que incluya categorías amplias de alto nivel de información. Cada encabezamiento debe tener un enlace a la página de inicio del sitio.

### 1.7 La técnica de Card Sorting en Cuba

En Cuba respecto al Card Sorting aunque se han hecho varias investigaciones no se ha implementado ninguna aplicación que desarrolle el ejercicio.

Debido a que esta técnica se centra en el usuario con ella se pueden diseñar productos más usables y accesibles por lo que con un mayor uso de la técnica en la universidad se lograría en un menor tiempo arquitecturas de la información óptimas y con un resultado final más acorde a las necesidades del cliente.

Con la implementación de un software que automatice esta técnica se lograría un mayor apoyo de los arquitectos de la información en ella y por ende un mayor aprovechamiento del tiempo. Cada vez que se vaya a aplicar, no se tendrían que crear tarjetas y con ello un gran cúmulo de archivos, todo estaría digital y sería mucho más viable a la hora de llegar a cualquier decisión final. Además de las ventajas que trae contar con un software como este en el país. El mismo es de fácil comercialización debido a que existen pocos similares a este y ninguno con la cantidad de funcionalidades que el mismo brindará.

Es válido mencionar en este punto la importancia que posee la presencia de las metodologías de trabajo en el desarrollo de las actividades concernientes a la ejecución del Card Sorting, pues sin su desempeño, todo lo antes expuesto no sería posible, lo cual se sustenta seguidamente.

### 1.8 Metodologías para desarrollo de software:

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo.

La comparación o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas.

A grandes rasgos, si se toma como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, se pueden clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales. Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso. A continuación se revisan brevemente cada una de estas categorías de metodologías. [10]

#### 1.8.1 Metodologías estructuradas

Los métodos estructurados comenzaron a desarrollarse en la década del 70 con la Programación Estructurada, luego aparecieron técnicas para el Diseño (por ejemplo: el diagrama de Estructura)

primero y posteriormente para el Análisis (por ejemplo: Diagramas de Flujo de Datos). Estas metodologías son particularmente apropiadas en proyectos que utilizan para la implementación lenguajes de 3ra y 4ta generación.

Ejemplos de metodologías estructuradas de ámbito gubernamental: MERISE2 (Francia), MÉTRICA3 (España), SSADM4 (Reino Unido). Ejemplos de propuestas de métodos estructurados en el ámbito académico: Gane & Sarson, Ward & Mellor, Yourdon & DeMarco e Information Engineering**.**

### 1.8.2 Metodologías orientadas a objetos

Su historia va unida a la evolución de los lenguajes de programación orientada a objeto, los más representativos: a fines de los 60's SIMULA, a fines de los 70's Smalltalk-80, la primera versión de C++ por Bjarne Stroustrup en 1981 y actualmente Java9 o C# de Microsoft. A fines de los 80's comenzaron a consolidarse algunos métodos Orientadas a Objeto.

En 1995 Booch y Rumbaugh proponen el Método Unificado con la ambiciosa idea de conseguir una unificación de sus métodos y notaciones, que posteriormente se reorienta a un objetivo más modesto, para dar lugar al Unified Modeling Language (UML), la notación OO más popular en la actualidad.

### 1.8.3 Metodologías tradicionales (no ágiles)

Las metodologías no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema.

Aunque en el caso particular de RUP, por el especial énfasis que presenta en cuanto a su adaptación a las condiciones del proyecto (mediante su configuración previa a aplicarse), realizando una configuración adecuada, podría considerarse Ágil también.

Entre las Metodologías tradicionales se encuentran:

METRICA 3

RUP (Rational Unified Process).

OPEN

### 1.8.4 Metodologías ágiles

Un proceso es ágil cuando el desarrollo de software es **incremental** (entregas pequeñas de software, con ciclos rápidos), **cooperativo** (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), **sencillo** (el método en sí mismo es fácil de

aprender y modificar, bien documentado), y **adaptable** (permite realizar cambios de último momento).

Entre las metodologías ágiles identificadas están:

• Extreme Programming .

• Scrum.

• Familia de Metodologías Crystal.

• Feature Driven Development. [11]

### 1.8.5 Metodologia Utilizada

Se decidió utilizar metodologías tradicionales pues se tiene que lograr en todo el proceso de desarrollo de software una planificación estable, haciendo hincapié en el análisis y diseño del sistema. La tabla puesta a disposición muestra elementos que sirven para fundamentar la elección de la metodología utilizada.

| Metodologías Ágiles | Metodologías Tradicionales. |
|---|---|
| Basada en heurísticas provenientes de prácticas de producción de códigos. | Basada en normas provenientes de estándares seguidos por el entorno de desarrollo. |
| Proceso menos controlado, con pocos principios | Proceso mucho más controlado, con numerosas políticas y normas. |
| No existe contrato tradicional o es bastante flexible. | Existe un contrato prefijado. |
| El cliente es parte del equipo de desarrollo. | El cliente interactúa con el equipo de desarrolladores mediante reuniones. |
| Pocos artefactos. | Más artefactos. |
| Pocos roles. | Más roles. |
| Menos énfasis en la arquitectura del software. | La arquitectura del software es esencial y se expresa mediante modelos. |

Tabla 1 Diferencias entre la Metodología Tradicional y Ágil.

**A continuación se hace un análisis de las diferentes metodologías tradicionales existentes.**

### 1.8.5 El Proceso Unificado de Desarrollo (RUP)

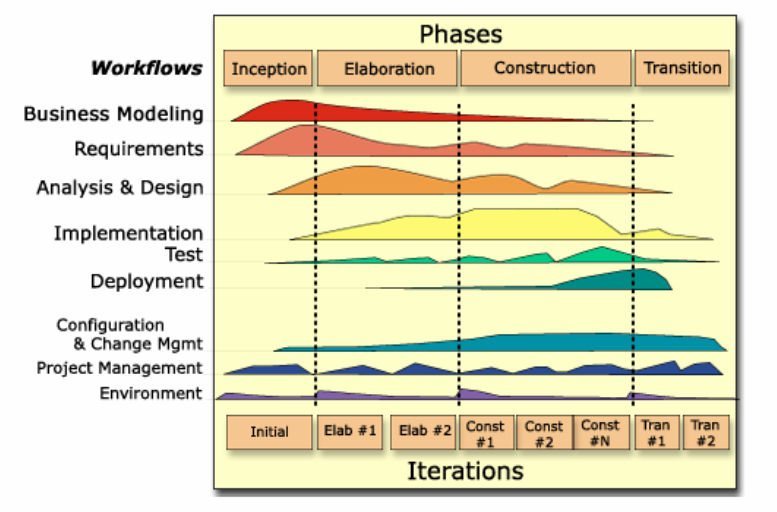
RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como

Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Como RUP es un proceso, en su modelación define como sus principales elementos:

| | |
|---|---|
| Trabajadores (“quién”) | Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos. |
| Actividades (“cómo”) | Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos. |
| Artefactos (”qué”) | Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables. |
| Flujo de actividades (“Cuándo”) | Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable. |

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. En la Figura: RUP en Dos Dimensiones se representa el proceso en el que se grafican los flujos de trabajo y las fases y muestra la dinámica expresada en iteraciones y puntos de control.

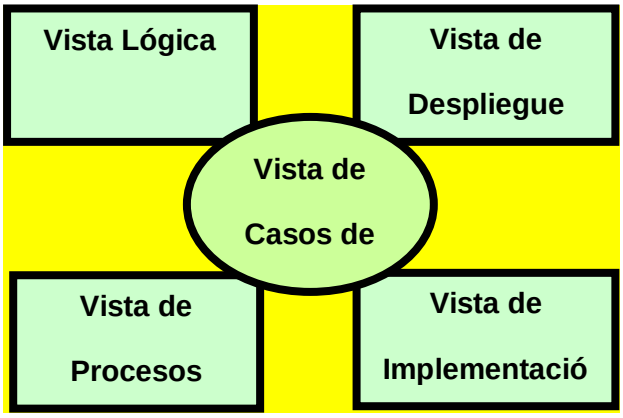



**RUP en dos Dimensiones**

El ciclo de vida de RUP se caracteriza por:

1.Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

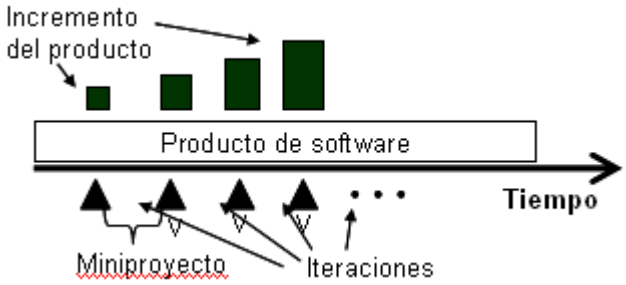
2.Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. Tal como se aprecia en la Figura, el modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.



**Vista del modelo de arquitectura**

3.Iterativo e Incremental: Aunque la figura: RUP en Dos Dimensiones puede sugerir que los flujos de trabajo se desarrollan en cascada, la "lectura" de este gráfico tiene que ser vertical y horizontal. RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son miniproyectos.

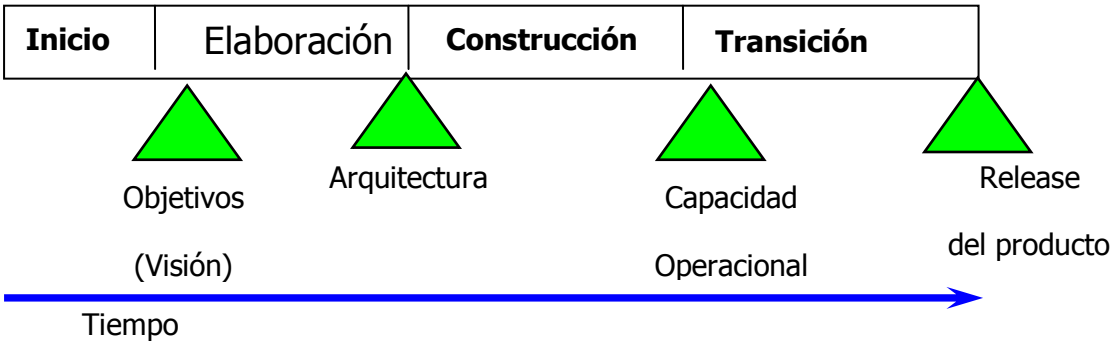



Proceso Iterativo e incremental

Aunque cada iteración tiene que proponerse un incremento en el proceso de desarrollo, todas deben aportar al principal resultado de la fase en la que se desarrolla, por lo que los puntos de control evalúan:

| | |
|---|---|
| Conceptualización | Objetivos |
| Elaboración | Arquitectura |
| Construcción | Funcionalidad operativa |
| Transición | Release del sistema |

Para lograr estos cuatro hitos, hay que construir determinados artefactos definidos dentro de los flujos de trabajo involucrados.




Fases e Hitos [12]

### 1.8.6 Métrica 3

La metodología MÉTRICA Versión 3 ofrece a las Organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software dentro del marco que permite alcanzar los siguientes objetivos:

- Proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la

Organización mediante la definición de un marco estratégico para el desarrollo de los mismos.

- Dotar a la Organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos.

- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible.

- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos.

- Facilitar la operación, mantenimiento y uso de los productos software obtenidos.

La nueva versión de MÉTRICA contempla el desarrollo de Sistemas de Información para las distintas tecnologías que actualmente están conviviendo y los aspectos de gestión que aseguran que un Proyecto cumple sus objetivos en términos de calidad, coste y plazos.

En una única estructura la metodología MÉTRICA Versión 3 cubre distintos tipos de desarrollo: estructurado y orientado a objetos, facilitando a través de interfaces la realización de los procesos de apoyo u organizativos: Gestión de Proyectos, Gestión de Configuración, Aseguramiento de Calidad y Seguridad.

La automatización de las actividades propuestas en la estructura de MÉTRICA Versión 3 es posible ya que sus técnicas están soportadas por una amplia variedad de herramientas de ayuda al desarrollo disponibles en el mercado.

Los procesos de la estructura principal de MÉTRICA Versión 3 son los siguientes:
Planificación de Sistemas de Información.
Desarrollo de Sistemas de Información.
Mantenimiento de Sistemas de Información.

El enfoque del Proceso de Planificación de Sistemas de Información, al no estar dentro del ámbito de la norma ISO 12.207 de Procesos del Ciclo de Vida de Software, se ha determinado a partir del estudio de los últimos avances en este campo, la alta competitividad y el cambio a que están sometidas las organizaciones. El entorno de alta competitividad y cambio en el que actualmente se encuentran las organizaciones, hace cada vez más crítico el requerimiento de disponer de los sistemas y las tecnologías de la información con flexibilidad para adaptarse a las nuevas exigencias, con la velocidad que demanda dicho entorno.

La existencia de tecnología de reciente aparición, permite disponer de sistemas que apoyan la toma de decisiones a partir de grandes volúmenes de información procedentes de los sistemas de gestión e integrados en una plataforma corporativa. MÉTRICA Versión 3 ayuda en la planificación de sistemas de información facilitando una visión general necesaria para posibilitar dicha integración y un modelo de información global de la organización.

En cuanto al Proceso de Desarrollo de Sistemas de Información, para facilitar la comprensión y dada su amplitud y complejidad se ha subdividido en cinco procesos:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Diseño del Sistema de Información (DSI).
- Construcción del Sistema de Información (CSI).
- Implantación y Aceptación del Sistema (IAS). [13]

### 1.8.7 OPEN

Esta metodología tiene como objetivo dotar a los Project Managers de un método para gestionar todo el ciclo de vida de los proyectos.

Se han definido cinco fases en el ciclo de vida de los proyectos:

Iniciación

La Fase de Iniciación es la primera fase en el Ciclo de Gestión del Proyecto. En esta fase se define el proyecto, necesidades del negocio, justificación del proyecto, descripción, alcance y entregables quedan reflejados en el Acta de Constitución del Proyecto

Planificación

Durante la Fase de Planificación se desarrolla el Plan de Gestión del Proyecto. En esta fase se identifica y define el alcance, las actividades los costes y se planifica el cronograma del proyecto. El Project Manager desarrolla Plan de Gestión del Proyecto que lo aprueba el Sponsor del Proyecto.

Ejecución y Control

Durante la Fase de Ejecución y Control se lleva a cabo el trabajo definido en el Plan de Gestión del Proyecto.

El Project Manager dirige y gestiona la ejecución del proyecto asegurando el nivel de calidad exigido en los requerimientos del proyecto.

- En esta fase el Project Manager:
- Adquiere el Equipo del Proyecto
- Ejecuta el Plan de Compras
- Elabora los Informe de Estado del Proyecto
- Recopila las Petición de Cambios
- Identifica las Acciones Correctoras y las Acciones Preventivas

Gestión de Cambios del Proyecto

En la Fase de Gestión de Cambios del Proyecto es cuando se aprueban o rechazan las Petición de Cambios, las Acciones Correctoras y las Acciones Preventivas.

El Project Manager propone los cambios, las acciones correctivas y preventivas que tienen que ser aprobadas o rechazadas por el Sponsor del Proyecto.

Cierre

La Fase de Cierre es la fase en la que se finaliza el proyecto formalmente.

En esta fase se chequea que el Producto, Servicio, Resultado Final cumple los objetivos del proyecto y que el Sponsor del Proyecto está satisfecho con el resultado. Además, esta fase también chequea que el contrato con el cliente y con los proveedores está cerrado y conforme.

Esta metodología solo define dos roles, si necesitas añadir más roles, procesos o ayuda para implementar o adaptar esta metodología a tu organización contacta con GEDPRO. [14]

### 1.8.8 ¿Por qué usar RUP?

La metodología de desarrollo a utilizar es la de el Proceso Unificado de Desarrollo (RUP) por su siglas en inglés debido a que, esta aplica varias de las mejores prácticas en el desarrollo moderno de software en una forma que se adapta a un amplio rango de proyectos y de organizaciones, reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente al principio, permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema, distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración, facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones lo cual además permite que se aprecien oportunidades de mejoras en el diseño.

Provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software. Esta metodología permite que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software utilizando un lenguaje de modelado común: UML puede ser adoptado y extendido para satisfacer las necesidades de la organización que lo utilice seleccionando las fases e iteraciones, los flujos de trabajo y disciplinas que se van a recorrer y los entregables o productos (artefactos) que se van a construir.

## 1.9 Lenguaje de Modelado (UML)

RUP utiliza UML (por sus siglas en inglés, Unified Modeling Language) pues es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

• Diagramas de estructura estática: Describen las propiedades estructurales del sistema.
• Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
• Diagrama de objetos: Conjunto de objetos y sus relaciones.

• Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.

• Diagramas de comportamiento:

• Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.

• Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.

• Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

• Diagramas de implementación:

• Diagrama de componentes: Organización y las dependencias entre un conjunto de componentes.
• Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

**1.10 Herramientas Case**

Existen software para el modelado en UML, herramientas CASE que permiten el modelado de sistemas.

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Herramientas software Libre

• Papyrus

- gModeler
- ArgoUML

Herramientas de libre uso (Freeware)

- Visual Paradigm for UML
- JUDE Community
- Omondo plugin para Eclipse

Herramientas comerciales

- Rational Rose de IBM
- Microsoft Visio
- MagicDraw UML [15]

### 1.10.1 Enterprise Architect

Enterprise Architect es una herramienta flexible, completa y potente de modelado en UML bajo plataforma Windows. Provee lo más nuevo en desarrollo de sistemas, administración de proyectos y análisis de negocio. Abarca integralmente el ciclo de vida, cubriendo el desarrollo de software desde el levantamiento de los requerimientos, a través de las etapas de análisis, modelos de diseño, pruebas y finalmente el mantenimiento y re-uso.

Es utilizada para el desarrollo de varios tipos de software para un amplio rango de industrias, incluyendo: bancos, desarrollo web, ingeniería, finanzas, medicina, investigación, educación, transporte, ventas, energía, ingeniería electrónica y muchas más. También es utilizado con efectividad para el entrenamiento en UML y arquitecturas de negocio en empresas de entrenamiento y universidades alrededor del mundo. Enterprise Architect 7.1 fue construido en base al excepcional éxito de las versiones previas con un completo soporte para el estándar UML 2.1 como lo ha definido la OMG (en inglés, Object Management Group).

Con Enterprise Architect, los diseñadores tienen todo el poder y la expresividad de los 13 diagramas de UML 2.1 en sus manos, incluyendo:

Diagramas de Estructura: Clases, Objetos, Compuesto, Paquetes, Componentes, Despliegue.

Diagramas de Comportamiento: Casos de uso, Comunicación, Secuencia, Interacción, Actividad, Estado.

Extensiones Temporales: Análisis, Personalizados (requisitos, diseño de UI) [16].

### 1.10.2 Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Lista de características:

Soporte de UML versión 2.1

Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento

Modelado colaborativo con CVS y Subversion (nueva característica)

Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).

Ingeniería de ida y vuelta.

Ingeniería inversa - Código a modelo, código a diagrama.

Ingeniería inversa Java, C++, Esquemas XML, XML,.NET exe/dll, CORBA IDL

Generación de código - Modelo a código, diagrama a código.

Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

Diagramas EJB - Visualización de sistemas EJB.

Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.

Diagramas de flujo de datos.

Soporte ORM - Generación de objetos Java desde la base de datos.

Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.

Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Generador de informes para generación de documentación.

Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

Importación y exportación de ficheros XMI.

Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.

Editor de figuras.

Otras herramientas y plugins de modelado UML:

Plataforma Java (Windows/Linux/Mac OS X)

SDE para Eclipse

SDE para NetBeans

SDE para Sun ONE

SDE para Oracle JDeveloper

SDE para JBuilder

SDE para IntelliJ IDEA

SDE para WebLogic Workshop

Plataforma Windows:

SDE para Microsoft Visual Studio [17].

### 1.10.3 El Rational Rose

Es la herramienta líder en el mundo de modelación visual para el proceso de modelación del negocio, análisis de requerimientos y diseño de arquitectura de componentes. Permite especificar, analizar, diseñar el sistema antes de codificarlo.

Es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. [18]

### 1.10.4 Umbrello

Umbrello es una herramienta libre para crear y editar diagramas UML, que ayuda en el proceso del desarrollo de software. Fue desarrollada por Paul Hensgen, y está diseñado principalmente para KDE, aunque funciona en otros entornos de escritorio.

Umbrello maneja gran parte de los diagramas estándar UML pudiendo crearlos, además de manualmente, importándolos a partir de código en C++, Java, Python, IDL, Pascal/Delphi, Ada, o también Perl (haciendo uso de una aplicación externa). Así mismo, permite crear un diagrama y generar el código automáticamente en los lenguajes antes citados, entre otros. El formato de fichero que utiliza está basado en XMI.

También permite la distribución de los modelos exportándolos en los formatos DocBook y XHTML, lo que facilita los proyectos colaborativos donde los desarrolladores no tienen acceso directo a Umbrello o donde los modelos van a ser publicados vía web.

En la actualidad, Umbrello permite la creación de los siguientes tipos de diagramas:

Diagrama de casos de uso

Diagrama de componentes

Diagrama de despliegue

Diagrama de modelo entidad-relación

Diagrama de clases

Diagrama de secuencia

Diagrama de estados

Diagrama de actividades

Diagrama de colaboración [19]

### 1.10.5 ¿Por qué usar Visual Paradigm?

La herramienta Case que se decidió emplear es el Visual Paradigm, pues es una herramienta libre que en la universidad ha incrementado los niveles de aceptación por su robustez, usabilidad y portabilidad, además es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; permite control de versiones; facilita la organización, visualización, diseño, integración y despliegue mediante diagramas; ayuda al equipo de desarrollo a mejorar la construcción del modelo del proceso de desarrollo de software, maximizando y acelerando la producción del equipo y las contribuciones individuales. También cuenta con una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, lo cual garantiza la calidad del producto final.

## 1.11 Entorno Integrado de Desarrollo

A solicitud del cliente y teniendo en cuenta de que los lugares en que la aplicación va a ser implantada puede no contar con conectividad se ha decidido hacer una aplicación de escritorio y no una web.

### 1.11.1 NetBeans 6.0

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans 6.0 es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans 6.0 soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Permite la edición del código, capacidades de navegación e

inspección, historia local, soporte integrado para un controlador de versiones, y mayores capacidades de personalización. [20]

### 1.12 Lenguaje de Programación

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre). [21]

### Conclusiones

En este capítulo se ha realizado un estudio del estado del arte del tema tratado analizando las aplicaciones similares desarrolladas tanto en Cuba como en el resto del mundo, arrojando la conclusión de que hay una imperiosa necesidad de crear una aplicación que automatice la técnica de Card Sorting. Se revisaron minuciosamente las diferentes metodologías existentes actualmente se compararon y adecuándose a las características de nuestro trabajo se escogió el Proceso Unificado de Desarrollo (RUP) por su siglas en ingles. La notación que se utilizará es UML. En casi todos los casos se utilizaron herramientas libres para la realización de este sistema. Utilizando así Visual Paradigm para modelar el sistema y se definió la utilización de NetBeans 6.0 para el desarrollo de la aplicación.

## Capitulo 2: Características del Sistema

Este capítulo incluye una descripción de los procesos que son objeto de automatización, descripción general de la propuesta de sistema, como debe funcionar. Se define el modelo del negocio derivándose de estos últimos los actores, trabajadores, casos de usos y diagrama de actividades, modelo de objetos; el diagrama de casos de uso del sistema; la descripción de estos así como los requisitos funcionales y no funcionales del sistema.

### 2.1 Flujo Actual de los Procesos Involucrados en la aplicación de la técnica.

Para lograr una gestión de la información centrada en el usuario utilizando la técnica de Card Sorting se tienen una serie de procesos que se ampliaran a continuación, el diseñador de la aplicación solicita al arquitecto de información realizar la arquitectura de la información a la aplicación, el Arquitecto de la Información prepara la prueba y selecciona a la audiencia, la audiencia realiza los ejercicios, se los entrega al Arquitecto de Información, el cual confecciona los resultados y se los entrega al diseñador.

#### 2.1.1 Procesos fundamentales de la Técnica

**Determinar Tópicos**

Los tópicos no deben ser ni muy genéricos ni muy específicos. Debe representar una parte de la funcionalidad que necesita organizarse. Debe evitarse las pistas que conduzcan a los usuarios a organizar los tópicos de una forma determinada.

**Crear Tarjetas**

Cada tópico se escribe en una tarjeta. En esta parte del proceso hay que ser cuidadoso pues los usuarios no pueden agrupar correctamente los elementos que no entiendan. Por lo que es importante en algunos casos tener descripciones de las mismas.

**¿Qué escribir en una tarjeta?**

Las tarjetas no son categorías intermedias, es decir, son elementos finales. No se puede poner como nombre de una tarjeta algo que tenga implícito varios tópicos dentro de ella.

**Número de tarjetas**

El número máximo de tarjetas puede ser alrededor de 50. Con más de 50 la prueba es demasiado larga, los participantes se cansan y las categorías creadas son de peor calidad. A mayor número de tarjetas, la calidad de las categorías será menor y más participantes serán necesarios.

**Tarjetas con nombres problemáticos**

Se presencia un problema grave. El usuario no puede agrupar cuando no sabe lo que está agrupando. Ante este problema se tienen dos opciones:

Se puede aprovechar el test para demostrar que los nombres de algunos tópicos son difíciles de entender, por lo que nadie los agrupa bien y por tanto nadie los encuentra en el sitio. En ese caso debería repetirse el test una vez que se cambie el nombre a esas categorías.

Si lo que se quiere es descubrir las mejores categorías posibles sin cambiar el nombre de los elementos, se puede añadir una breve explicación de 3 ó 4 palabras, al nombre de la tarjeta para que el usuario lo entienda y pueda agruparla.

**Seleccionar los participantes**

En la selección hay que tener en cuenta que éstos deben tener características y perfiles acordes con el público objetivo de la aplicación informática. De nada serviría contar con participantes jóvenes y recién titulados en ingeniería informática, cuando el sitio web está destinado a ser usado por amas de casa, sin estudios y de la tercera edad, en otras palabras deben ser los usuarios finales del producto.

**Número de participantes**

Es imprescindible como mínimo 5 participantes para realizar la prueba y obtener resultados interesantes. Sin embargo, es recomendable un mayor número de participantes.

**Hacer sesión de ordenación**

Una explicación por escrito del proceso antes de comenzar su aplicación asegura que todos tienen el mismo nivel de comprensión. En el caso del Card Sorting abierto la ordenación es sin grupos preestablecidos, siendo útil para arquitecturas nuevas; sin embargo en el caso del cerrado los grupos de ordenación están predefinidos. Si el participante es el jefe del usuario en vez del usuario final, los resultados no reflejarán un modelo mental que sea natural al usuario final. Cada sesión se debe dedicar a un grupo homogéneo de usuarios. Si se hace participar a varios grupos de usuarios, se debe hacer una sesión de Card Sorting diferente para cada grupo. Se orienta a los

participantes mirar todas las etiquetas antes de empezar la ordenación y organizar las tarjetas utilizando un criterio común. El observador observa y escucha pero no guía a los participantes durante la prueba, se recomienda utilizar un cuaderno para tomar notas sobre cualquier acontecimiento importante, pregunta o expresión no verbal que se presente durante la sesión.

**Almacenar los resultados**

Concluido el ejercicio se procede a almacenar los resultados haciendo un informe y tomando este como premisa para la realización de la aplicación informática. La práctica de este ejercicio ha demostrado que los usuarios coinciden en la agrupación entre el 60% y 80% de las tarjetas. Hay contenidos que son inherentemente difíciles de clasificar, pero si se tiene menos de un 60% de coincidencia, entonces hay que revisar las tarjetas.

¿Qué hacer con los elementos no agrupados?

Ese 20-40% de tarjetas que no han sido agrupadas por usuarios en el mismo grupo son tarjetas que presentan problemas. Frente a esto hay varias opciones.

Se puede cambiar el nombre y repetir la prueba.

Es permitido agruparlas forzosamente, aunque solo 4 de 10 usuarios las hayan puesto juntas.

Hay elementos que sencillamente no pueden ser ubicados en el contexto que se encuentran, forzar la agrupación no tendría sentido. Lo más adecuado es facilitar la localización de este elemento, situándolo en el primer nivel, como si fuese una categoría propia.

**Categorizar no es la solución**

Un margen de error de entre el 20% y 40% que da el Card Sorting no es algo aceptable como solución porque significa que ese porcentaje de usuarios o de elementos no serán localizados en la aplicación. Por lo que se infiere que los sistemas de categorización son un elemento de la interfaz que debe existir, pero más como un complemento al trabajo del arquitecto de la información.

**Análisis de los Resultados**

Con los resultados de las pruebas se tienen dos tipos de análisis, los cualitativos y los cuantitativos.

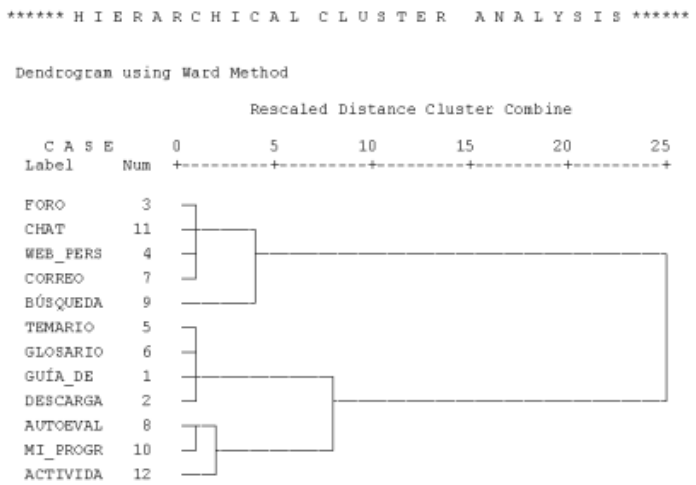
El análisis cualitativo es aconsejable hacerlo solo cuando el número de categoría y de participantes no es numeroso. Esta consiste en la observación de forma individual de cada

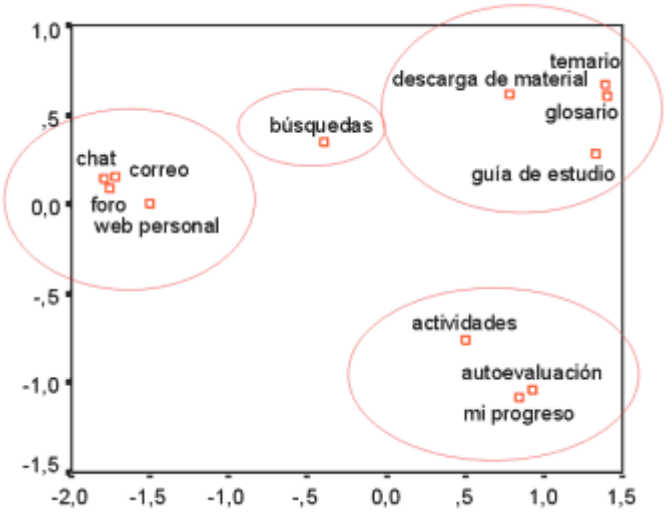
usuario durante el ejercicio, y tener en control de la forma en que cada usuario enfrenta la organización de las tarjetas, qué problemas se le presentan en el transcurso del ejercicio, y algo que es de vital importancia es estar al tanto para ver que tarjetas las agrupa de forma intuitiva y cuales le causan más dudas. Luego, se analizan detenidamente los resultados obtenidos.

Por su parte en análisis cuantitativo consiste en el procesamiento estadístico de los datos, y el posterior resumen de los resultados a través de representaciones gráficas que faciliten su interpretación por parte del arquitecto de información. Es, por tanto, un análisis adecuado para pruebas con gran número de participantes y categorías a ordenar.

Primero se crea una tabla de co-ocurrencias en una hoja de cálculo. En esta, con tantas filas y columnas como número de categorías diferentes, se indica el número de veces que cada par de categorías han sido colocadas en un mismo grupo.

Después, sobre esta tabla de co-ocurrencias se aplican algoritmos de reducción dimensional, como son los algoritmos de clustering y de escalamiento multidimensional (MDS), que tiene como objetivo simplificar las relaciones entre categorías a un número de dimensiones fácilmente interpretables por inspección visual. El resultado de aplicar a la tabla de co-ocurrencia el algoritmo de clustering es un dendograma y en el caso de el MDS lo que arroja es una representación geométrica de las categorías. [22]






### 2.2 Objeto automatización

En el proyecto Grupo de Investigación y Desarrollo sobre Internet (GIDI) el proceso de gestión de la información centrada en el usuario no está automatizado, provocando que se haga poco uso de la misma para la organización de la información de las aplicaciones informáticas. Esto hace que para la obtención de una buena arquitectura de la información se genere un gasto de tiempo excesivo.

Para darle solución a esta problemática se decidió automatizar el proceso de crear las tarjetas: es decir definirlas y ponerles nombre; preparar los bloques: es cómo hacer subgrupos; crear clases: consiste en definir las clases que posteriormente se agruparan en bloques; preparar las instrucciones: aquí es donde se elabora una series de preguntas en cuanto a las dudas que la audiencia pueda tener a la hora de realizar el ejercicio; preparar plantillas, y almacenar al final las agrupaciones de clases, de bloques y el orden en el caso del ejercicio de secuencia.

### 2.3 Propuesta de Sistema

El sistema contará con 3 módulos, Diseñar Ejercicio, Realizar Ejercicio y Procesar Datos.

El módulo Diseñar Ejercicio debe permitir preparar orientaciones y ejercicios de establecimientos de clases, secuencia y bloques, incluyendo la posibilidad de combinarlos, modificarlos y de crear un nuevo proyecto, se guarda en un fichero. El módulo de Realizar Ejercicio debe permitir acceder a ver las orientaciones, la lectura de ficheros con los ejercicios preparados por los arquitectos de información, realizar los ejercicios indicados y guardar la misma en el fichero de ejercicios

resueltos. El módulo Procesar Datos debe brindar la posibilidad de ver gráficamente el resultado general de la técnica del Card Sorting.

### 2.4 Modelo del Negocio

Para conseguir sus objetivos, una entidad organiza su actividad por medio de un conjunto de procesos de negocio. Cada uno de ellos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes (por ejemplo, trabajadores o departamentos) participan de acuerdo a un flujo de trabajo determinado. Además, estos procesos se hallan sujetos a un conjunto de reglas de negocio, que determinan la estructura de la información y las políticas de la entidad. Por tanto, la finalidad del modelado del negocio es describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio.

#### 2.4.1 Descripción de los procesos del negocio

La técnica del Card Sorting suele realizarse en un local donde un grupo de personas son reunidas en calidad de audiencia, se le suministra un grupo de tarjetas de cartón numeradas y con un concepto en el reverso. La audiencia las agrupa en dependencia del ejercicio que previamente se les orientó realizar. El Card Sorting consta de tres ejercicios:

Establecimiento de Clases: La audiencia agrupa según el criterio propio las tarjetas en clases.

Abierto: La audiencia crea tantas clases como entienda necesarias.
Cerrado: La audiencia no puede crear clases, tiene que ajustarse a las predefinidas por el arquitecto.

Establecimiento de Secuencia: Se procesa el orden en que la audiencia agrupó las clases. Establecimiento de Bloque: La audiencia coloca cada clase en un bloque determinado. Los resultados de la prueba son recuperados, procesados por el arquitecto de la información y luego se guardan en una carpeta.

#### 2.4.2 Actores del Negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. [23]

| Actor | Descripción |
|---|---|
| Arquitecto de Información | Persona encargada de confeccionar y aplicar la técnica de Card Sorting. |

| | |
|---|---|
| Diseñador | Persona que resulta el beneficiario fundamental del Negocio. |

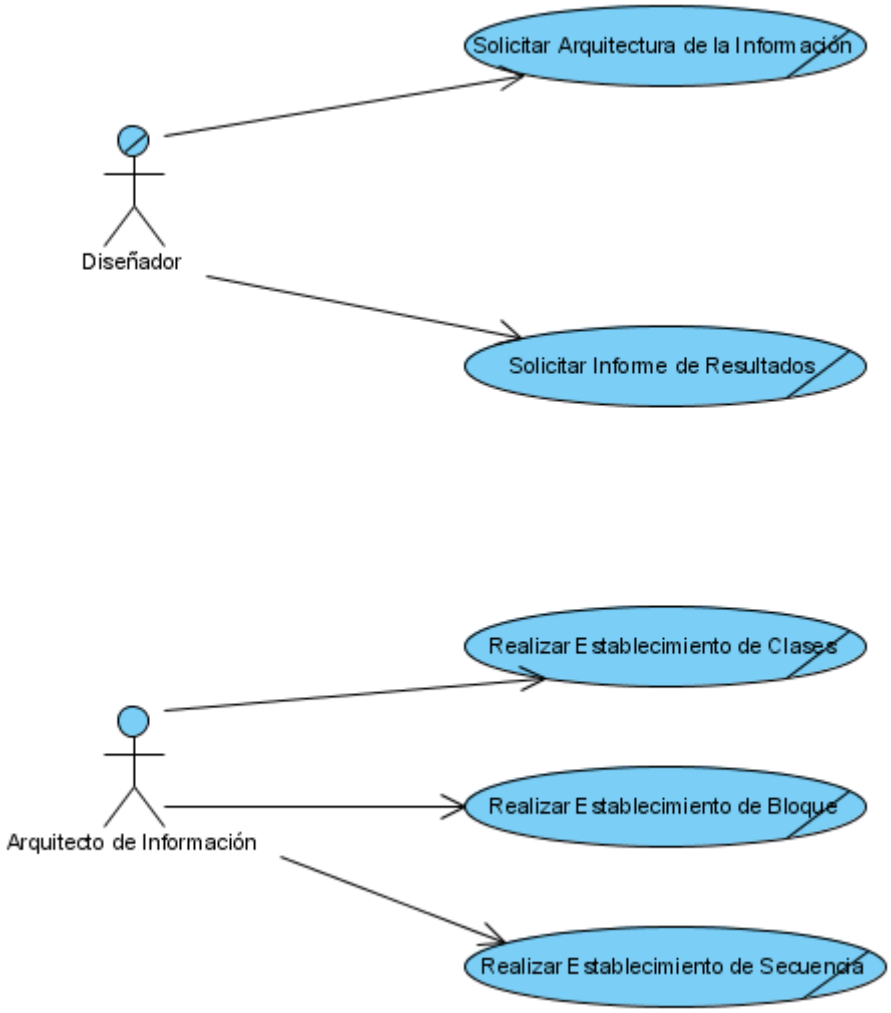
### 2.4.3 Trabajadores del Negocio

Un trabajador del negocio representa una persona o un sistema automatizado (software) que actúa en el negocio realizando una o varias actividades comprendidas dentro del caso de uso, interactuando con otros trabajadores del negocio y manipulando entidades del negocio.

| Trabajador | Descripción |
|---|---|
| Audiencia | Persona común que realiza los ejercicios del Card Sorting. (Preferiblemente el cliente). |
| Arquitecto de Información | Persona encargada de Aceptar o no la aplicación de un ejercicio y de brindar los resultados del mismo. |
| Observador | Encargado de orientar y supervisar a la audiencia. |

### 2.4.4 Diagrama de Casos de Uso del Negocio

Un caso de uso del negocio representa un proceso dentro del negocio que se estudia, por lo que se corresponde con una secuencia de acciones con un orden lógico y que producen un resultado observable para ciertos actores del negocio.



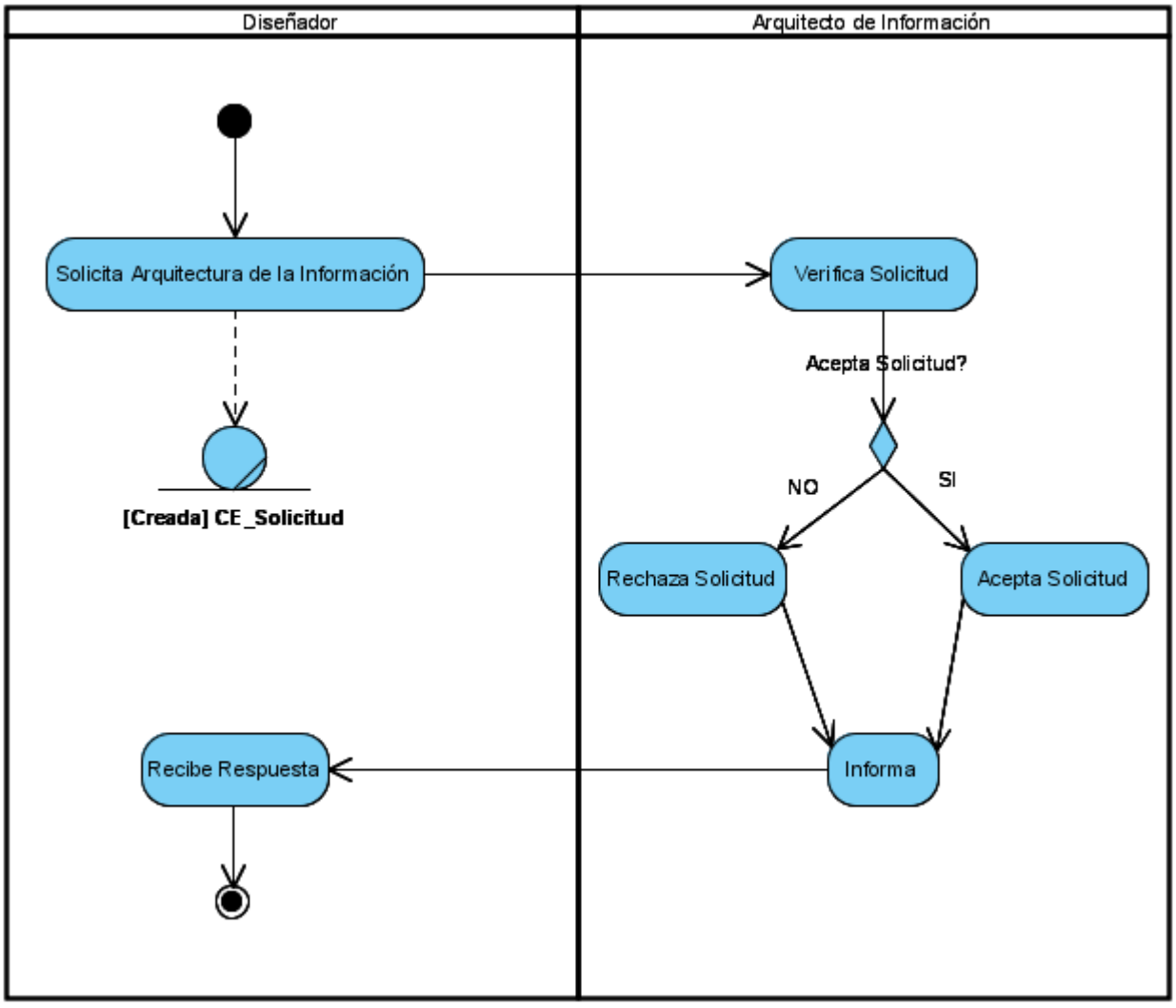

### 2.4.5 Descripción de Casos de Usos del Negocio.

**Descripción del Caso de Uso Solicitar Arquitectura de la Información**

| | |
|---|---|
| **Caso de Uso:** | **Solicitar Arquitectura de la Información** |
| **Actores:** | **Diseñador** |
| **Trabajadores:** | **Arquitecto de la Información** |
| **Resumen:** | **El Caso de Uso comienza cuando el Diseñador solicita la realización de la arquitectura de la información a una aplicación informática termina con la aceptación o no por parte del arquitecto de la misma, finalizando así el caso de uso.** |
| **Precondiciones:** | **El diseñador tiene que tener lista toda la información necesaria para aplicar la técnica.** |

| Flujo Normal de Eventos | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. Solicita la Arquitectura de la Información.** | **1.1 Acepta o no la Solicitud.** |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| | |
| **Poscondiciones** | **Quedan listas las pruebas a aplicar.** |

**Diagrama de Actividades del Caso de Uso Solicitar Arquitectura de la Información**



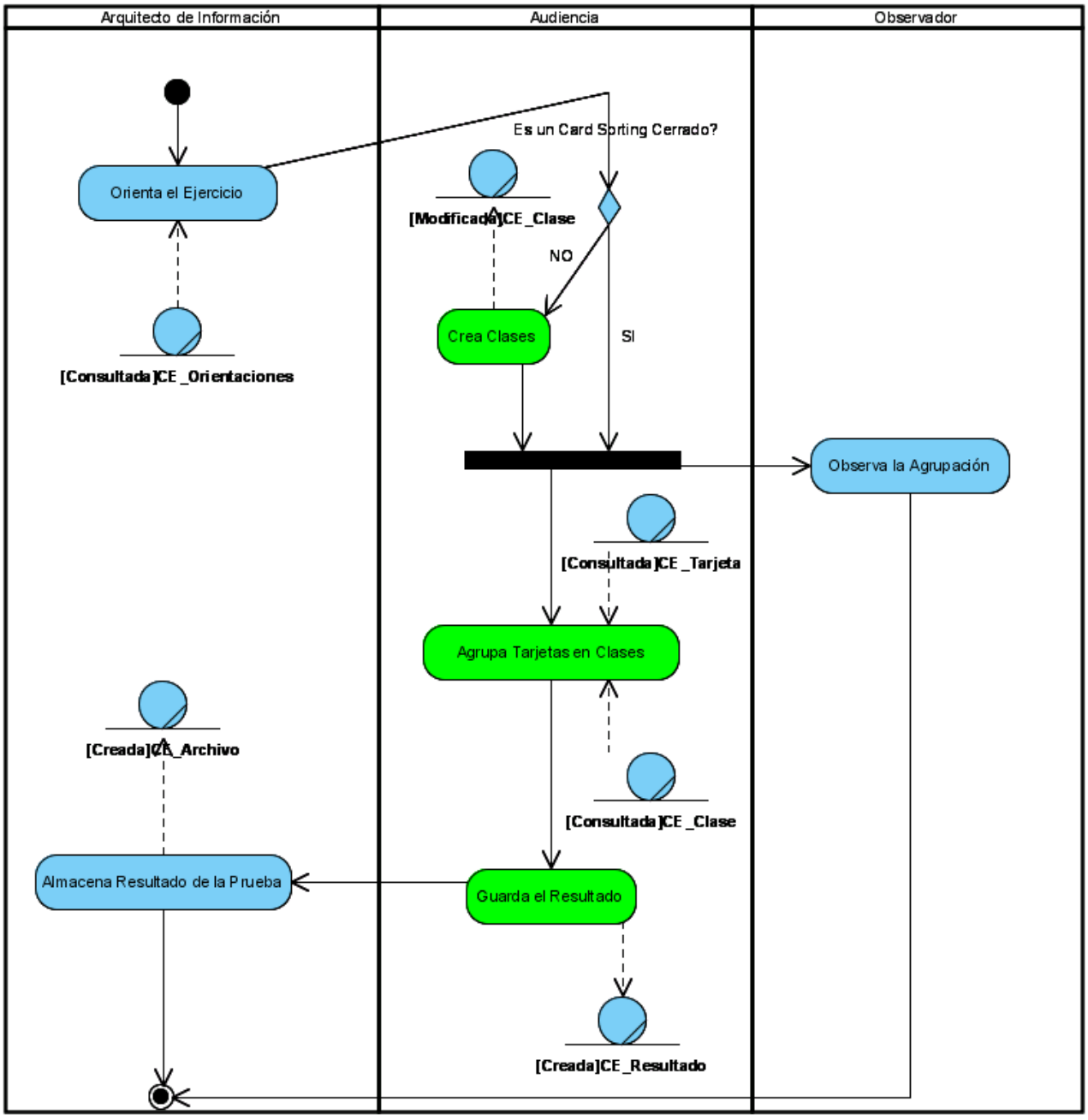
**Descripción del Caso de Uso Realizar Ejercicio Establecimiento de Clases**

| | |
|---|---|
| **Caso de Uso:** | **Realizar ejercicio Establecimiento de Clases** |
| **Actores:** | **Arquitecto de Información** |
| **Trabajadores:** | **Audiencia** |
| **Resumen:** | **El Caso de Uso comienza con la facilitación por parte del arquitecto de información de las tarjetas y/o clases, en caso de ser un Card Sorting cerrado la audiencia solo agrupará las tarjetas en clases predefinidas, en caso de ser un Card Sorting abierto la audiencia agrupará las tarjetas en clases creadas basándose en su criterio, culmina con la recolección de los resultados, finalizando así el caso de uso.** |
| **Precondiciones:** | **La audiencia debe ser instruida para realizar el ejercicio y debe poseer las tarjetas y clases.** |

| **Flujo Normal de Eventos** | |
|---|---|
| **Card Sorting Cerrado** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. El Arquitecto de Información facilita a la audiencia las clases y tarjetas para realizar el ejercicio.** | **1.1 El Observador analiza detalladamente el agrupamiento de las tarjetas en clases.** |
| | **1.2 El Arquitecto almacena los agrupamientos (clases) de tarjetas.** |
| **Card Sorting Abierto** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. El Arquitecto de Información facilita a la audiencia las tarjetas para realizar el ejercicio y le pide crear las clases.** | **1.1 El Observador analiza detalladamente la creación de las clases** |
| | **1.2 El Observador analiza detalladamente el agrupamiento de las tarjetas en clases.** |
| **3. Guarda los Resultados** | **3.1 Se almacenan los agrupamientos (clases) de tarjetas.** |

| Flujos Alternos | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| | |
| **Poscondiciones** | **Quedan establecidas las clases para su recuperación.** |

**Diagrama de Actividades del Caso de Uso Realizar ejercicio Establecimiento de Clases**



**Descripción del Caso de Uso Realizar ejercicio Establecimiento de Secuencia**

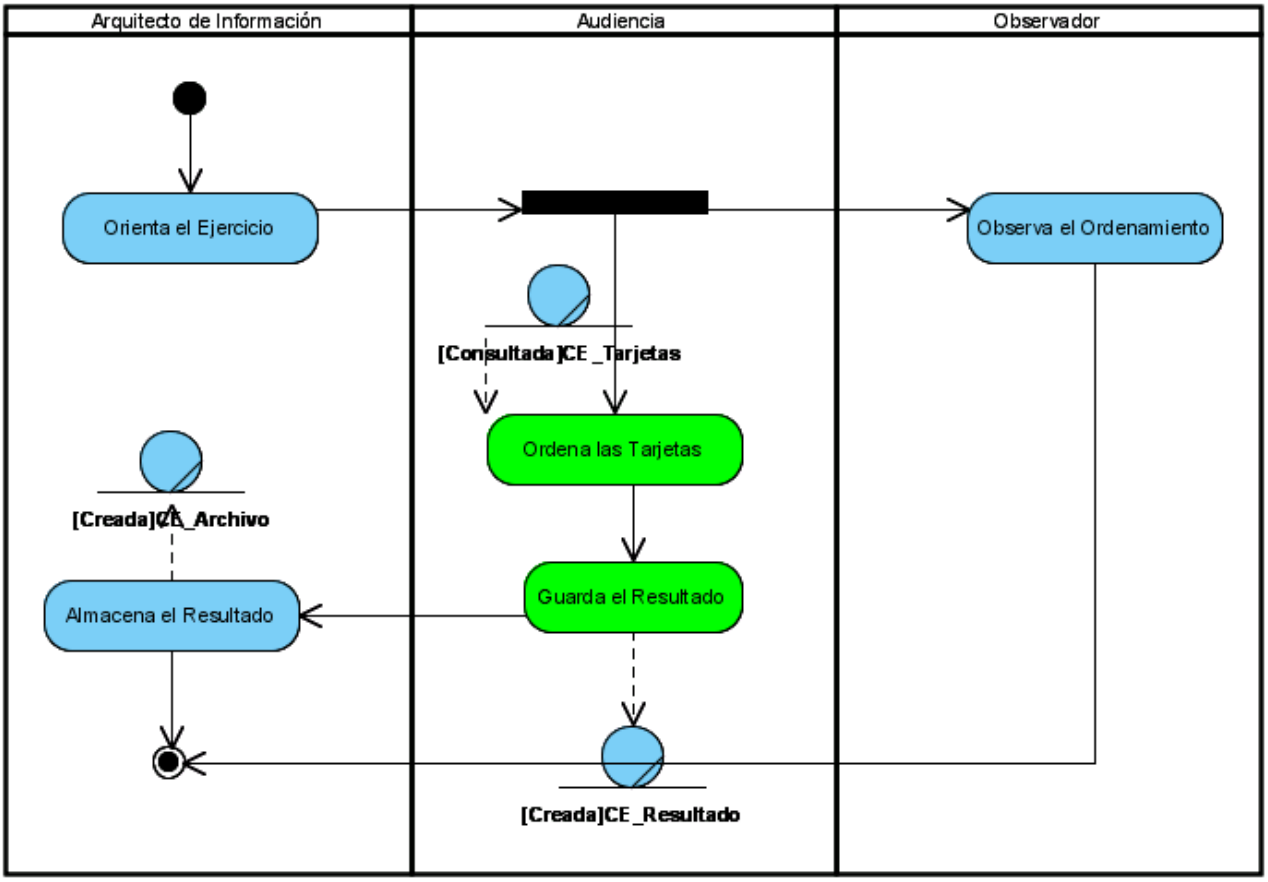
| | |
|---|---|
| **Caso de Uso:** | **Realizar ejercicio Establecimiento de Secuencia** |
| **Actores:** | **Audiencia** |
| **Trabajadores:** | **Observador, Arquitecto de Información.** |
| **Resumen:** | **El Caso de Uso comienza cuando el arquitecto información facilita las clases y tarjetas a la audiencia que agrupa las basándose en su criterio propio, en este caso es importante el orden dado en ese agrupamiento termina con la recuperación de esos datos por parte del arquitecto de información, finalizando así el caso de uso.** |
| **Precondiciones:** | **La audiencia debe ser instruida para realizar el ejercicio y debe poseer las tarjetas y las clases** |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. El Arquitecto de Información facilita a la audiencia las clases y tarjetas para realizar el ejercicio.** | **1.1 El observador analiza detalladamente el ordenamiento.** |
| **2. Guarda los Resultados.** | **2.1 Se almacenan el orden de dicho agrupamiento.** |

| **Flujos Alternos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| | |

| | |
|---|---|
| **Poscondiciones** | **Quedan establecidas las secuencias que deben ser recuperadas** |

**Diagrama de Actividades del Caso de Uso Realizar ejercicio Establecimiento de Secuencia**



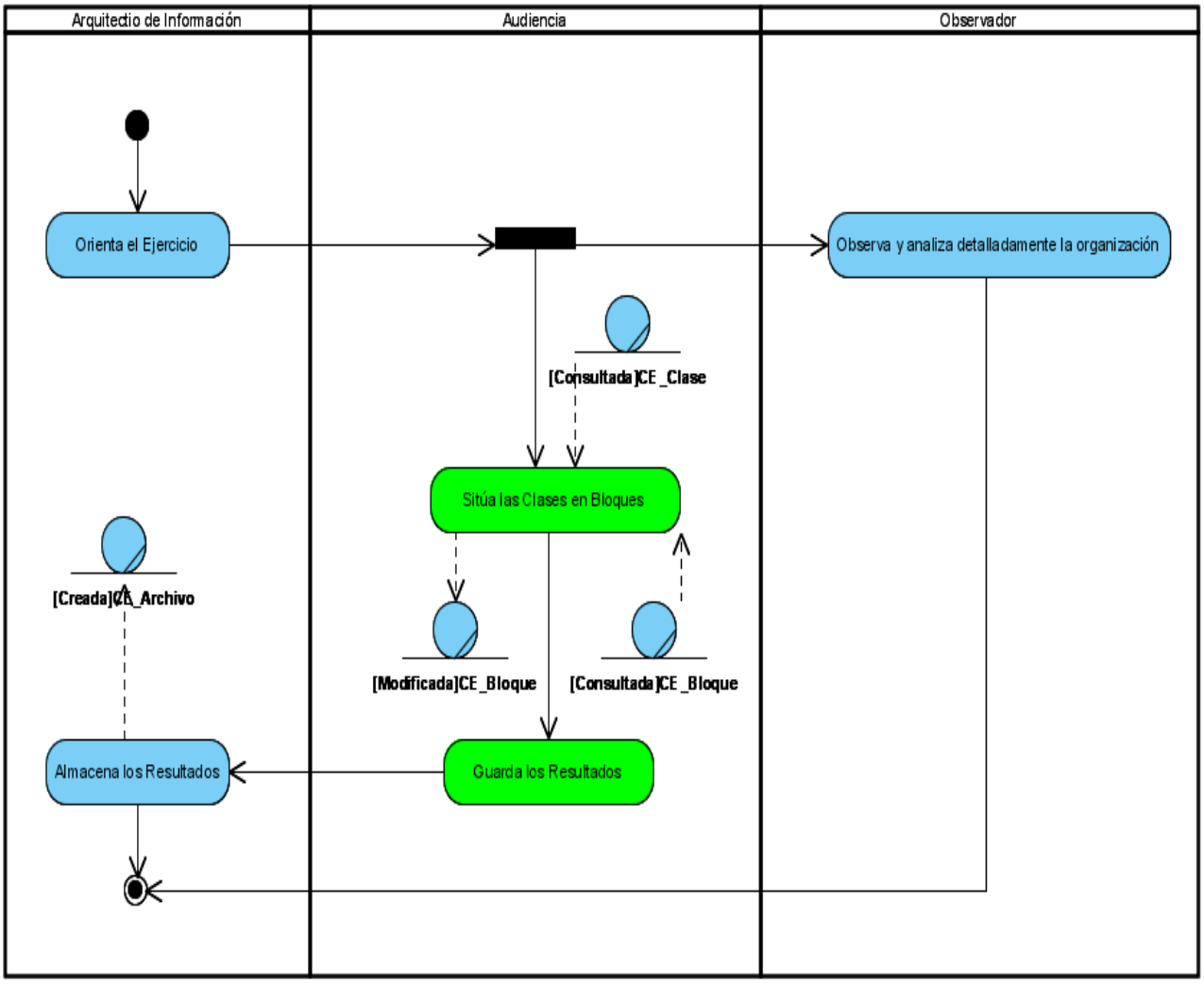

**Descripción del Caso de Uso Realizar ejercicio Establecimiento de Bloques**

| | |
|---|---|
| **Caso de Uso:** | **Realizar ejercicio Establecimiento de Bloques** |
| **Actores:** | **Audiencia** |
| **Trabajadores:** | **Observador, Arquitecto Información.** |
| **Resumen:** | **El Caso de Uso comienza con la facilitación por parte del arquitecto de información de las tarjetas y/o clases, la audiencia realiza el agrupamiento por bloques de las clases y/o tarjetas. Concluye con la recolección de los resultados por parte del arquitecto de la información, finalizando así el caso de uso.** |
| **Precondiciones:** | **La audiencia debe ser instruida para realizar el ejercicio y además debe poseer las clases y los bloques para agrupar.** |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. El Arquitecto de Información facilita a la audiencia las clases** | **1.1 El Observador analiza detalladamente** |

| | |
|---|---|
| **para realizar el ejercicio.** | |
| **2. Guarda los resultados.** | **2.1 Se almacenan los agrupamientos de clases de acuerdo a los bloques.** |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| | |
| **Poscondiciones** | **Quedan establecidos los bloques, que deben ser recuperados.** |

**Diagrama de Actividades del Caso de Uso Realizar Ejercicio Establecimiento de Bloques**



**Descripción del Caso de Uso Solicitar Resultados**

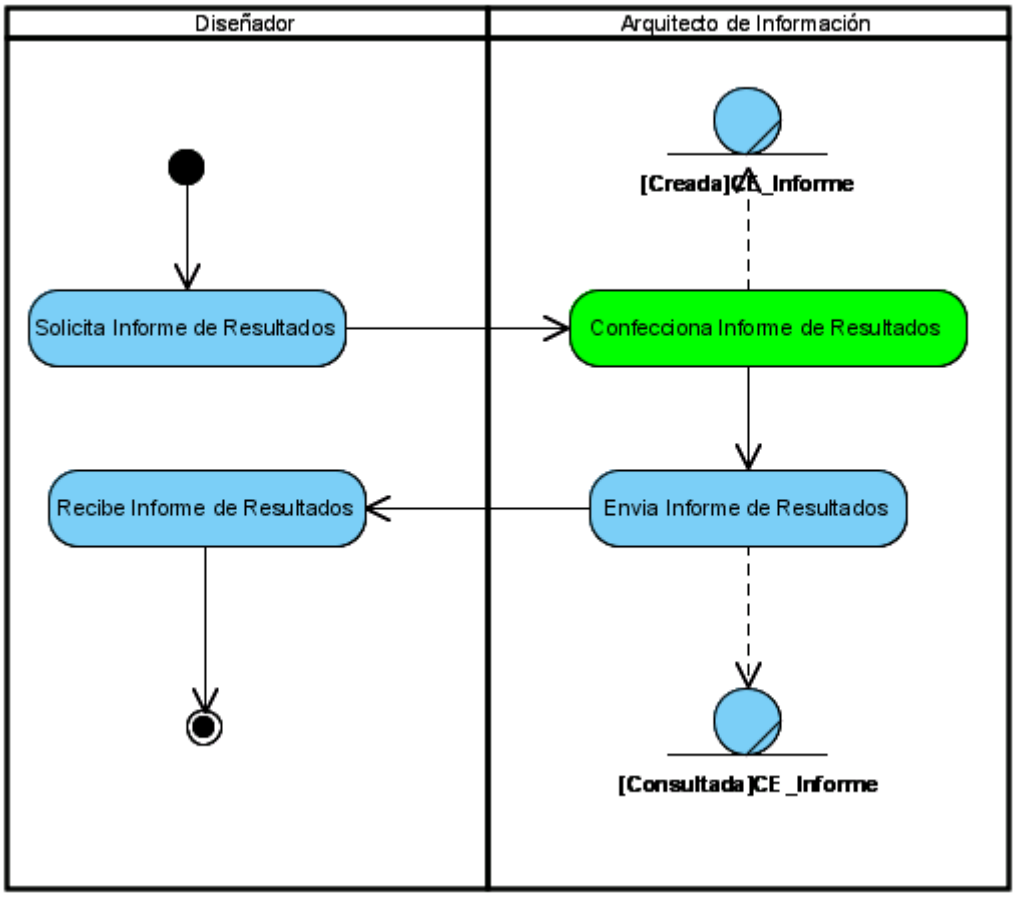
| | |
|---|---|
| **Caso de Uso:** | **Solicitar Resultados** |
| **Actores:** | **Diseñador** |
| **Trabajadores:** | **Arquitecto de la información** |
| **Resumen:** | **El Diseñador solicita al Arquitecto de la información un informe sobre los resultados de la aplicación del Card Sorting. El Arquitecto de la información procesa los resultados recuperados, de los ejercicios aplicados mediante técnicas de clustering y entrega los mismos, finalizando así el caso de uso.** |
| **Precondiciones:** | **Se deben haber realizado el o los ejercicios de Card Sorting previamente.** |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| **1. Solicita informe se resultados** | **1.2 Se elabora un Informe de Resultados.** |
| | **1.3 Se envía el Informe de Resultados** |
| **2. Recibe Informe de Resultados** | |

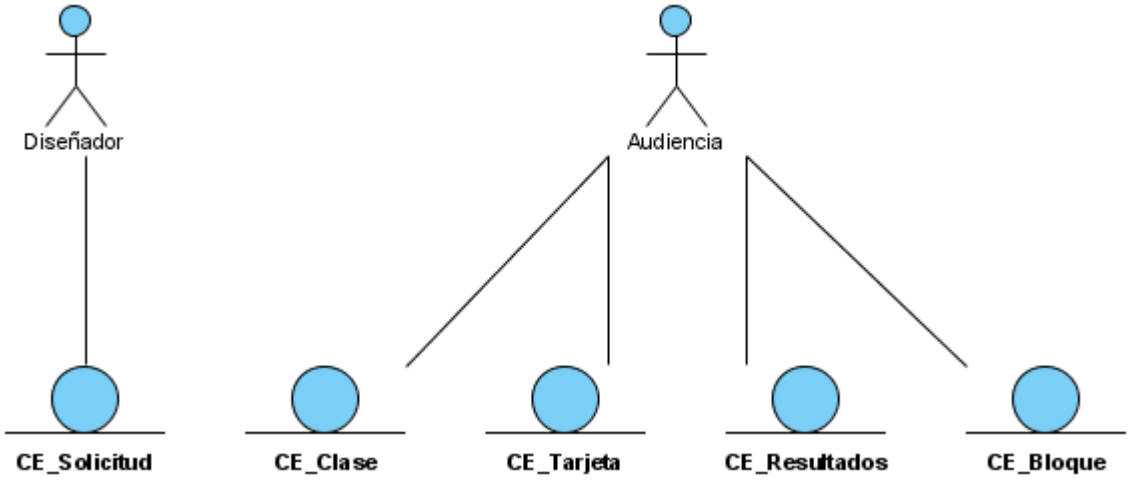
| **Flujos Alternos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Negocio** |
| | |

| | |
|---|---|
| **Poscondiciones** | **Se obtienen los patrones más persistentes entre la audiencia.** |

**Diagrama de Actividades del Caso de Uso Solicitar Resultados**




## 2.5 Modelo de Objetos



## 2.6 Requerimientos del Sistema

Se desarrolla en la fase de Inicio del RUP y tiene como entrada el modelamiento de negocio. Su resultado final una vez concluida la fase es tener acabado el Modelo del sistema. El levantamiento de los requisitos tiene como objetivos:

- Identificar, enunciar y clasificar los requerimientos.
- Obtener las propiedades o cualidades que el producto debe cumplir.
- Obtener un listado de las capacidades o funciones que el sistema debe cumplir.
- Determinar actores y casos de uso del sistema.

### 2.6.1 Requisitos Funcionales

Los requisitos funcionales son condiciones o capacidades que el sistema debe cumplir. Este sistema debe ser capaz de:

RF 1. Preparar Orientaciones de los ejercicios de Card Sorting.
RF 2. Preparar Clases de la temática a diseñar.
RF 3. Crear Tarjetas con el contenido necesario que se estime conveniente.
RF 4. Modificar Tarjetas.
RF 5. Eliminar Tarjetas.
RF 6. Crear Clases.
RF 7. Modificar Clases.
RF 8. Eliminar Clases.
RF 9. Preparar Ejercicios de Establecimiento de Clases.
RF 10. Actualizar Ejercicios de Establecimiento de Clases.
RF 11. Preparar Ejercicios de Establecimiento de Bloques.
RF 12. Actualizar Ejercicios de Establecimiento de Bloques.
RF 13. Preparar Ejercicios de Establecimiento de Secuencia.
RF 14. Actualizar Ejercicios de Establecimiento de Secuencia.
RF 15. Guardar ejercicios preparados en un fichero.
RF 16. Mostrar ejercicio o proyecto de Card Sorting.
RF 17. Mostrar orientaciones de los Ejercicios Preparados.
RF 18. Realizar Ejercicios de Establecimiento de Clases Abierto.
RF 19. Realizar Ejercicios de Establecimiento de Clases Cerrado.
RF 20. Realizar Ejercicios de Establecimiento de Bloques.

RF 21. Determinar las clases seleccionadas en el Establecimiento de Clases para que la audiencia haga sobre ella el Establecimiento de Secuencia.
RF 22. Realizar Ejercicios de Establecimiento de Secuencia.
RF 23. Actualizar Ejercicios de Establecimiento de Secuencia.
RF 24. Ver descripción de las clases y tarjetas creadas en los sets del proyecto de Card Sorting
RF 25. Guardar diseño de ejercicio o proyecto de Card Sorting.
RF 26. Exportar Proyecto Card Sorting.
RF 27. Procesar los resultados de los ejercicios aplicados a la Audiencia
RF 28. Aplicar el algoritmo de clustering a los ejercicios.
RF 29. Mostrar resultados finales de los ejercicios.
RF 30. Obtener el Informe de los resultados finales.

### 2.6.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener para su máximo rendimiento.

**Software**

Para la implementación del sistema se debe disponer de un Sistema Operativo Windows 98 o superior y del NetBeans 6.0.

**Apariencia o Interfaz Interna**

El software debe contar con una interfaz fácil, amigable, sencilla, permitiendo que la audiencia sea capaz de interactuar con este aún teniendo conocimientos básicos y se sienta familiarizado con el sistema.

**Usabilidad**

El sistema será flexible y de fácil aprendizaje, pues se trata en todo lo posible de mantener un estándar de operatividad que logre que las interacciones del usuario con el sistema sean predecibles y familiares. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

**Rendimiento**

Las pantallas estarán poco cargadas de imágenes para garantizar la ejecución de los hipervínculos, las adiciones, modificaciones no excedan los 4 segundos y garantizar de esta manera una respuesta rápida del sistema.

**Soporte**

Debe elaborarse un paquete de instalación que abarque verificación de componentes ya instalados y la instalación de los nuevos.

**Portabilidad**

El sistema podrá ejecutarse sobre plataforma Linux, Windows 98 o superior.

**Seguridad**

La herramienta garantiza que la información sea actualizada únicamente por quien tiene acceso a trabajar con el sistema. Tiene protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

**Confiabilidad**

La herramienta a utilizar tiene un mecanismo eficiente para recuperar las pérdidas de información ante fallos, errores en caso de alguna ocurrencia, por lo que deberá existir un plan de salvas y mantenimiento garantizando con esto una rápida protección y recuperación ante un problema dado. También debe montarse sistemas de respaldo eléctrico en los locales de los servidores para mantener la vitalidad de los servicios.

**Interfaz**

El diseño de la aplicación debe ser con colores claros, adecuado y lo más legible posible.

**Ayuda**

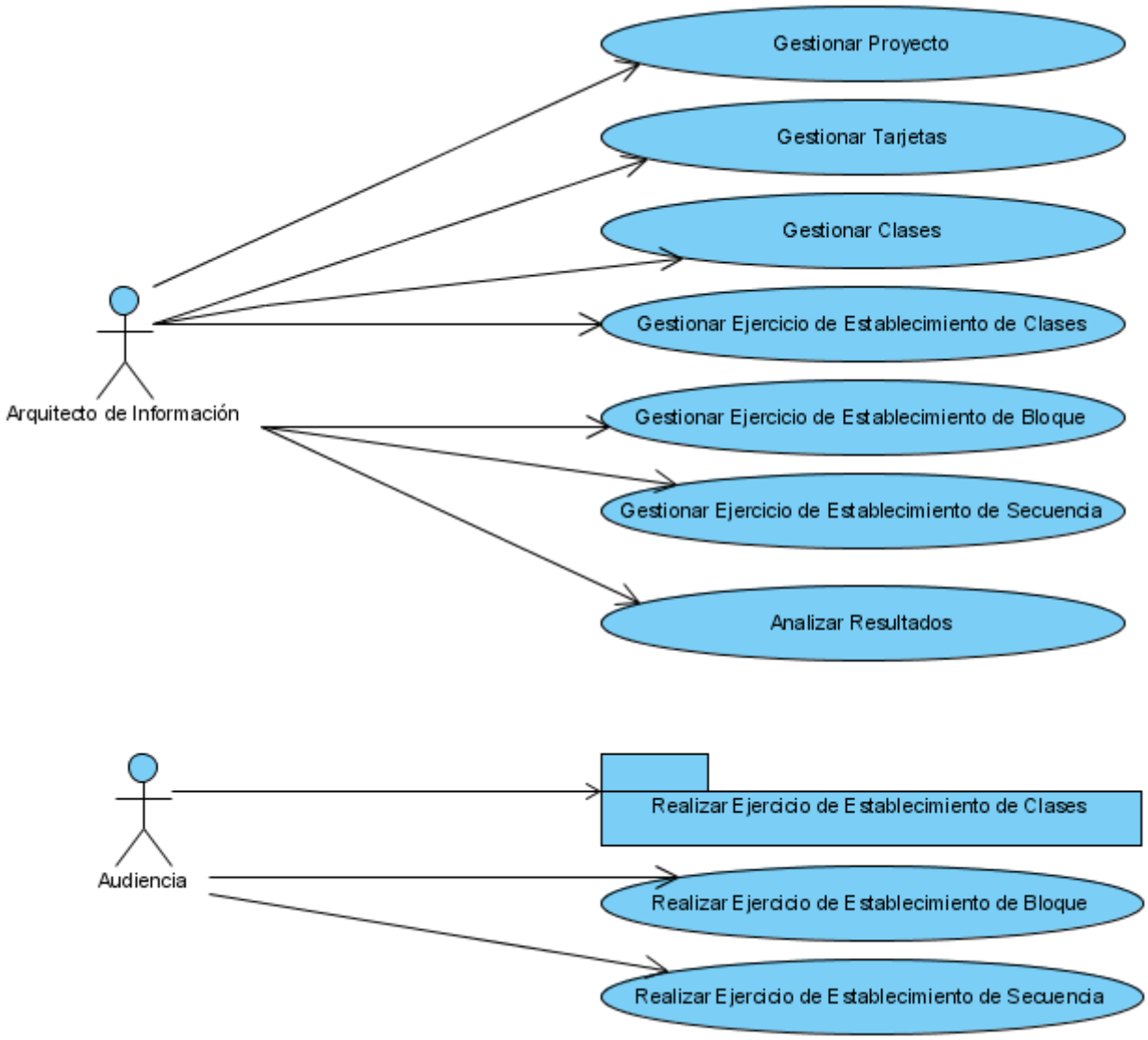
La aplicación debe contar con una ayuda, así la audiencia tendrá conocimiento de las funcionalidades del mismo y hacer un mejor uso de este.

## 2.7 Modelado del Sistema

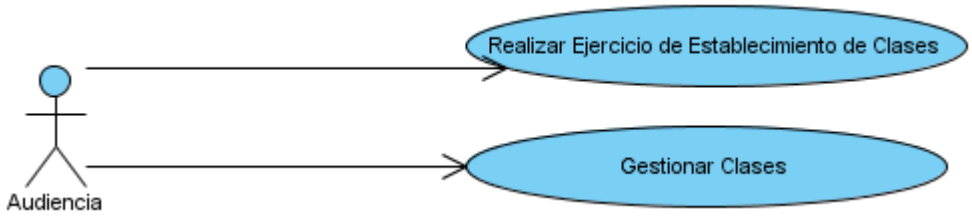
### 2.7.1 Actores del Sistema

| Actor | Descripción |
|---|---|
| Arquitecto de Información | Es el encargado de preparar todos los tipos de ejercicios, poner a disposición de la audiencia los mismos, recoger los resultados del ejercicio y mostrar un informe de estos. |
| Audiencia | Es el encargado de realizar cada uno de los ejercicios propuestos por el arquitecto. |

### 2.7.2 Modelo de Casos de Uso del Sistema




### Paquete de Casos de Uso Realizar Ejercicio de Establecimiento de Clases



### 2.7.3 Descripción de los Casos de Uso del Sistema

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Proyecto de Card Sorting.** |
| **Actores:** | **Arquitecto de la información.** |
| **Resumen:** | **El Caso de Uso comienza cuando el Arquitecto de Información prepara un proyecto de Card Sorting, este puede modificar uno existente, crear uno nuevo o eliminarlo, al igual que confecciona las orientaciones para los ejercicios y los guarda en un fichero de Ejercicios Preparados, finalizando así el caso de uso.** |
| **Precondiciones:** | **El diseñador debe tener claro todos los datos para la realización del ejercicio.** |
| **Referencias** | **RF 1, RF 22, RF 26.** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Sección "Modificar Proyecto"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. El Arquitecto accede a modificar un determinado dato del Proyecto de Card Sorting.** | |
| **2. El Arquitecto carga el Proyecto a modificar.** | **2.1 El sistema muestra todos los campos del proyecto buscado permitiendo que sean modificados.** |
| **3. El Arquitecto modifica los campos necesarios y selecciona actualizar.** | **3.1 El sistema verifica que todos los campos obligatorios estén llenos.** |
| | **3.2 El sistema le solicita que actualice el Proyecto en el fichero de Ejercicios Preparados.** |
| **4. El Arquitecto guarda el Proyecto actualizado.** | **4.1 El sistema muestra un mensaje informando que el Proyecto ha sido modificado satisfactoriamente.** |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje para que se llenen todos los campos obligatorios.** |

| Sección "Crear Proyecto" | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El arquitecto accede a crear un nuevo proyecto de Card Sorting. | 1.1 El Sistema muestra una opción para poner nombre y tipo del proyecto. |
| 2. El arquitecto inserta los datos del nuevo proyecto y selecciona la opción crear. | 2.1 El sistema verifica que todos los campos obligatorios estén llenos. |
| | 2.2 El sistema muestra un mensaje informando que se ha guardado el nuevo proyecto en el fichero de Ejercicios Preparados. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 2 | Se emite un mensaje para que se llenen todos los campos obligatorios. |
| **Sección "Eliminar Proyecto"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El arquitecto accede a eliminar un proyecto de Card Sorting. | 1.1 Muestra el proyecto a cargado. |
| 3. El Arquitecto elije la opción eliminar proyecto. | 3.1 Es eliminado del archivo el proyecto seleccionado. |
| **Poscondiciones** | Queda guardado el Proyecto de Card Sorting en su conjunto. |

| Caso de Uso: | Gestionar Tarjetas |
|---|---|
| Actores: | Arquitecto de la información. |
| Resumen: | El caso de uso comienza cuando el arquitecto decide crear, modificar, visualizar o eliminar tarjetas, ya sea para un proyecto o de forma independiente. |
| Precondiciones: | Tienen que estar identificados los nombres de las tarjetas. |
| Referencias | RF 3, RF 4, RF5, RF 25 |
| Prioridad | Crítico |
| **Flujo Normal de Eventos** | |
| **Sección "Crear Tarjeta"** | |

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. Accede para crear, una o varias tarjetas. | 1.1 Muestra un set de tarjetas. |
| 2. Selecciona la o las tarjetas a crear así como sus nombres y descripciones. | 2.2 El sistema verifica que todos los campos obligatorios estén llenos. |
| | 2.3 Se crean la o las tarjetas con los datos introducidos. |
| 3. Elije la opción guardar tarjetas. | 3.1 Guarda las tarjetas creadas. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 2 | Se emite un mensaje para que se llenen todos los campos obligatorios. |
| **Sección "Modificar Tarjeta"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1 Accede para modificar una o varias tarjetas. | |
| 2. Carga el set de tarjetas a modificar o el proyecto de Card Sorting. | 2.1 Se muestra el set de tarjetas seleccionado. |
| 3. Se modifican los datos seleccionados. | 3.1 El sistema verifica que todos los campos obligatorios estén llenos. |
| | 3.2 Se actualizan los cambios. |
| 4. Se guardan los cambios realizados. | 4.1 Quedan actualizados los cambios realizados en la o las tarjetas seleccionadas. |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 3 | Se emite un mensaje para que se llenen todos los campos obligatorios. |
| **Sección "Eliminar Tarjeta"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Accede para eliminar una o varias tarjetas. | |
| 2. Carga el set de tarjetas a eliminar o el proyecto de Card Sorting. | 2.1 Se muestra el set de tarjetas seleccionado. |
| 3. Se eliminan la o las tarjetas previstas previamente. | 3.1 Se quitan del set las tarjetas eliminadas. |

| | |
|---|---|
| **4. Se guardan los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en los set o proyectos de Card Sorting.** |
| **Poscondiciones** | **Queda guardada la tarjeta en su totalidad.** |

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Clases** |
| **Actores:** | **Arquitecto de la información.** |
| **Resumen:** | **El caso de uso comienza cuando el arquitecto decide crear, modificar, visualizar o eliminar clases, ya sea para un proyecto o de forma independiente.** |
| **Precondiciones:** | **Tienen que estar identificados los nombres de las clases.** |
| **Referencias** | **RF 2, RF 6, RF7, RF 8, RF 24** |
| **Prioridad** | **Crítico** |

| **Flujo Normal de Eventos** | |
|---|---|
| **Sección "Crear Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para crear, una o varias clases.** | **1.1 Muestra un set de clases.** |
| **2. Selecciona la o las clases a crear así como sus nombres y descripciones.** | **2.2 El sistema verifica que todos los campos obligatorios estén llenos.** |
| | **2.3 Se crean la o las clases con los datos introducidos.** |
| **3. Guarda las clases creadas.** | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **2** | **Se emite un mensaje para que se llenen todos los campos obligatorios.** |
| **Sección "Modificar Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1 Accede para modificar una o varias clases.** | |
| **2. Carga el set de clases a modificar o el proyecto de Card Sorting.** | **2.1 Se muestra el set de clases seleccionado.** |

| | |
|---|---|
| **3. Se modifican los datos seleccionados.** | **3.1 El sistema verifica que todos los campos obligatorios estén llenos.** |
| | **3.2 Se actualizan los cambios.** |
| **4. Se guardan los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en la o las tarjetas seleccionadas.** |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje para que se llenen todos los campos obligatorios.** |
| **Sección "Eliminar Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para eliminar una o varias clases.** | |
| **2. Carga el set de clases a eliminar o el proyecto de Card Sorting.** | **2.1 Se muestra el set de clases seleccionado.** |
| **3. Se eliminan la o las clases previstas previamente.** | **3.1 Se quitan del set las clases eliminadas.** |
| **4. Se guardan los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en los set o proyectos de Card Sorting.** |
| **Poscondiciones** | **Queda guardada la clase en su totalidad.** |

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Ejercicio de Establecimiento de Clases.** |
| **Actores:** | **Arquitecto de la información.** |
| **Resumen:** | **El Caso de Uso comienza cuando el Arquitecto de Información decida crear, modificar, visualizar o eliminar el Ejercicio de Establecimiento de Clases luego lo almacena en el fichero de Ejercicios Preparados, finalizando así el Caso de Uso.** |
| **Precondiciones:** | **Tienen que estar identificados los nombres de las tarjetas y si es cerrado los nombres de las clases.** |
| **Referencias** | **RF 1, RF 9, RF 10, RF 15, RF 25.** |

| Prioridad | Crítico |
|---|---|
| **Flujo Normal de Eventos** | |
| **Sección "Crear Ejercicio de Establecimiento de Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede a preparar las orientaciones y el ejercicio de establecimiento de clases para la Audiencia.** | **1.1 El sistema muestra un formulario para redactar las orientaciones** |
| **2. Redacta las orientaciones y selecciona guardar.** | **2.1 El sistema lo guarda y muestra una interfaz con las orientaciones redactadas.** |
| | **2.2 Muestra una interfaz donde permite seleccionar tarjetas y en caso de ser cerrado el ejercicio permite también seleccionar clases.** |
| **3. Selecciona las tarjetas y de caso de ser cerrado las clases que van a conformar el proyecto de Card Sorting.** | **3.1 El sistema verifica que todos los elementos estén seleccionados.** |
| | **3.2 Crea el Ejercicio de Establecimiento de Clases.** |
| **4. Selecciona la opción guardar.** | **4.1 Guarda el ejercicio.** |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio.** |
| **Sección "Modificar Ejercicio de Establecimiento de Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para modificar un ejercicio.** | |
| **2. Carga el ejercicio a modificar.** | **2.1 Muestra el ejercicio seleccionado.** |
| **3. Realiza los cambios previstos.** | **3.1 El sistema verifica que todos los elementos estén seleccionados.** |
| | **3.2 Se actualizan los cambios realizados.** |
| **4. Se guardan los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en la o las tarjetas seleccionadas.** |
| **Flujo Alterno** | |

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 3 | Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio. |
| **Sección "Eliminar Ejercicio de Establecimiento de Clases"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Accede para eliminar un ejercicio. | |
| 2. Carga el ejercicio a eliminar. | 2.1 Se muestra el ejercicio seleccionado. |
| 3. Se elimina el ejercicio previsto. | 3.1 Se actualizan los cambios realizados. |
| 4. Se guardan los cambios realizados. | 4.1 Queda eliminado el ejercicio o actualizado el proyecto de Card Sorting. |

| | |
|---|---|
| **Poscondiciones** | Queda confeccionado el ejercicio de establecimiento de clases. |
| **Caso de Uso:** | Gestionar Ejercicio de Establecimiento de Bloque. |
| **Actores:** | Arquitecto de la información. |
| **Resumen:** | El Caso de Uso comienza cuando el Arquitecto de Información decida crear, modificar, visualizar o eliminar el Ejercicio de Establecimiento de Bloque luego lo almacena en el fichero de Ejercicios Preparados, finalizando así el Caso de Uso. |
| **Precondiciones:** | Tienen que estar identificados los nombres de las clases. |
| **Referencias** | RF 1, RF 11, RF 12, RF 15, RF 25. |
| **Prioridad** | Crítico |

| **Flujo Normal de Eventos** | |
|---|---|
| **Sección "Crear Ejercicio de Establecimiento de Bloque"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Accede a preparar las orientaciones y el ejercicio de establecimiento de bloque para la Audiencia. | 1.1 El sistema muestra un formulario para redactar las orientaciones |
| 2. Redacta las orientaciones y selecciona guardar. | 2.1 El sistema lo guarda y muestra una interfaz con las orientaciones redactadas. |
| 3. Selecciona las clases y la plantilla que van a conformar el proyecto de Card Sorting. | 3.1 El sistema verifica que todos los elementos estén seleccionados. |
| | 3.2 Crea el Ejercicio de Establecimiento de Bloques. |

| | |
|---|---|
| **4. Selecciona la opción guardar.** | **4.1 Guarda el ejercicio.** |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio.** |
| **Sección "Modificar Ejercicio de Establecimiento de Bloque"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para modificar un ejercicio.** | |
| **2. Carga el ejercicio a modificar.** | **2.1 Muestra el ejercicio seleccionado.** |
| **3. Realiza los cambios previstos.** | **3.1 El sistema verifica que todos los elementos estén seleccionados.** |
| | **3.2 Se actualizan los cambios realizados.** |
| **4. Selecciona la opción guardar los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en la o las tarjetas seleccionadas.** |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio.** |
| **Sección "Eliminar Ejercicio de Establecimiento de Bloque"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para eliminar un ejercicio.** | |
| **2. Carga el ejercicio a eliminar.** | **2.1 Se muestra el ejercicio seleccionado.** |
| **3. Se elimina el ejercicio previsto.** | **3.1 Se actualizan los cambios realizados.** |
| **4. Se guardan los cambios realizados.** | **4.1 Queda eliminado el ejercicio o actualizado el proyecto de Card Sorting.** |
| **Poscondiciones** | **Queda confeccionado el ejercicio de establecimiento de bloque.** |
| **Caso de Uso:** | **Gestionar Ejercicio de Establecimiento de Secuencia.** |
| **Actores:** | **Arquitecto de la información.** |
| **Resumen:** | **El Caso de Uso comienza cuando el Arquitecto de Información decida crear, modificar, visualizar o eliminar el Ejercicio de Establecimiento de Secuencia luego lo almacena en el fichero de** |

| | |
|---|---|
| | **Ejercicios Preparados, finalizando así el caso de uso.** |
| **Precondiciones:** | **Tienen que estar identificados los nombres de las clases.** |
| **Referencias** | **RF 1, RF 13, RF 14, RF 15, RF 25.** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Sección "Crear Ejercicio de Establecimiento de Secuencia"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede a preparar las orientaciones y el ejercicio de establecimiento de bloque para la Audiencia.** | **1.1 El sistema muestra un formulario para redactar las orientaciones** |
| **2. Redacta las orientaciones y selecciona guardar.** | **2.1 El sistema lo guarda y muestra una interfaz con las orientaciones redactadas.** |
| | **2.2 Muestra una interfaz donde permite seleccionar clases y tarjetas para la conformación del ejercicio.** |
| **3. Selecciona las clases y tarjetas que van a conformar el proyecto de Card Sorting.** | **3.1 El sistema verifica que todos los elementos estén seleccionados.** |
| | **3.2 Crea el Ejercicio de Establecimiento de Secuencia.** |
| **4. Guarda el ejercicio.** | |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio.** |
| **Sección "Modificar Ejercicio de Establecimiento de Secuencia"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para modificar un ejercicio.** | |
| **2. Carga el ejercicio a modificar.** | **2.1 Muestra el ejercicio seleccionado.** |
| **3. Realiza los cambios previstos.** | **3.1 El sistema verifica que todos los elementos estén seleccionados.** |
| | **3.2 Se actualizan los cambios realizados.** |
| **4. Se guardan los cambios realizados.** | **4.1 Quedan actualizados los cambios realizados en la o las tarjetas seleccionadas.** |

| Flujo Alterno | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| **3** | **Se emite un mensaje informándole que se deben escoger todos los elementos necesarios para realizar el ejercicio.** |
| **Sección "Eliminar Ejercicio de Establecimiento de Secuencia"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede para eliminar un ejercicio.** | |
| **2. Carga el ejercicio a eliminar.** | **2.1 Se muestra el ejercicio seleccionado.** |
| **3. Se elimina el ejercicio previsto.** | **3.1 Se actualizan los cambios realizados.** |
| **4. Se guardan los cambios realizados.** | **4.1 Queda eliminado el ejercicio o actualizado el proyecto de Card Sorting.** |
| **Poscondiciones** | **Queda confeccionado el ejercicio de establecimiento de bloque.** |

| **Caso de Uso:** | **Analizar Resultados** |
|---|---|
| **Actores:** | **Arquitecto de la información.** |
| **Resumen:** | **El Caso de uso comienza cuando el Arquitecto de Información procesa los resultados de los ejercicios de Card Sorting realizados por la Audiencia, le aplica el algoritmo de clustering, obtiene un Informe y lo guarda en el fichero de Resultados Finales, finalizando así el caso de uso.** |
| **Precondiciones:** | **Deben estar guardados los ejercicios realizados por la Audiencia.** |
| **Referencias** | **RF 27, RF 28, RF 29 RF 30** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Accede a procesar los resultados de los ejercicios realizados por la Audiencia.** | |
| **2. Carga el ejercicio a procesar.** | **2.1 El sistema levanta el ejercicio seleccionado.** |
| **3. Indica procesar el ejercicio seleccionado.** | **3.1 El sistema procesa, mediante el clustering el resultado del ejercicio.** |

| | |
|---|---|
| | **3.2 Muestra una interfaz con los resultados procesados.** |
| **4. Guarda los resultados.** | |
| **5. Solicita Informe de los resultados procesados** | **5.1 El sistema genera un informe que contiene los resultados del procesamiento de las respuestas generadas por la Audiencia.** |
| | **5.2 El sistema muestra en una interfaz el informe de los resultados antes procesados dando la opción de imprimirlo.** |
| Poscondiciones | **Queda generado e impreso un informe sobre los resultados de Ejercicios de Card Sorting resueltos por la Audiencia.** |

| | |
|---|---|
| **Caso de Uso:** | **Realizar Ejercicio de Establecimiento de Bloque** |
| **Actores:** | **Audiencia.** |
| **Resumen:** | **El Caso de Uso comienza cuando la Audiencia se decide a resolver el Ejercicio de Establecimiento de Bloques, y finaliza guardando la solución en el fichero de resultados, finalizando así el caso de uso.** |
| **Precondiciones:** | **Se tiene que disponer del ejercicio de Card Sorting y orientaciones previamente preparados por el Arquitecto de Información.** |
| **Referencias** | **RF 15, RF 16, RF 17, RF 20** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. La Audiencia accede a realizar el Ejercicio.** | **1.1 El sistema muestra una interfaz con los ejercicios y las orientaciones correspondientes al Ejercicio de Card Sorting.** |
| **2. La audiencia selecciona el ejercicio** | **2.1 El sistema muestra en la misma** |

| | |
|---|---|
| **indicado.** | **interfaz el ejercicio seleccionado.** |
| **3. La Audiencia organiza las clases en los bloques situados en la pantalla y selecciona actualizar.** | **3.1 El sistema muestra una interfaz con la solución del ejercicio.** |
| **4. La Audiencia revisa la solución y selecciona guardar.** | **4.1 El sistema guarda la solución en un fichero de Ejercicios Realizados, finalizando así el Caso de Uso.** |
| Poscondiciones | **Queda resuelto el Ejercicio de Establecimiento de Bloques, listo para procesar.** |

| | |
|---|---|
| **Caso de Uso:** | **Realizar Ejercicio de Establecimiento de Secuencia.** |
| **Actores:** | **Audiencia.** |
| **Resumen:** | **El caso de Uso comienza cuando la Audiencia se decide a resolver el ejercicio de Establecimiento de Secuencia, los guarda en el fichero de Ejercicios Realizados, finalizando así el caso de uso.** |
| **Precondiciones:** | **Se tiene que disponer del ejercicio de Card Sorting y orientaciones previamente preparados por el Arquitecto de Información.** |
| **Referencias** | **RF 15, RF 21, RF 22, RF 23, RF 24** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. La Audiencia accede a realizar el Ejercicio.** | **1.1 El sistema muestra una interfaz con los ejercicios y las orientaciones correspondientes al Ejercicio de Card Sorting.** |
| **2. La audiencia selecciona el ejercicio indicado.** | **2.1 El sistema muestra en la misma interfaz el ejercicio seleccionado.** |
| | **2.2 El sistema permite ordenar las tarjetas dentro de las clases.** |
| **3. La Audiencia le da un orden según su criterio a las tarjetas dentro de las clases dadas.** | **3.1 El sistema muestra una interfaz con la solución del ejercicio.** |

| | |
|---|---|
| **4. La Audiencia revisa la solución y selecciona guardar.** | **4.1 El sistema guarda la solución en un fichero de Ejercicios Realizados, finalizando así el Caso de Uso.** |
| Poscondiciones | **Queda resuelto el Ejercicio de Establecimiento de Secuencia, listo para procesar.** |

| | |
|---|---|
| **Caso de Uso:** | **Realizar Ejercicio de Establecimiento de Clases.** |
| **Actores:** | **Audiencia.** |
| **Resumen:** | **El Caso de Uso comienza cuando la Audiencia se decide a resolver el Ejercicio de Establecimiento de Clases, este puede hacerlo por Establecimiento de Clases Abierto, Cerrado o Mixto, luego guarda el ejercicio en el fichero de Ejercicios Realizados, terminando así el caso de uso.** |
| **Precondiciones:** | **Se tiene que disponer del ejercicio de Card Sorting y orientaciones previamente preparados por el Arquitecto de Información.** |
| **Referencias** | **RF 6, RF 7, RF 8, RF 9 RF 10, RF 15, RF 16, RF 17, RF 18, RF 19** |
| **Prioridad** | **Crítico** |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **2. La Audiencia accede a realizar el Ejercicio.** | **1.1 El sistema muestra una interfaz con los ejercicios y las orientaciones correspondientes al Ejercicio de Card Sorting.** |
| **2. La audiencia selecciona el ejercicio. indicado.** | **2.1 El sistema muestra en la misma interfaz el ejercicio seleccionado.** |
| | **2.2 En caso de seleccionar Resolver un Ejercicio de Establecimiento de Clases Abierto ir a la sección "Resolver Ejercicio de Establecimiento de Clases Abierto".** |
| | **2.3 En caso de seleccionar Resolver un Ejercicio de Establecimiento de Clases Cerrado ir a la sección "Resolver Ejercicio** |

| | |
|---|---|
| | **de Establecimiento de Clases Cerrado".** |
| **Sección "Resolver Ejercicio de Establecimiento de Clases Abierto".** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.1 Brinda la posibilidad de crear más clases y tarjetas con conceptos de la temática del sitio a diseñar.** |
| **2. Gestiona nuevas clases y tarjetas según entienda la audiencia.** | **2.1 Muestra las clases y tarjetas creadas junto con las tarjetas ya existentes.** |
| **3. Da solución al ejercicio agrupando las tarjetas en clases.** | **3.1 Muestra una vista previa de cómo quedará resuelto el ejercicio.** |
| **4. Solicita guardar el ejercicio.** | **4.1 El sistema informa que se guardo el ejercicio en el fichero de Ejercicios Realizados, finalizando así el Caso de Uso.** |
| **Sección "Resolver Ejercicio de Establecimiento de Clases Cerrado".** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1. Da solución al ejercicio agrupando las tarjetas en clases.** | **Muestra una vista previa de cómo quedará resuelto el ejercicio.** |
| **4. Solicita guardar el ejercicio.** | **4.1 El sistema informa que se guardo el ejercicio en el fichero de Ejercicios Realizados, finalizando así el Caso de Uso.** |
| Poscondiciones | **Queda resuelto el Ejercicio de Establecimiento de Clases, listo para procesar.** |

**Conclusiones**

En este capítulo se realizó la modelación del negocio, dando a conocer el funcionamiento de la realización de la técnica de Card Sorting. Se describieron los procesos de negocio que tienen lugar en dicha técnica. Se identificaron los actores y trabajadores del negocio, así como las reglas a tener en cuenta durante todo el proceso, los casos de uso del negocio y la descripción de los mismos.

Se definieron los requisitos funcionales y se obtuvo un listado de requerimientos no funcionales a tener en cuenta para el desarrollo de la aplicación. Así como, los casos de uso del sistema que

pueden agruparse lógicamente, mostrándose los diagramas de casos de uso. Además, se describieron los casos de uso, así como los actores del sistema, dando de esta forma una vista global de cómo está concebido y cómo funcionará el sistema.

# Capitulo 3 Análisis y Diseño del Sistema.

En el presente capítulo se abordan los aspectos relacionados con el flujo de trabajo de Análisis y Diseño, este es el tercer flujo planteado por la metodología RUP. Es uno de los mecanismos más importantes dentro del proceso unificado del desarrollo de un software. Se incluyen los diagramas de clases del diseño de los casos de uso del sistema más significativos y los diagramas de interacción de los casos de uso más críticos así como los subsistemas de diseño.

## 3.1 Patrones de Diseño

GRASP: Patrones para Asignar Responsabilidades

Un sistema orientado a objetos se compone de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones requeridas. Los diagramas de interacción describen gráficamente estas operaciones, a partir de los objetos en interacción, que se responsabilizan de una actividad determinada.

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP se codifican algunos de los principios, que se aplican al preparar los diagramas de interacción.

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software) van formando un amplio repertorio de principios generales y de expresiones que los guían al crear software. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. A continuación se describen los patrones básicos de asignación de responsabilidades. [24]

**Patrón Experto**

Problema:

¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades.

Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Solución:

Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Beneficios de su uso:

Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión. [25]

**Patrón Creador**

Problema:

¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos.

En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.

El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.

El propósito fundamental de este patrón es encontrar un creador que se conecta con el objeto producido en cualquier evento.

Al escogerlo como creador, se da soporte al bajo acoplamiento.

Solución:

Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o
- B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A. [26]

**Patrón Bajo Acoplamiento**

Problema:

¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

Acoplamiento bajo significa que una clase no depende de muchas clases.

Acoplamiento alto significa que una clase recurre a muchas otras clases. Esto presenta los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Difíciles de entender cuando están aisladas.
- Difíciles de reutilizar puesto que dependen de otras clases.

Solución:

Asignar una responsabilidad para mantener bajo acoplamiento.

El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño. [27]

**Patrón Alta Cohesión**

Problema:

¿Cómo mantener la complejidad dentro de límites manejables?

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. Esto presenta los siguientes problemas:

- Son difíciles de comprender
- Difíciles de reutilizar
- Difíciles de conservar
- Las afectan constantemente los cambios.

Solución:

Asignar una responsabilidad de modo que la cohesión siga siendo alta. [28]

**Patrón Controlador**

Problema:

¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

Solución:

Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

• El “sistema” global (controlador de fachada).

• La empresa u organización global (controlador de fachada).

• Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).

• Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador<NombreCasodeUso>” (controlador de casos de uso).

Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad. [29]

**3.2 Arquitectura en tres capas**

**La capa de presentación o interfaz de usuario**

En este caso, está formada por los formularios y los controles que se encuentran en los formularios. Capa con la que interactúa el usuario.

**La capa de negocio**

Esta capa está formada por las entidades empresariales, que representan objetos que van a ser manejados o consumidos por toda la aplicación. En este caso, están representados por las clases que se crean relacionadas con el procedimiento.

**La capa de acceso a datos**

Contiene clases que interactúan con la fuente de almacenamiento de datos, estas clases altamente especializadas se encuentran en la arquitectura y permiten, realizar todas las operaciones con la fuente de almacenamiento de datos de forma transparente para la capa de negocio.

### 3.3 Análisis

Durante el análisis se estudian los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura.

**Modelo de Análisis**

El modelo de análisis está escrito en el lenguaje del desarrollador, es la vista interna del sistema. Estructura los requisitos de un modo que facilita su comprensión, su preparación, su modificación y en general, su mantenimiento. Puede considerarse como una primera aproximación al modelo de diseño.

**Clases del Análisis**

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Las clases del análisis siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad.

**Clases de Interfaz:** Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores (es decir, usuarios y sistemas externos).

**Clases de Entidad:** Las clases de entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente. Modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real.

**Clases de Control:** Las clases de control representan coordinación, secuencia, transacciones, y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto.

#### 3.3.1 Diagrama de Clases del Análisis

El diagrama de clases del análisis representa básicamente los conceptos de un dominio del problema. Representa los procesos, no de la implementación automatizada de los mismos.

**Diagrama de Clases del Análisis Gestionar Proyecto de Card Sorting**

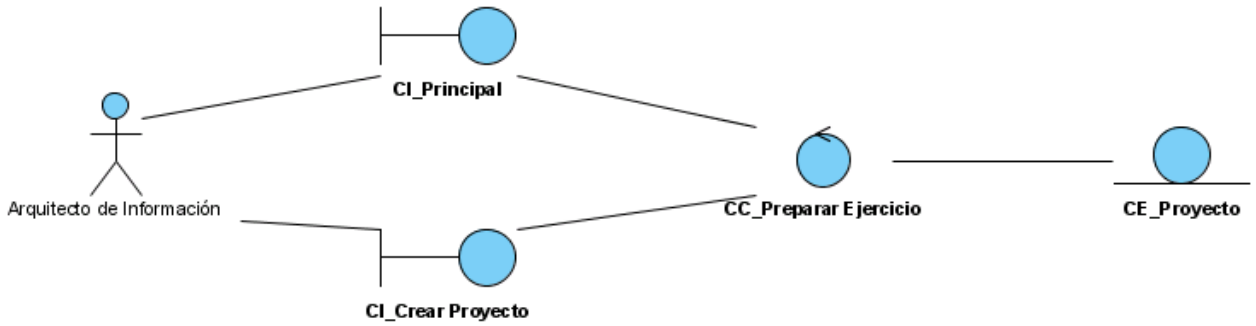



**Diagrama de Clases del Análisis Gestionar Tarjetas**

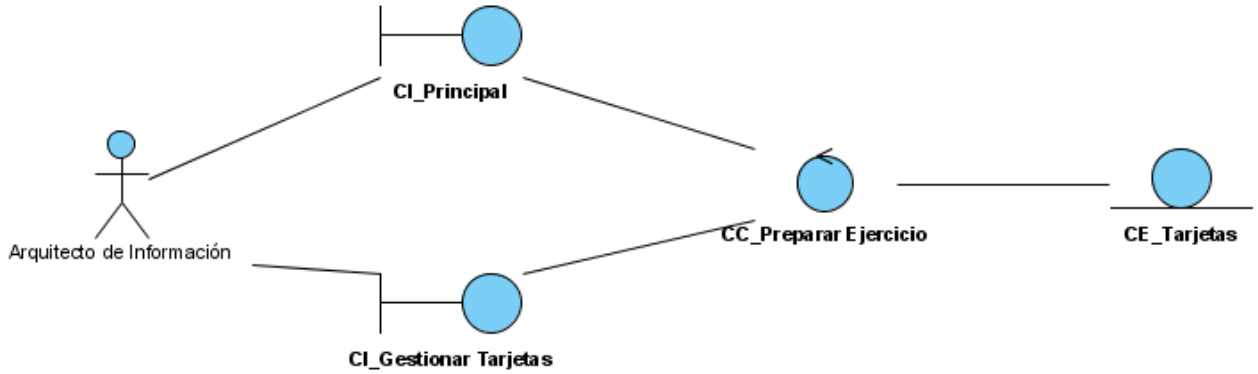



**Diagrama de Clases del Análisis Gestionar Clases**

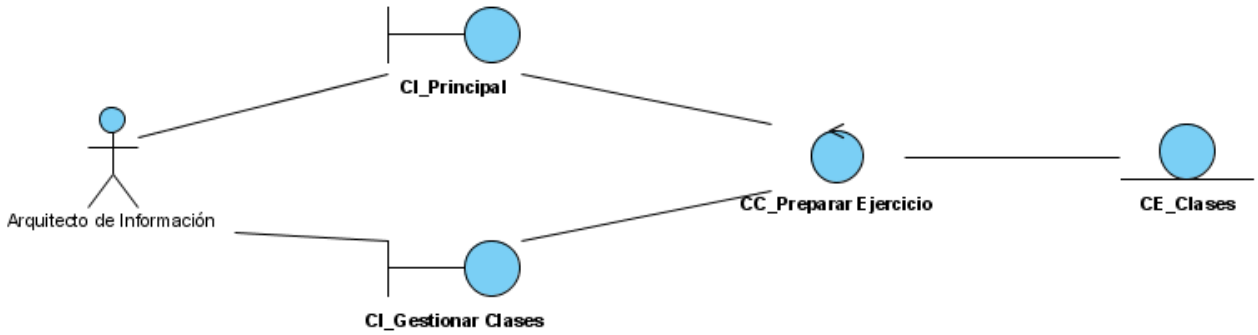


**Diagrama de Clases del Análisis Gestionar Ejercicio de Establecimiento de Clases**

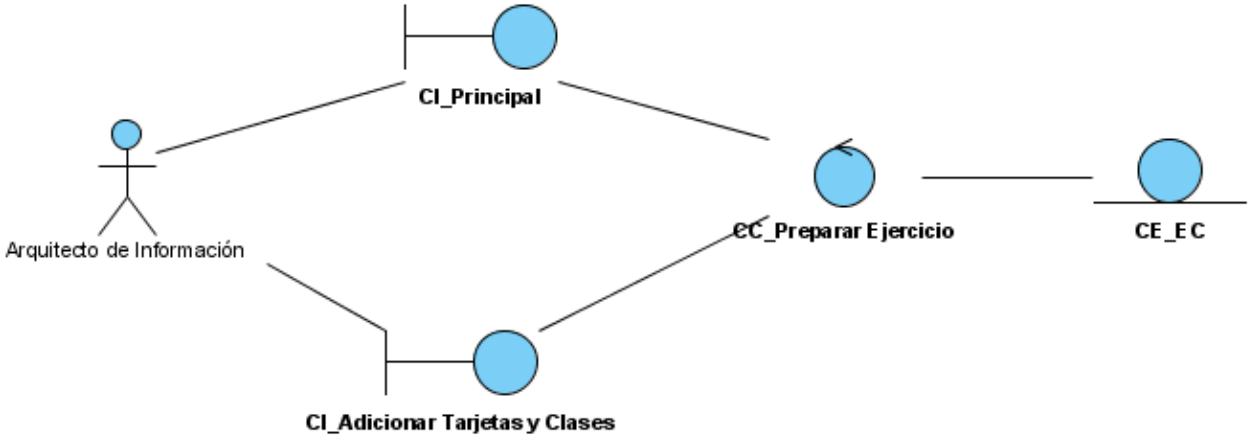



**Diagrama de Clases del Análisis Gestionar Establecimiento de Secuencia**

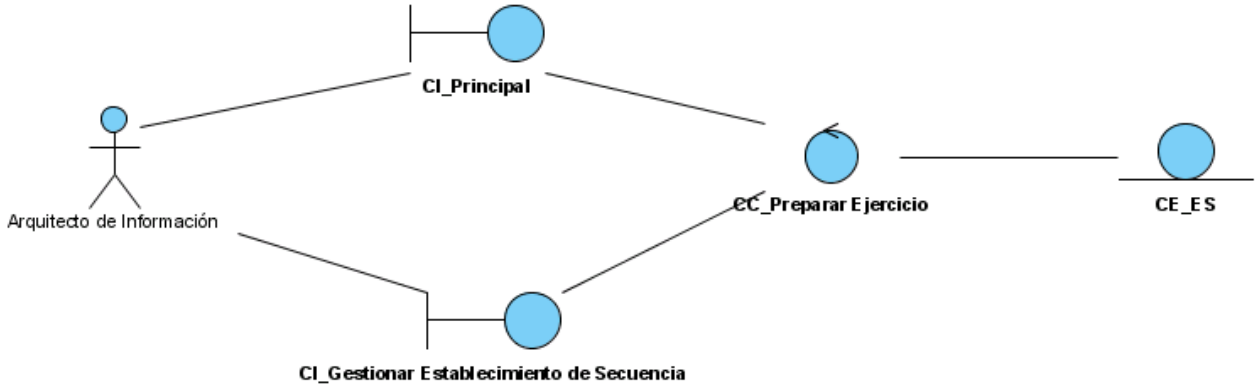



**Diagrama de Clases del Análisis Gestionar Ejercicio de Establecimiento de Bloque**

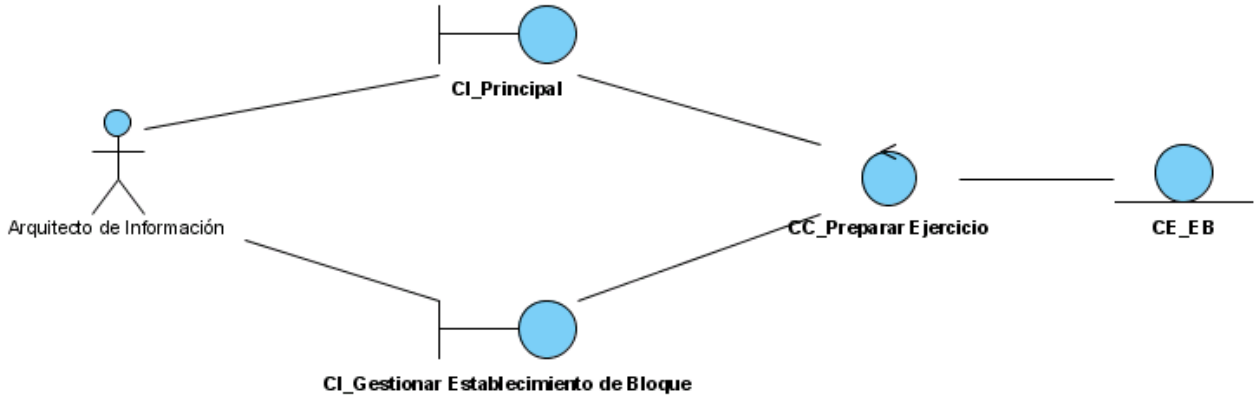


**Diagrama de Clases del Análisis Resolver Ejercicio de establecimiento de Secuencia**

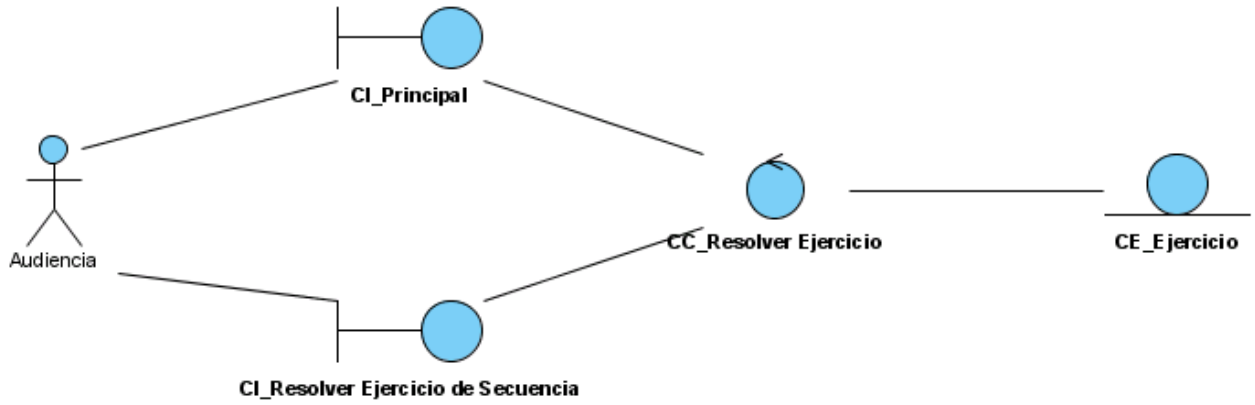



**Diagrama de Clases del Análisis Resolver Establecimiento de Bloque**

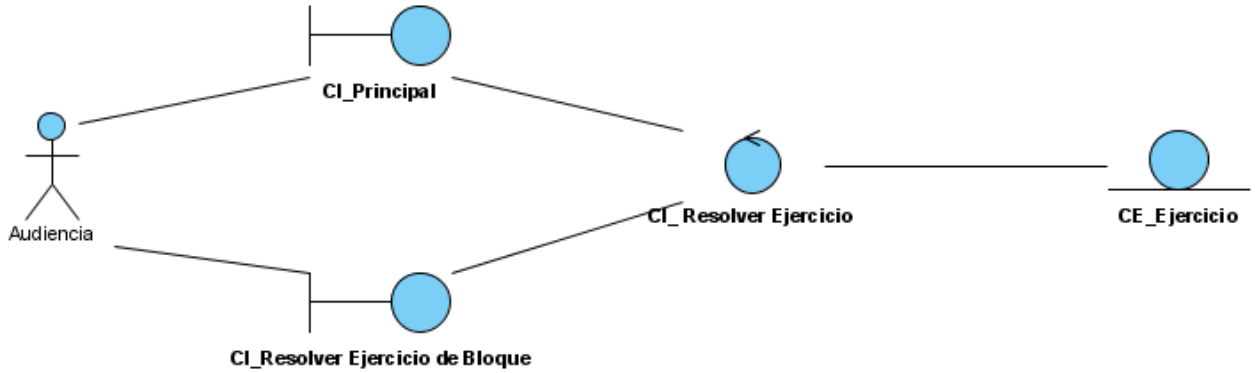



**Diagrama de Clases del Análisis Resolver Establecimiento de Clases**

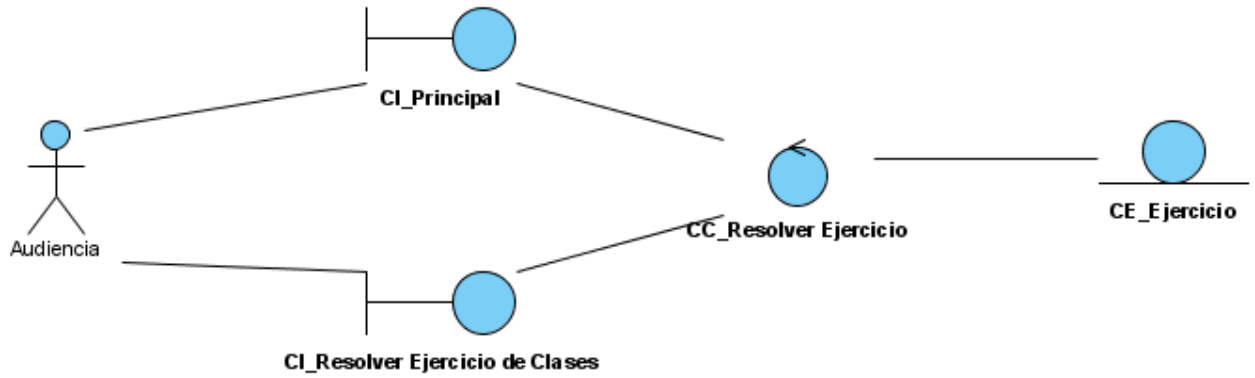
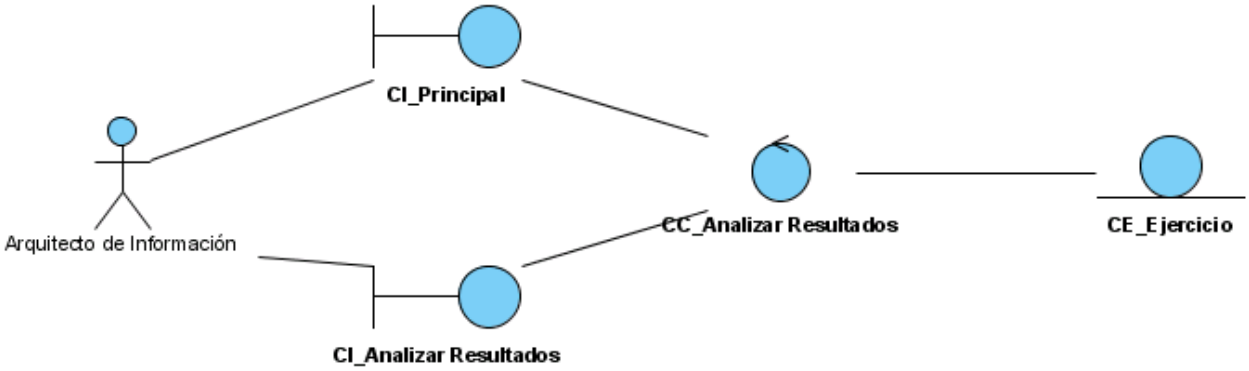


**Diagrama de Clases del Análisis Analizar Resultados**




### 3.4 Diseño

El diseño debe definir una solución que satisfaga de modo efectivo y eficiente los requisitos especificados en el análisis, incorpora nuevos artefactos (nuevas clases, nuevos atributos y operaciones para las clases). Es un modelo físico que crea una entrada apropiada y un punto de partida para la implementación.

#### 3.4.1 Diagrama de Clases del Diseño

**Diagrama de Clases del Diseño Paquete de Casos de Usos Preparar Ejercicio**

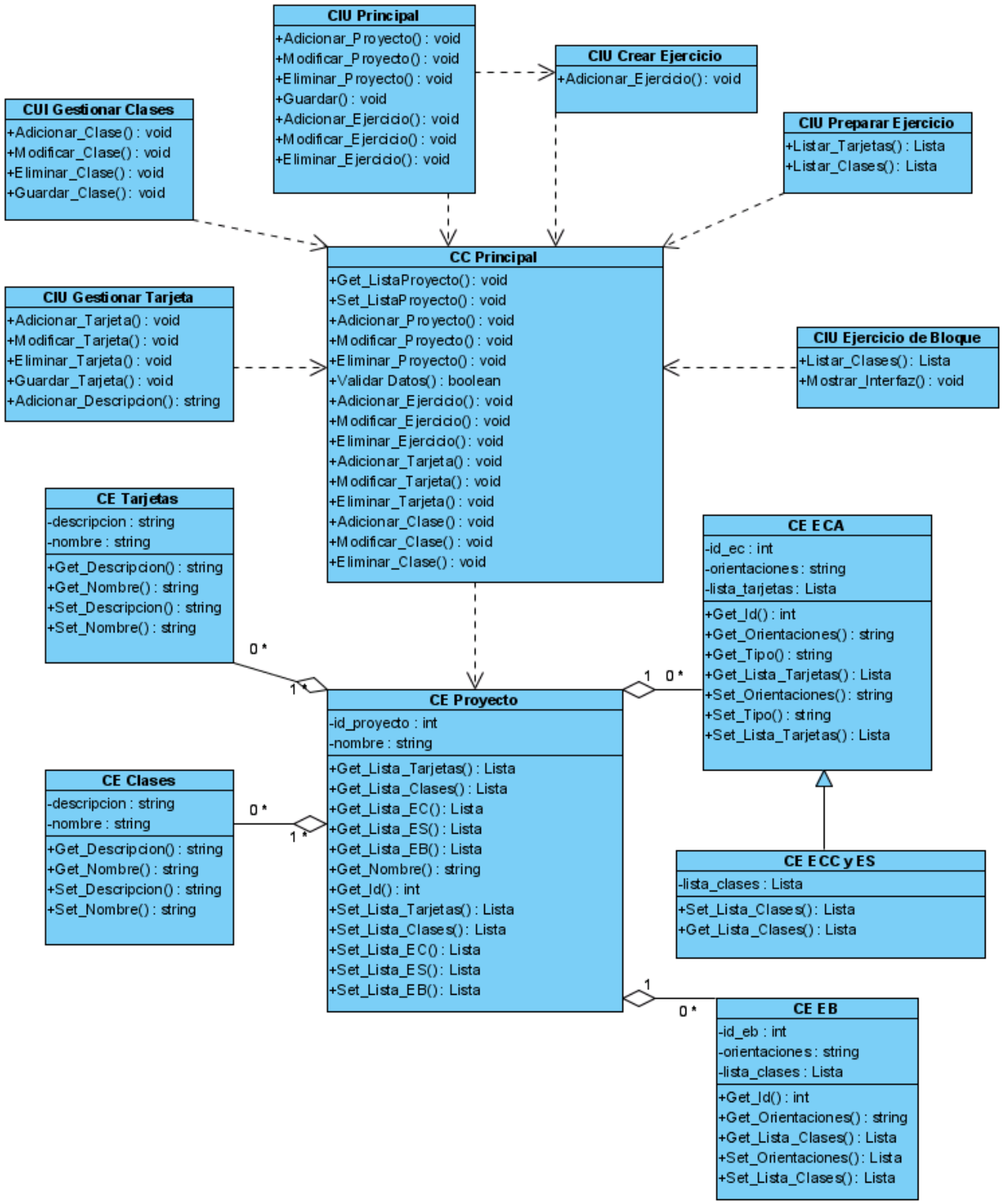


**Diagrama de Clases del Diseño Paquete de Casos de Usos Resolver Ejercicio**

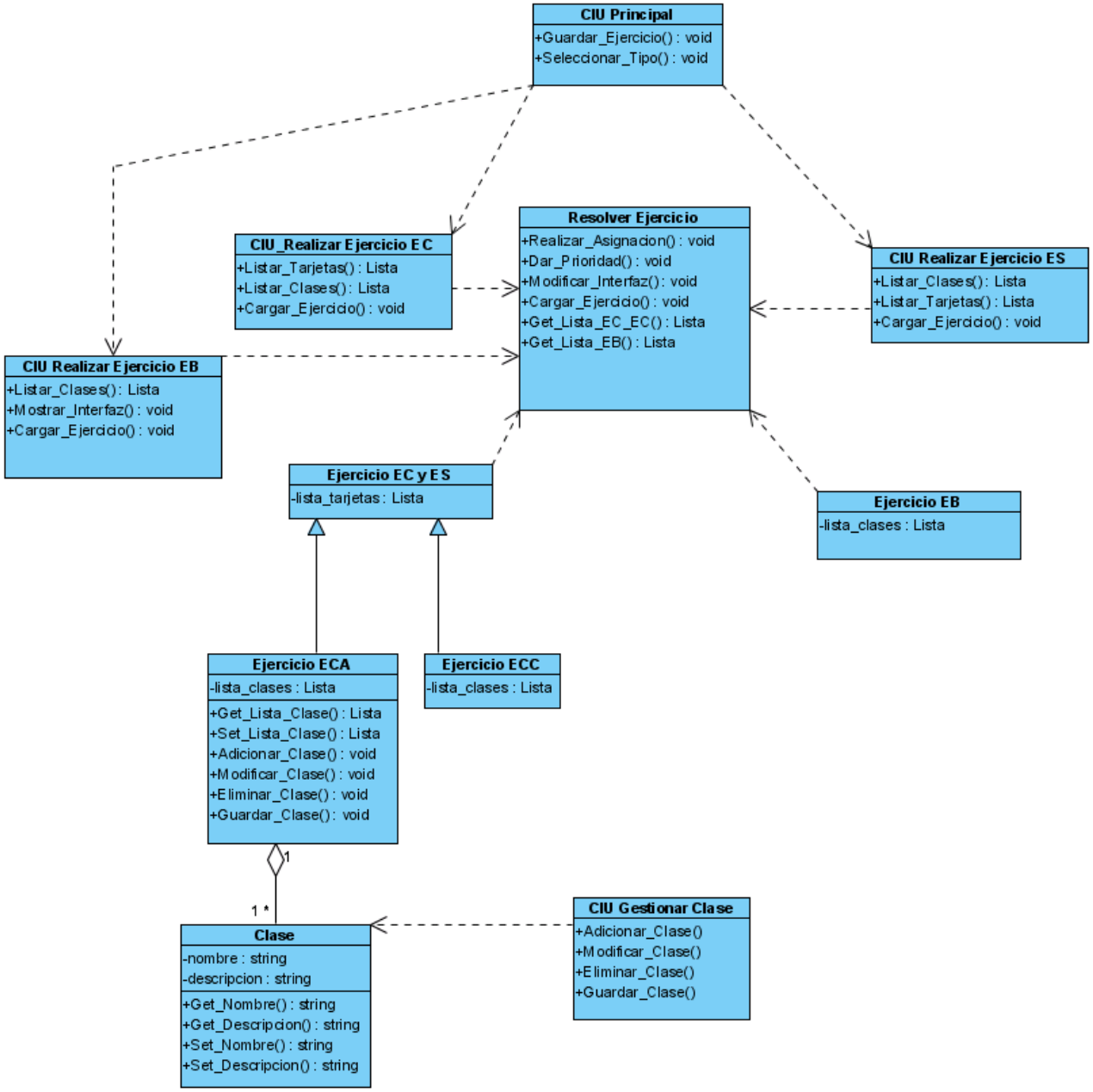
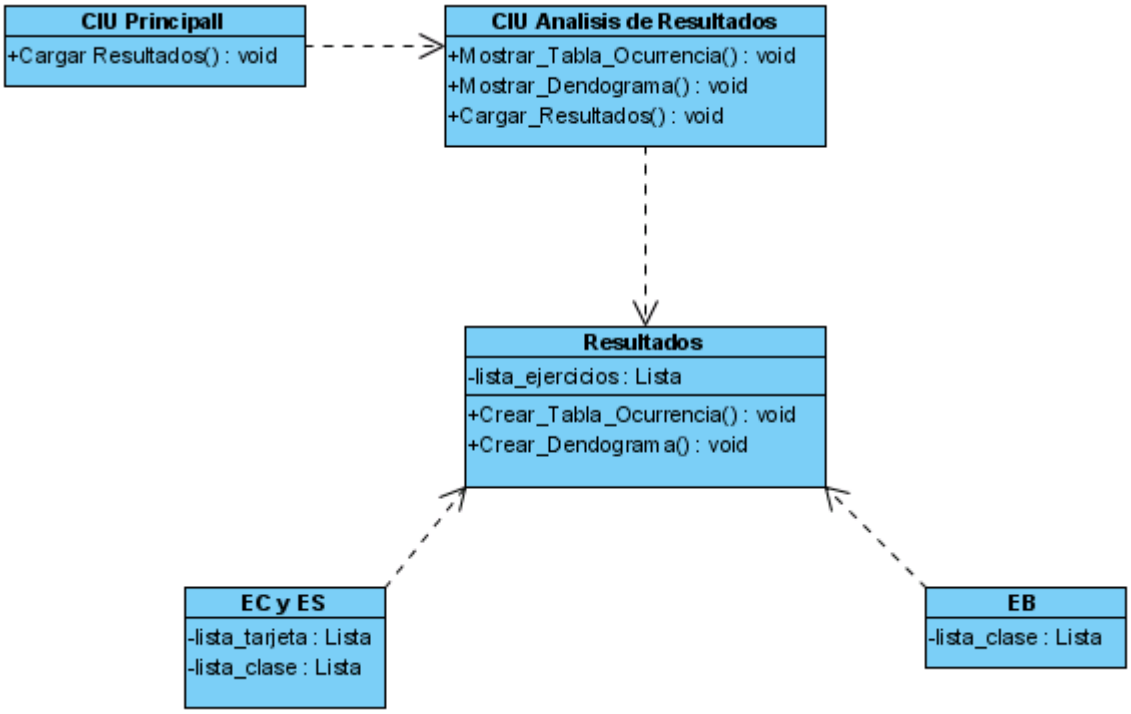


**Diagrama de Clases del Diseño Caso de Uso Analizar Resultados**




### 3.4.2 Diagramas de Interacción

**Diagrama de Secuencia Caso de Uso Gestionar Proyecto: Escenario Adicionar**

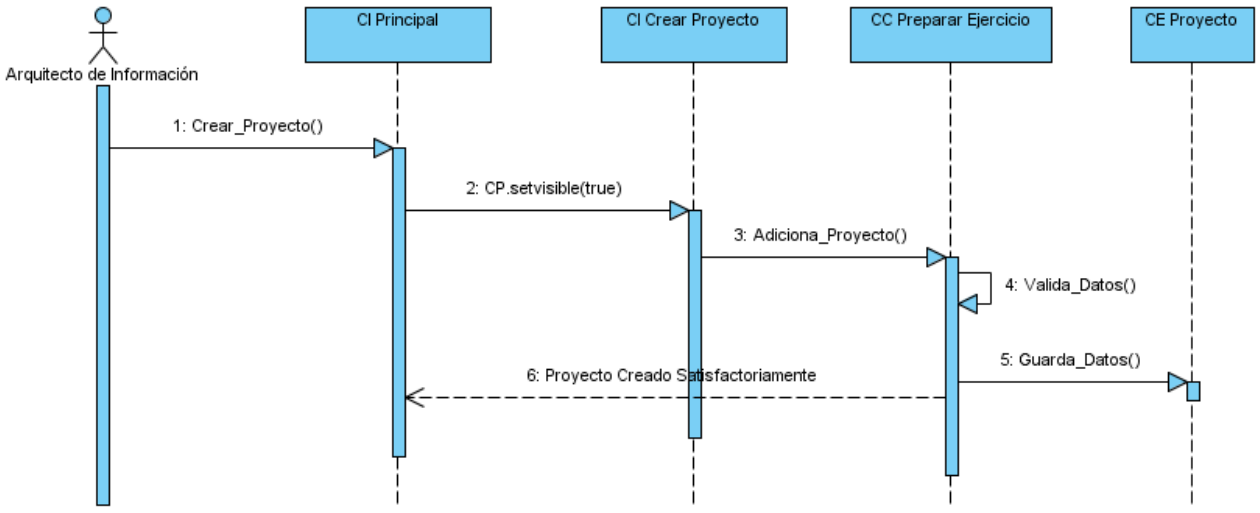


**Diagrama de Secuencia Caso de Uso Gestionar Proyecto: Escenario Modificar**

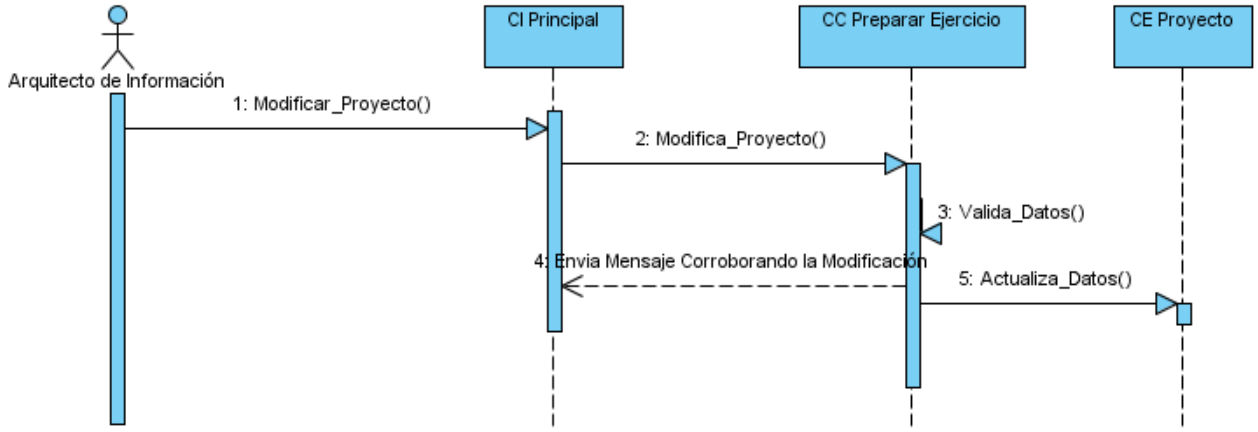



**Diagrama de Secuencia Caso de Uso Gestionar Proyecto: Escenario Eliminar**

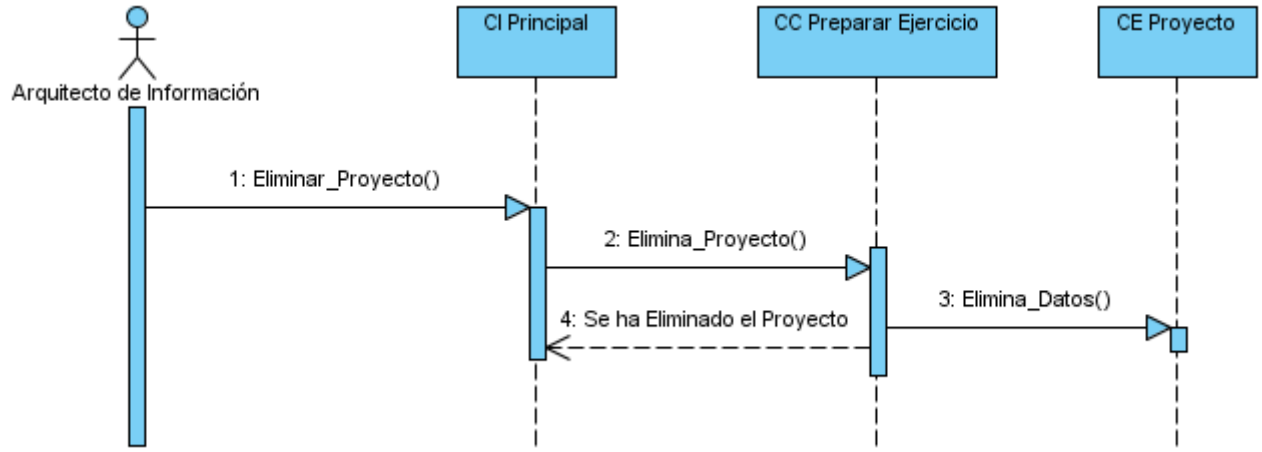



**Diagrama de Secuencia Caso de Uso Gestionar Clase: Escenario Adicionar**

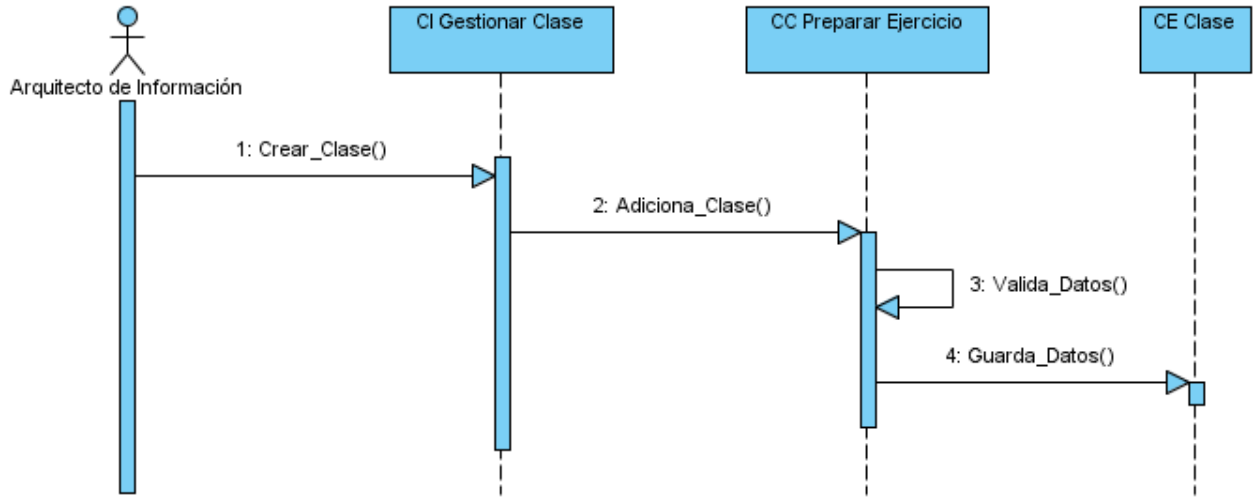


**Diagrama de Secuencia Caso de Uso Gestionar Clase: Escenario Modificar**

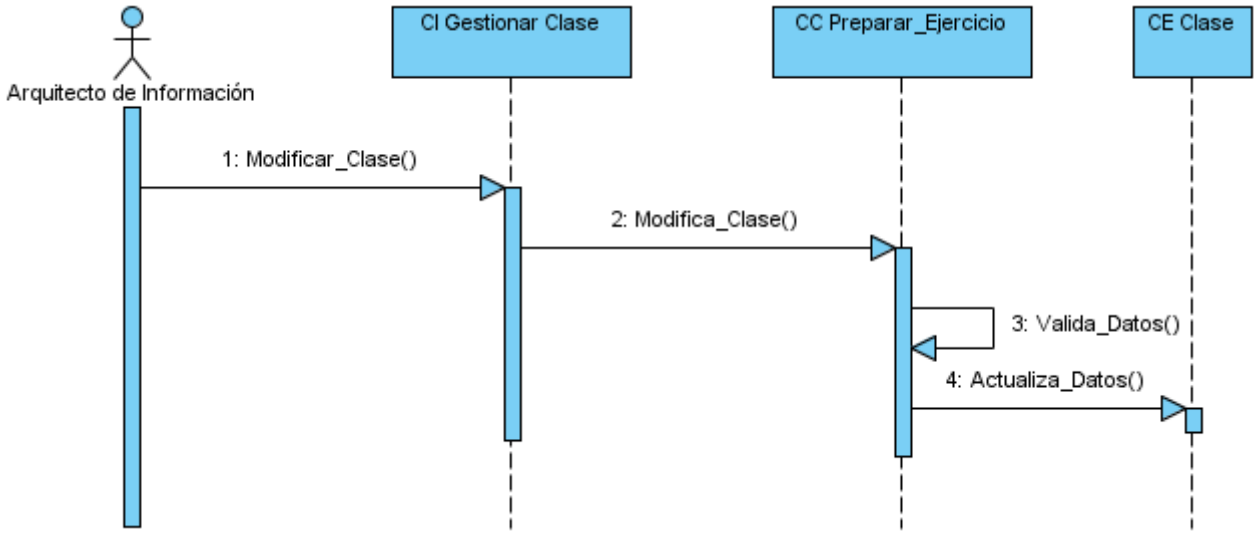



**Diagrama de Secuencia Caso de Uso Gestionar Clase: Escenario Eliminar**

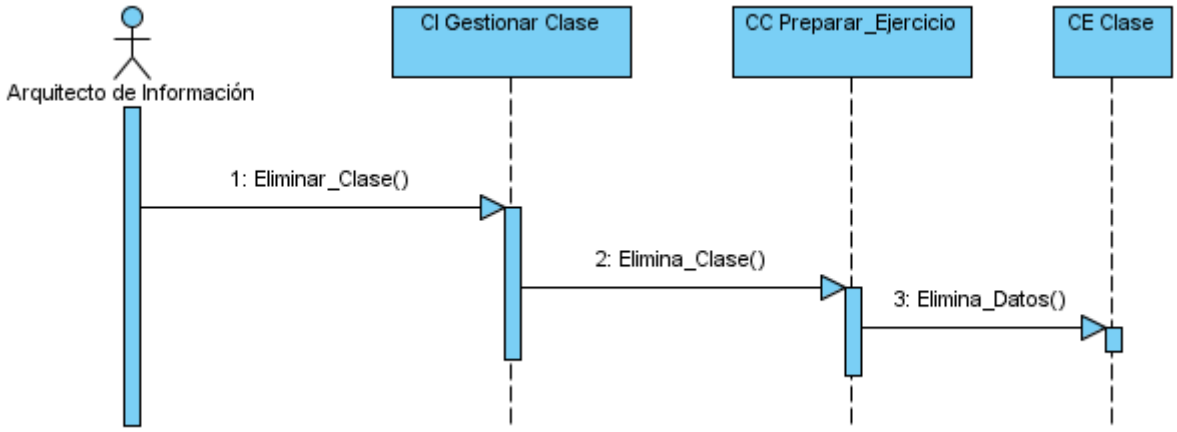



**Diagrama de Secuencia Caso de Uso Gestionar Tarjeta: Escenario Adicionar**

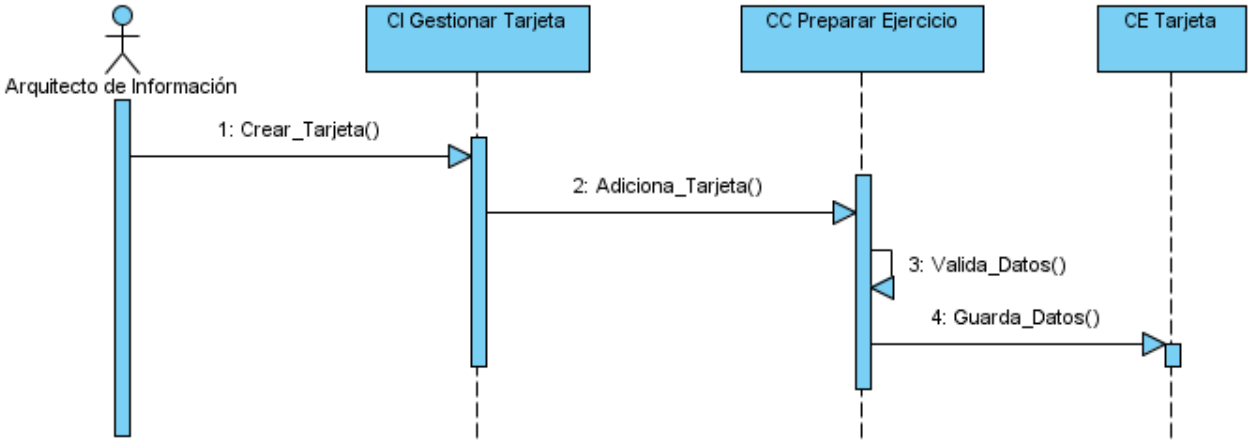


**Diagrama de Secuencia Caso de Uso Gestionar Tarjeta: Escenario Modificar**

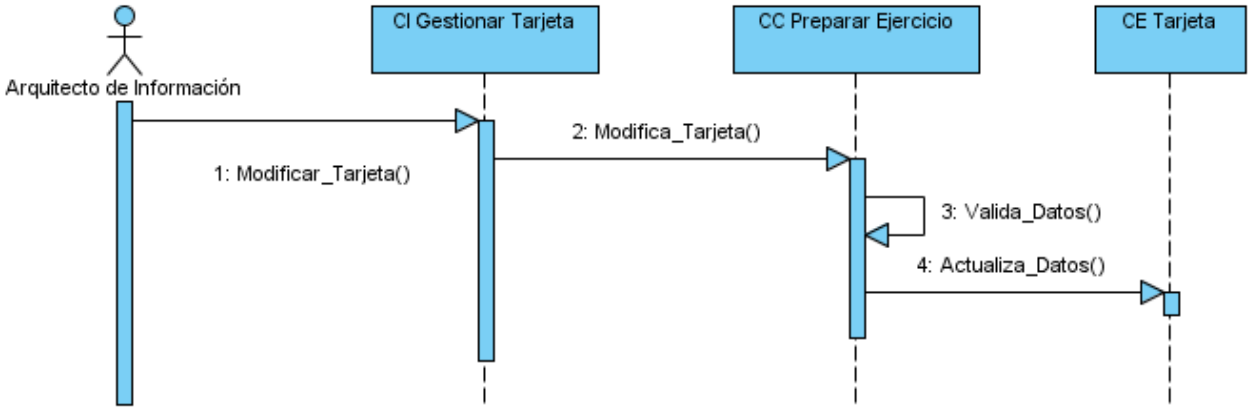



**Diagrama de Secuencia Caso de Uso Gestionar Tarjeta: Escenario Eliminar**

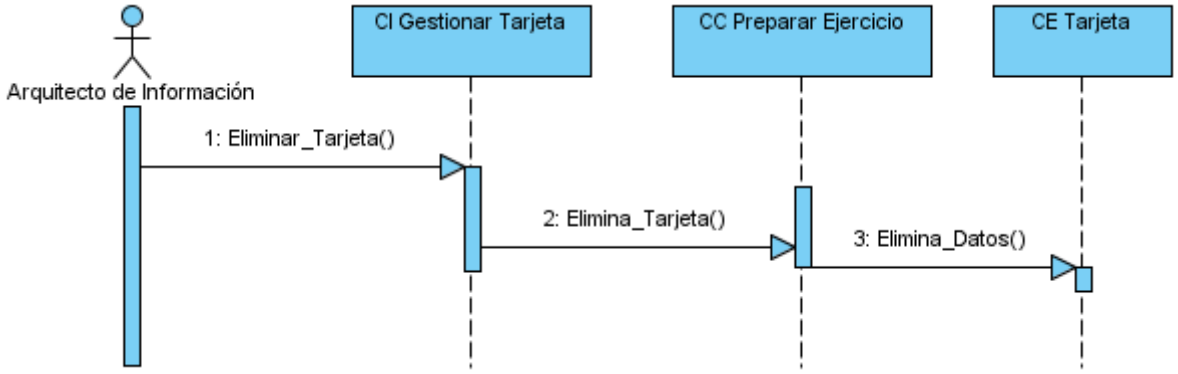



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Bloque: Escenario Adicionar**

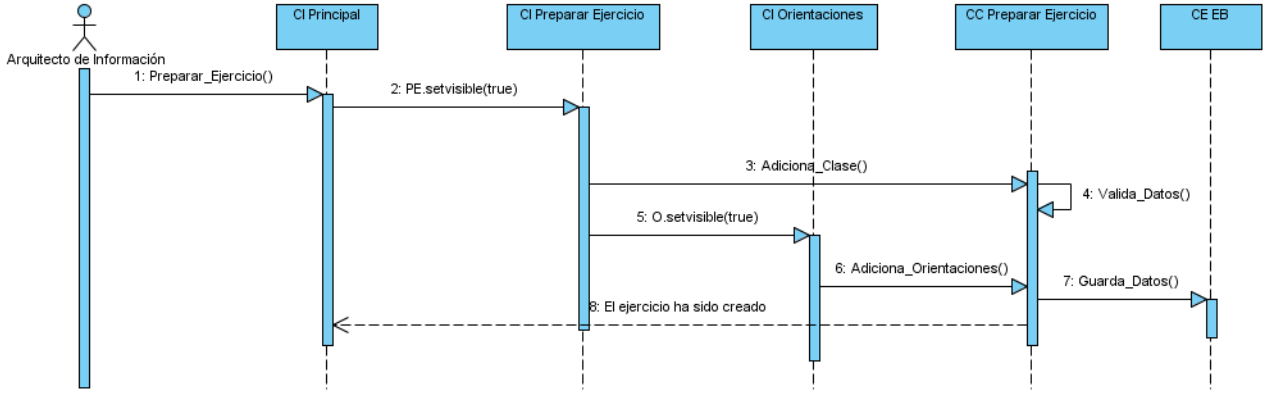


**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Bloque: Escenario Modificar**

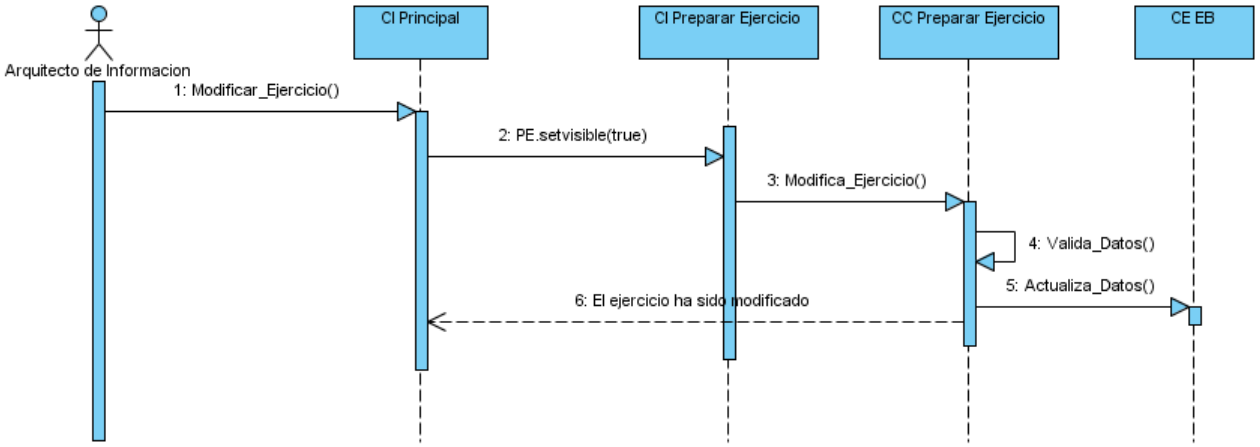



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Bloque: Escenario Eliminar**

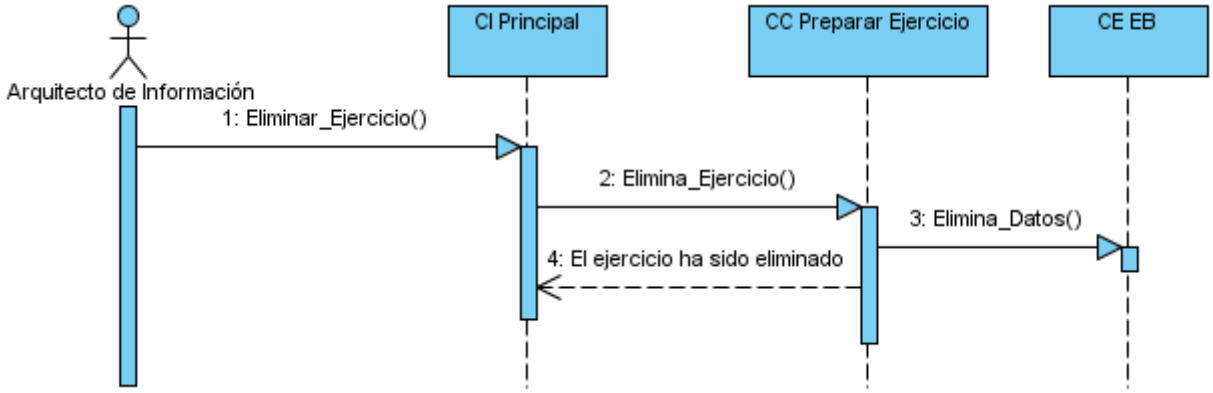



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Clase: Escenario Adicionar**

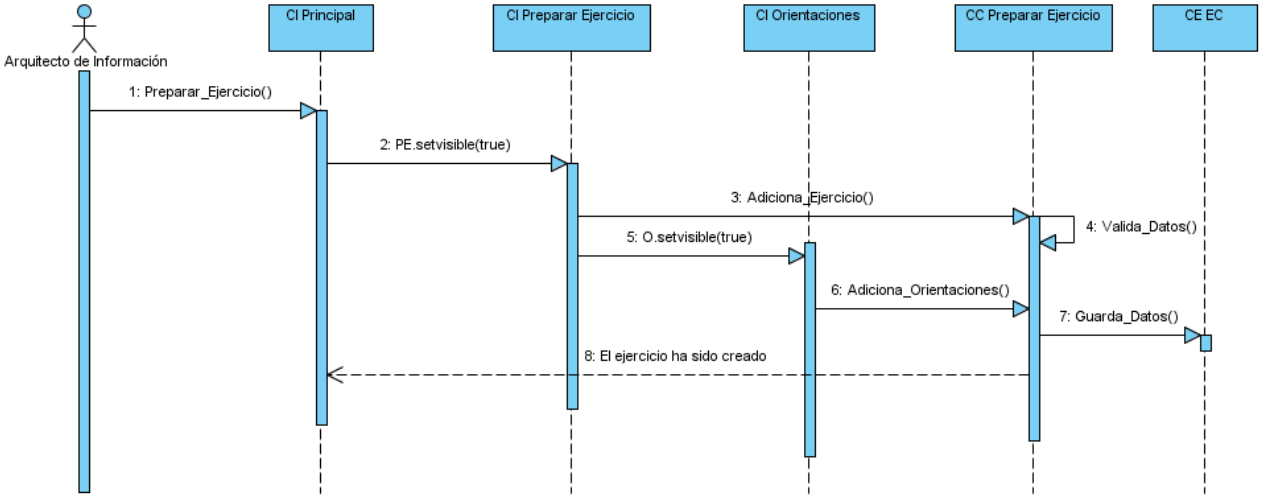


**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Clase: Escenario Modificar**

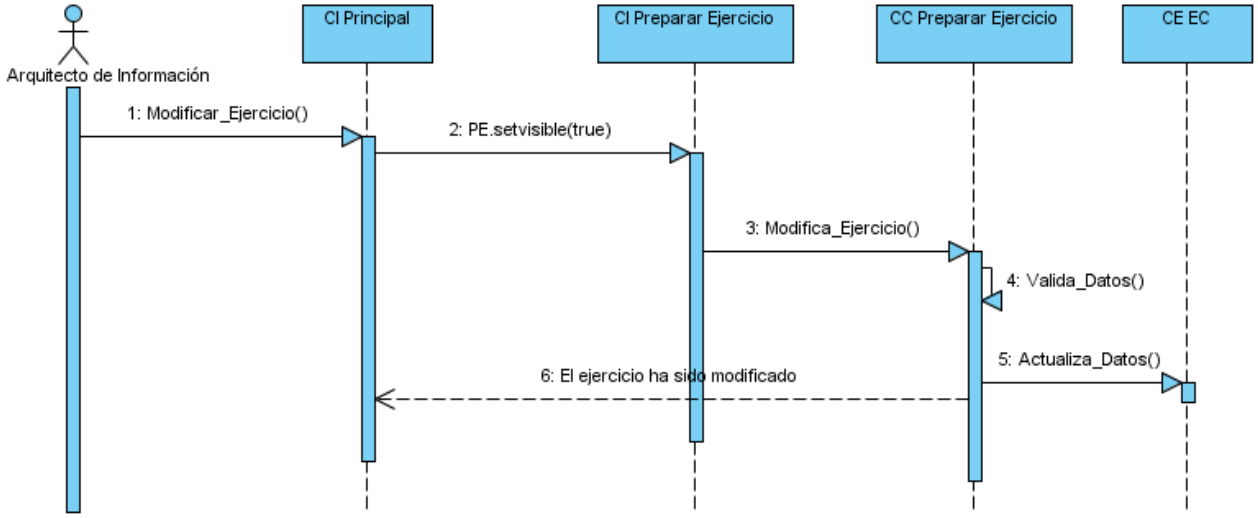



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Clase: Escenario Eliminar**

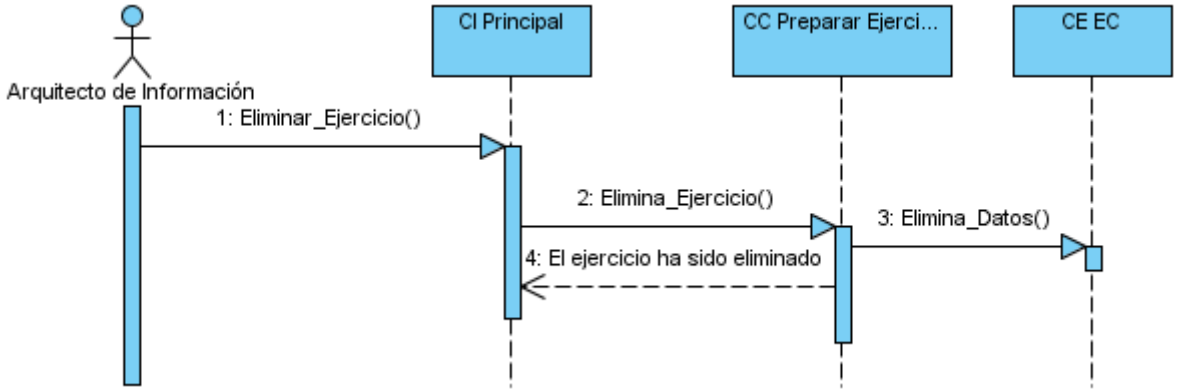



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Secuencia: Escenario Adicionar**

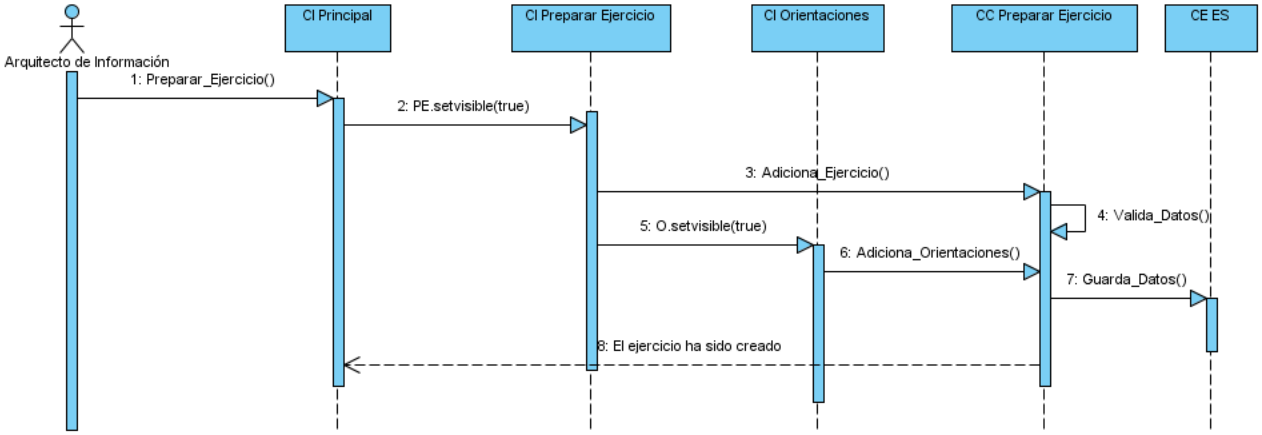


**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Secuencia: Escenario Modificar**

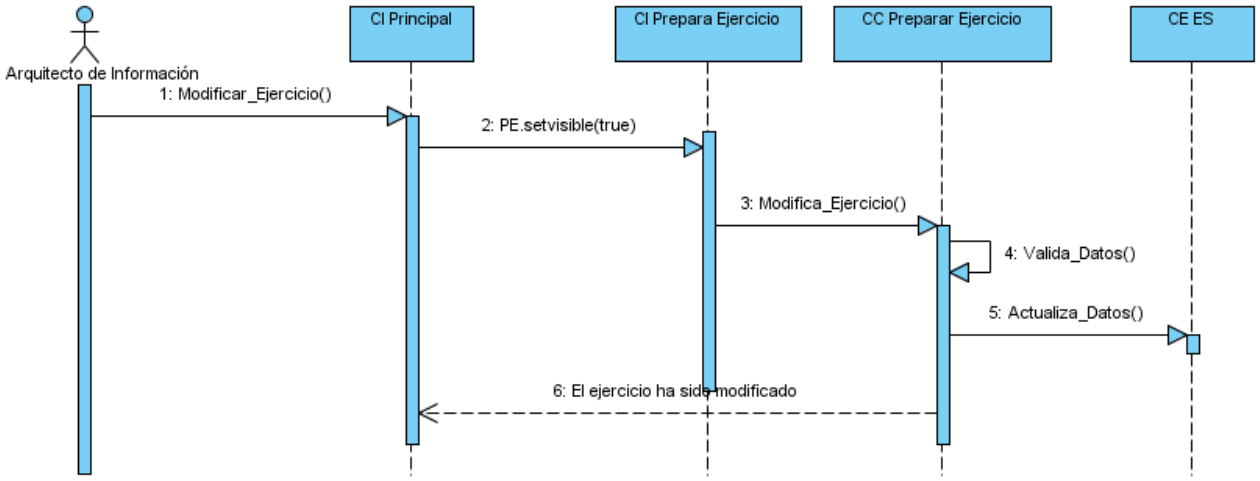



**Diagrama de Secuencia Caso de Uso Gestionar Ejercicio Secuencia: Escenario Eliminar**

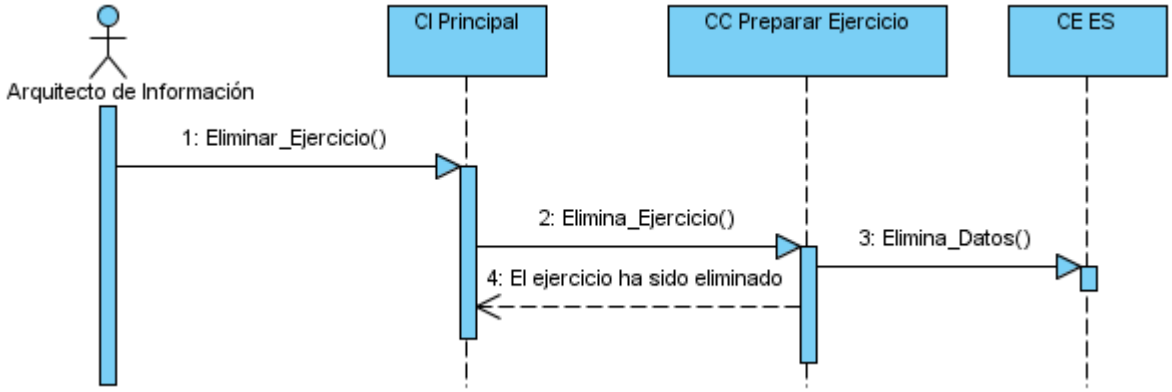
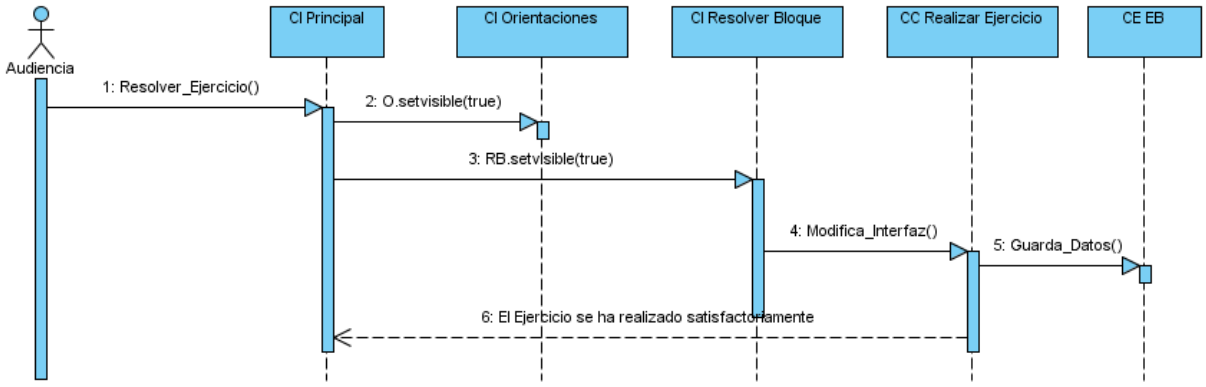
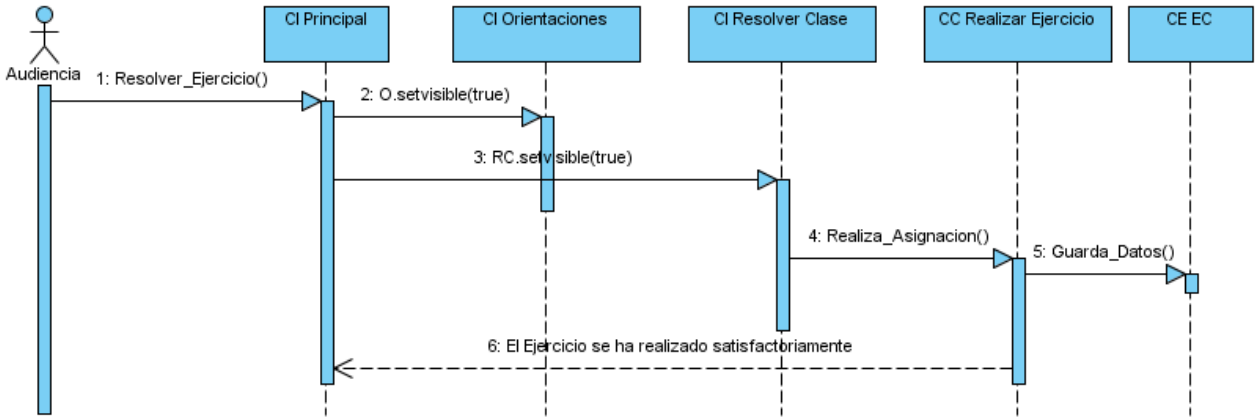


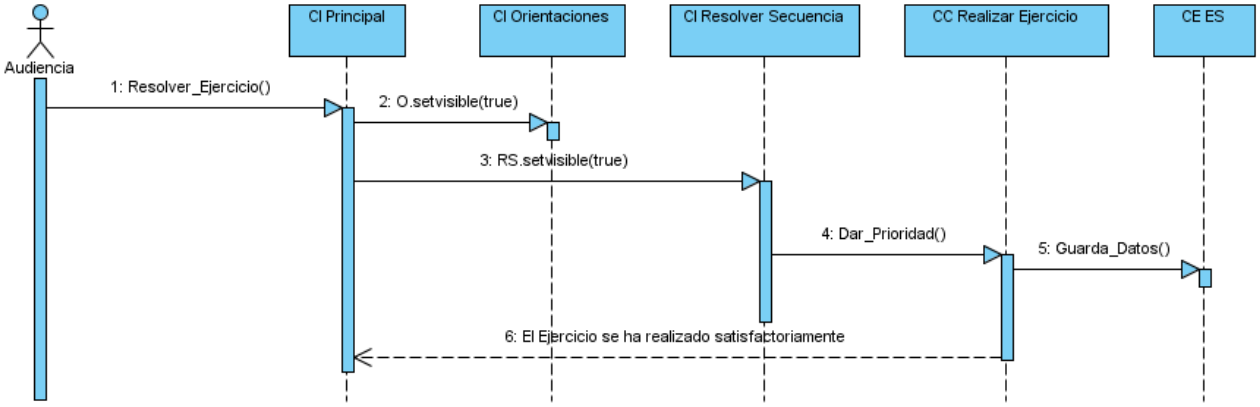

**Diagrama de Secuencia Caso de Uso Resolver Ejercicio Bloque**



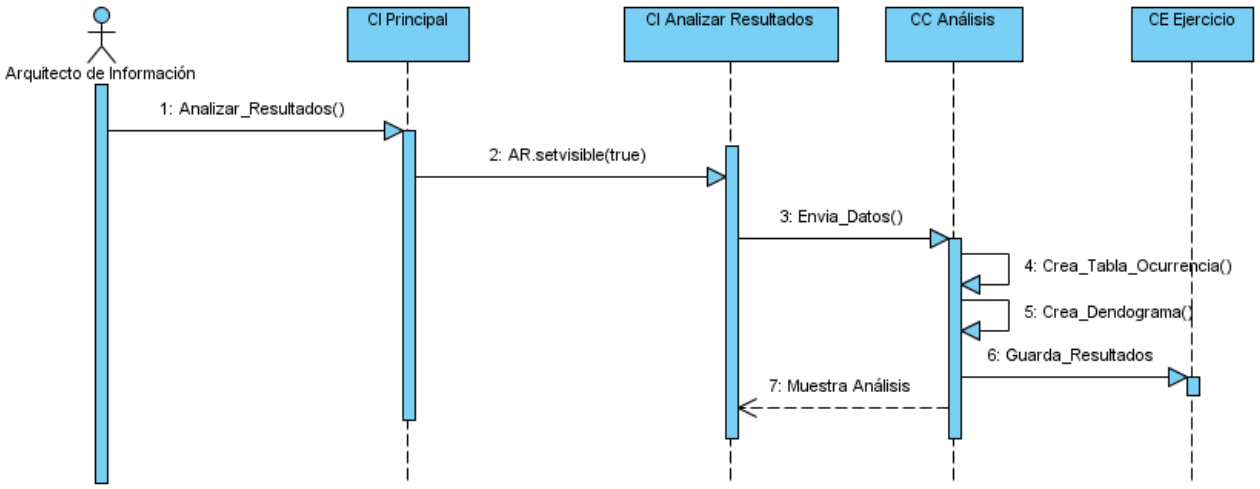
### Diagrama de Secuencia Caso de Uso Resolver Ejercicio Clase




### Diagrama de Secuencia Caso de Uso Resolver Ejercicio Secuencia




### Diagrama de Secuencia Caso de Uso Analizar Resultados



### 3.4.3 Descripción de las Clases del Diseño

| **Nombre: Principal** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| lista_proyecto | Lista<Proyecto> |
| **Para cada responsabilidad**: | |
| **Nombre:** | Adicionar_Proyecto |
| **Descripción:** | Inserta un nuevo proyecto con los datos pedidos. |
| **Nombre:** | Modificar_Proyecto dado un nombre. |
| **Descripción:** | Realiza cambios al proyecto(Ej: cambio de tipo de ejercicio). |
| **Nombre:** | Eliminar_Proyecto. |
| **Descripción:** | Elimina un proyecto dado un nombre. |

| **Nombre: Proyecto** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| id_proyecto | int |
| nombre | string |
| **Para cada responsabilidad**: | |
| **Nombre:** | Validar_Datos |
| **Descripción:** | Comprueba que los datos introducidos sean correctos de acuerdo a las especificidades del software. |
| **Nombre:** | Adicionar_Ejercicio. |

| | |
|---|---|
| **Descripción:** | Inserta un nuevo ejercicio, al proyecto. |
| **Nombre:** | Modificar_Ejercicio. |
| **Descripción:** | Modifica un ejercicio dado un id. |
| **Nombre:** | Eliminar_Ejercicio. |
| **Descripción:** | Elimina un ejercicio dado un id. |
| **Nombre:** | Adicionar Tarjeta. |
| **Descripción:** | Adiciona tarjeta con los datos pertinentes. |
| **Nombre:** | Modificar Tarjeta. |
| **Descripción:** | Modifica una tarjeta dado su nombre. |
| **Nombre:** | Eliminar Tarjeta. |
| **Descripción:** | Elimina una tarjeta dado su nombre. |
| **Nombre:** | Adicionar Clase. |
| **Descripción:** | Adiciona clase con los datos pertinentes. |
| **Nombre:** | Modificar Clase. |
| **Descripción:** | Modifica una clase dado su nombre. |
| **Nombre:** | Eliminar Clase. |
| **Descripción:** | Elimina una clase dado su nombre. |

| **Nombre: Tarjeta** | |
|---|---|
| **Tipo de clase: Entidad** | |
| **Atributo** | **Tipo** |
| nombre | string |
| descripción | string |

| **Para cada responsabilidad**: | |
|---|---|
| **Nombre:** | Get_Nombre |
| **Descripción:** | Devuelve el nombre de la tarjeta seleccionada. |
| **Nombre:** | Set_Descripción |
| **Descripción:** | Cambia el nombre de la tarjeta seleccionada. |
| **Nombre:** | Get_Descripción |
| **Descripción:** | Devuelve la descripción de la tarjeta seleccionada. |
| **Nombre:** | Set_Descripción |
| **Descripción:** | Cambia la descripción de la tarjeta seleccionada. |

| **Nombre: Clase** | |
|---|---|
| **Tipo de clase: Entidad** | |
| **Atributo** | **Tipo** |
| nombre | string |
| descripción | string |
| **Para cada responsabilidad**: | |
| **Nombre:** | Get_Nombre |
| **Descripción:** | Devuelve el nombre de la clase seleccionada. |
| **Nombre:** | Set_Descripción |
| **Descripción:** | Cambia el nombre de la clase seleccionada. |
| **Nombre:** | Get_Descripción |
| **Descripción:** | Devuelve la descripción de la clase seleccionada. |
| **Nombre:** | Set_Descripción. |

| | |
|---|---|
| **Descripción:** | Cambia la descripción de la clase seleccionada. |

| | | |
|---|---|---|
| **Nombre: ECA** | | |
| **Tipo de clase: Entidad** | | |
| **Atributo** | | **Tipo** |
| id | | int |
| orientaciones | | string |
| lista_tarjetas | | Lista<Tarjeta> |
| **Para cada responsabilidad**: | | |
| **Nombre:** | Get_Id | |
| **Descripción:** | Devuelve el id de la clase seleccionada. | |
| **Nombre:** | Set_Id | |
| **Descripción:** | Cambia el id de la clase seleccionada. | |
| **Nombre:** | Get_Orientaciones | |
| **Descripción:** | Devuelve las orientaciones de la clase seleccionada. | |
| **Nombre:** | Set_Orientaciones. | |
| **Descripción:** | Cambia las orientaciones de la clase seleccionada. | |
| **Nombre:** | Get_Lista_Tarjetas | |
| **Descripción:** | Devuelve la lista de tarjeta del ejercicio. | |
| **Nombre:** | Set_Lista_Tarjetas | |
| **Descripción:** | Cambia la lista de tarjeta del ejercicio. | |

| |
|---|
| **Nombre: ECC y ES** |

| **Tipo de clase: Entidad** | |
|---|---|
| **Atributo** | **Tipo** |
| lista_clase | Lista<Clase> |
| **Para cada responsabilidad**: | |
| **Nombre:** | Get_Lista_Clases |
| **Descripción:** | Devuelve la lista de clases pertenecientes al ejercicio. |
| **Nombre:** | Set_Lista_Clases |
| **Descripción:** | Cambia la lista de clases perteneciente al ejercicio. |

| **Nombre: Ejercicio de Bloque** | |
|---|---|
| **Tipo de clase: Entidad** | |
| **Atributo** | **Tipo** |
| id_eb | int |
| orientaciones | String |
| lista_clase | Lista<Clase> |
| **Para cada responsabilidad**: | |
| **Nombre:** | Get_Id_Eb |
| **Descripción:** | Devuelve el id del ejercicio. |
| **Nombre:** | Get_Orientaciones |
| **Descripción:** | Devuelve las orientaciones dado un id. |
| **Nombre:** | Set_ Orientaciones |
| **Descripción:** | Cambia las orientaciones dado un id. |
| **Nombre:** | Get_Lista_Clase |

| | |
|---|---|
| **Descripción:** | Devuelve la lista de las clases asociadas el ejercicio. |
| **Nombre:** | Set_Lista_Clase |
| **Descripción:** | Cambia la lista de las clases asociadas el ejercicio. |

| **Nombre: Resolver Ejercicio** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| lista_EC_ES | Lista<Ejercicio> |
| lista_EB | Lista<Ejercicio> |
| **Para cada responsabilidad**: | |
| **Nombre:** | Get_Lista_EC_ES |
| **Descripción:** | Devuelve la lista de ejercicios de Clases o Secuencia. |
| **Nombre:** | Get_ Lista_EB |
| **Descripción:** | Devuelve la lista de ejercicios de bloque. |
| **Nombre:** | Realizar Asignación |
| **Descripción:** | Permite Asignarle a clases las tarjetas escogidas por la audiencia. |
| **Nombre:** | Dar Prioridad |
| **Descripción:** | Permite darle una prioridad a las tarjetas dentro de las clases según entienda la audiencia. |
| **Nombre:** | Modificar Interfaz |
| **Descripción:** | Se organizan las clases en los lugares donde la audiencia entienda que son más accesibles dentro de la interfaz dada. |

| **Nombre: Resultados** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| lista_ejercicio | Lista<Ejercicio> |
| **Para cada responsabilidad**: | |
| **Nombre:** | Crear Tabla de Co-Ocurrencia |
| **Descripción:** | Se crea una tabla de co-ocurrencias en una hoja de cálculo. |
| **Nombre:** | Crear Dendograma |
| **Descripción:** | Arroja una representación geométrica de las categorías. |

**Conclusiones**

En este capítulo se han presentado los diagramas de clases del análisis y del diseño de los casos de uso del sistema y los diagramas de secuencia de los casos de uso del sistema. Se ha logrado modelar los procesos que han sido objeto de estudio; proporcionando una idea completa de lo que realmente es el software.

Los diagramas y especificaciones de diseño que se proponen constituyen una guía que puede ser fácilmente leída y comprendida por aquellos que construirán el código, por los que lo probarán y le darán mantenimiento.

**Conclusiones**

Con la realización del trabajo se obtuvo una información detallada de la técnica Card Sorting a partir de un estudio realizado de la misma.

Se lograron identificar las funcionalidades necesarias para dar una solución óptima.

Se realizó el análisis y diseño de las clases, para que el sistema final brinde facilidades de uso a los arquitectos de la información.

El prototipo de interfaz no funcional desarrollado facilitará la posterior implementación del sistema.

**Recomendaciones**

Este documento puede ser consultado por los desarrolladores del proyecto GIDI para la futura implementación del sistema.

Redefinir el modelo de diseño cuando se continúe la realización del resto de los flujos de trabajo propuestos por la metodología RUP (Implementación, Prueba y Despliegue).

Investigar a profundidad la forma en que se obtienen los resultados de la técnica.

**Referencias Bibliográficas**

[1] Revisión de técnicas de arquitectura de información. [Disponible en]: http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm

[2] El Profesional de la Información, 2004, marzo-abril. [Disponible en]: http://www.nosolousabilidad.com/hassan/cardsorting.pdf

[3] Gestión de la información en las organizaciones. Principios, conceptos y aplicaciones. s.1.: Empresa grafica Haydee Santamaría, Palma Soriano (Versión Digital)

[4] Information Architecture Institute.

[5] La Técnica de Card Sorting. [Disponible en]: http://www.nosolousabilidad.com/

[6] Hassan Montero, Y. 6 Marzo 2005. Card Sword: Organización de categorías centrada en el usuario. [Disponible en]: http://www.nosolousabilidad.com/hassan/curso_madrid.htm

[7] ALBA Software provee de la tecnología y el soporte necesarios para que cualquier empresa se administre todos sus servicios Internet, obteniendo así un mayor control, seguridad, integración con su sistema informático, y autonomía para la implementación de nuevos servicios.

[8] WEBCAT. Consultado 20 febrero del 2008. [Disponible en:] http://www.gestorweb.com/docu/webcat_intro.html

[9] Materiales del curso "Arquitectura de Información para la Web (3ª Edición)".

http://www.bitacoras.sidar.org/g4/index.php?2005/03/06/5-card-sword-organizacion-de-categorias-centrada-en-el-usuario-ejemplo-de-prueba-con-cardsword

[10] Ingeniería de Software 1[Disponible en]: http//teleformacion.uci.cu

[11] Ídem a la referencia 10.

[12] RUP Conferencia de Ingeniería de Software Introducción a la Ingeniería de Software. [Disponible en]: http//teleformacion.uci.cu

[13] Métrica 3 [Disponible en]: http://www.csae.map.es/csi/metrica3/introduccion.pdf

[14] Ídem a la referencia 12.

[15] Herramientas CASE.[Disponible en]: http://es.wikipedia.org/wiki/CASE#Lista_de_aplicaciones_CASE.

[16] Enterprise Architect [Disponible en]:
http://www.apexnet.com.ar/index.php/product/viewProducts/24/sl=0

[17] Visual Paradigm [Disponible en]:
http://www.softpile.com/Development/Editors_and_IDEs/Review_19078_index.html

[18] Rational Rose [Disponible en]:
http://www.ciao.es/Rational_Rose_Enterprise_Edition__Opinion_612900

[19] Umbrello [Disponible en]: http://es.wikipedia.org/wiki/Umbrello

[20] NetBeans 6.0 [Disponible en]: http://es.wikipedia.org/wiki/NetBeans _6.0

[21] Java [Disponible en]: http://es.wikipedia.org/wiki/ Java

[22] Documentación del Proyecto GIDI.

[23] JACOBSON, I. El proceso Unificado de Desarrollo de Software. La habana, 2004. 438 p.

[24] Ídem a la referencia 23.

[25] Conferencia de Ingeniería de Software Patrones de Diseño. [Disponible en]:
http//teleformacion.uci.cu

[26] Ídem a la referencia 24.

[27] Ídem a la referencia 24.

[28] Ídem a la referencia 24.

[28] Ídem a la referencia 24.

**Bibliografía**


- Ayuda urgente con Card Sorting, concretamente con CardCluster . [Online] septiembre 17, 2007. [Citado: mayo 3, 2008.] http://www.cadius.org/pipermail/lista_cadius.org/2007-September/005511.html.
- **Carlos García, Juan.** Card Sorting. El medio es el msje. *Card Sorting. El medio es el msje .* [Online] [Citado: mayo 4, 2008.] http://usalo.es/63/card-sorting-el-medio-es-el-mensaje/.
- **Carreras Plaza, Jesús y Guaderrama Hernández, Maritza.** El Enfoque Cualitativo en el desarrollo de Arquitecturas de Información: Card Sorting + Entrevista Abierta 2006**.**
- **Catuxa. 2005.** Card Sorting en la práctica. *Card Sorting en la práctica.* [Online] mayo 5, 2005. [Citado: mayo 20, 2008.] http://www.deakialli.com/2005/05/05/card-sorting-en-la-practica/.
- Ejemplo de desarrollo software utilizando la metodología XP. *Ejemplo de desarrollo software utilizando la metodología XP.* [Online] enero 9, 2004. [Citado: mayo 7, 2008.] http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/.
- Extreme Programming: A gentle introduction. Extreme Programming: A gentle introduction. [Online] febrero 17, 2006. [Citado: mayo 7, 2008.] http://www.extremeprogramming.org/.
- **Hassan Montero, Yusef and Martín Fernández, Francisco J. 2004.** Estructuración de la Información: Aproximación descendente. *Estructuración de la Información: Aproximación descendente.* [Online] enero 30, 2004. [Citado: mayo 13, 2008.] Estructuración de la Información: Aproximación descendente.
- **Hassan Montero, Yusef and Núnez Peña, Ana. 2005.** Diseño de Arquitecturas de Información: Descripción y Clasificación. *Diseño de Arquitecturas de Información: Descripción y Clasificación.* [Online] enero 14, 2005. [Citado: mayo 15, 2008.] http://www.nosolousabilidad.com/articulos/descripcion_y_clasificacion.htm.
- **Larman, Craig**. UML y Patrones. Introducción al análisis y diseño orientado a Objetos Volumen I. Prentice Hall Hispanoamericana, S.A. , 1999
- **Larman, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a Objetos Volumen II. Prentice Hall Hispanoamericana, S.A., 1999.
- **León Pavón, Eduardo: 2 abril, 2007.** Visual Paradigm, una herramienta de lo más útil. *Visual Paradigm, una herramienta de lo más útil.* [Online] abril 2, 2 abril, 2007. [Citado: mayo 8, 2008.] http://slion2000.blogspot.com/2007/04/visual-paradigm-una-herramienta-de-lo.html.
- **Miguel Ángel. 2006.** Creando pruebas de Card Sorting (agrupación de tarjetas) con CardSword. *Creando pruebas de Card Sorting (agrupación de tarjetas) con CardSword.* [Online] abril 10, 2006. [Citado: mayo 4, 2008.] http://mi-

blog.com/migue/2006/08/10/creando-pruebas-de-card-sorting-agrupacion-de-tarjetas-con-cardsword/.

- **Montero, Hassan and Yusef. 2006.** Indización Social y Recuperación de Información. *Indización Social y Recuperación de Información.* [Online] noviembre 16, 2006. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/indizacion_social.htm.
- **NIELSEN, JACKOB. 2004.** Card Sorting: A cuántos usuarios se necesita evaluar. *Card Sorting: A cuántos usuarios se necesita evaluar.* [Online] agosto 12, 2004. [Citado: mayo 15, 2008.] http://www.proyectoweb.cubaweb.cu/boletin/card-sorting-a-cuantos-usuarios-se-necesita-evaluar.html.
- **Ortega Santamaría, Sergio. 20005.** Desarrollo Conceptual y la técnica de Card Sorting. *Desarrollo Conceptual y la técnica de Card Sorting.* [Online] diciembre 14, 20005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/desarrollo_conceptual.htm.
- **Pressman, R**. Ingeniería de software. Un enfoque práctico Parte 1. Interamericana de España, S.A., 2002.
- **Pressman, R**. Ingeniería de software. Un enfoque práctico Parte 2. Interamericana de España, S.A., 2002.
- **Rdickelman. 2006.** IBM EZSort: Card Sorting Software. *IBM EZSort: Card Sorting Software.* [Online] junio 24, 2006. [Citado: mayo 3, 2008.] http://www.epsscentral.info/knowledgebase/desdev/ibmezsort.
- **Ronda León, Rodrigo. 2005.** La Arquitectura de Información y las Ciencias de la Información. En No Solo Usabilidad. *La Arquitectura de Información y las Ciencias de la Información. En No Solo Usabilidad.* [Online] mayo 4, 2005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/ai_cc_informacion.htm.
- **Rondan León, Rodrigo and Mesa Rábade, Yaima. 2005.** Análisis de Secuencia: una herramienta para la Arquitectura de Información. *Análisis de Secuencia: una herramienta para la Arquitectura de Información.* [Online] julio 6, 2005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/analisis_secuencia.htm.
- **Rondan León, Rodrigo. 2008.** Arquitectura de Información: análisis histórico-conceptual. *Arquitectura de Información: análisis histórico-conceptual.* [Online] abril 28, 2008. [Citado: mayo 17, 2008.] http://nosolousabilidad.com/articulos/historia_arquitectura_informacion.htm.
- **Schilb, Steffen.** About CardSort. *About CardSort.* [Online] [Citado: marzo 20, 2008.] http://www.cardsort.net/.

- Sistemas de Clasificación de Información. *Sistemas de Clasificación de Información.* [Online] febrero 14, 2004. [Citado: mayo 15, 2008.] http://www.nosolousabilidad.com/articulos/sistemas_clasificacion.htm.
- Umbrello.[Citado el: 23 de noviembre de 2007.] http://es.wikipedia.org/wiki/Umbrello.
- UML.[Citado el: 7 de noviembre de 2007.] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado.
- Visual Paradigm en http://es.wikipedia.org/wiki/Visual_Paradigm.

## Glosario de Términos

### Arquitecto de Información

El que estudia la Arquitectura de la Información.

### Arquitectura de la Información

Estudio de la organización de la información para llevar al usuario a encontrar un camino hacia el conocimiento y hacia la comprensión de la información.

### Audiencia

Personas encargadas de realizar los ejercicios propuestos por el arquitecto de información.

### Clases

Tópicos principales de la aplicación dada (Ej: Docencia, Servicios Telemáticos).

### Diseñador

Persona que solicita la realización de la arquitectura de la información a una aplicación dada.

### Establecimiento de Clases

Ejercicio que forma parte de la técnica de Card Sorting, se encarga específicamente de asignar tarjetas a clases.

### Establecimiento Secuencia

Ejercicio que forma parte de la técnica de Card Sorting, se encarga específicamente de dar una prioridad a las tarjetas dentro de las clases.

### Establecimiento de Bloque

Ejercicio que forma parte de la técnica de Card Sorting, se encarga específicamente de organizar las clases en la parte de la interfaz que crea más lógica de ubicar.

### Gestión de la Información

Proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico la gestión del ciclo de vida de este recurso y ocurre en cualquier organización. Es propia también de unidades especializadas que manejan este recurso en forma intensiva, llamadas unidades de información.

**Sistema de Etiquetado**

Un sistema de representación que utiliza términos y/o expresiones para identificar, de la forma más inequívoca posible, el contenido informativo. Las etiquetas serán entonces términos que representen sin lugar a error, bloques significativos de información.

**Técnica de Card Sorting**

Consiste en escribir en unas tarjetas el nombre y descripción de todas las secciones de un programa, y pedir a los usuarios que ordenen las tarjetas en grupos de una manera que les parezca lógica.

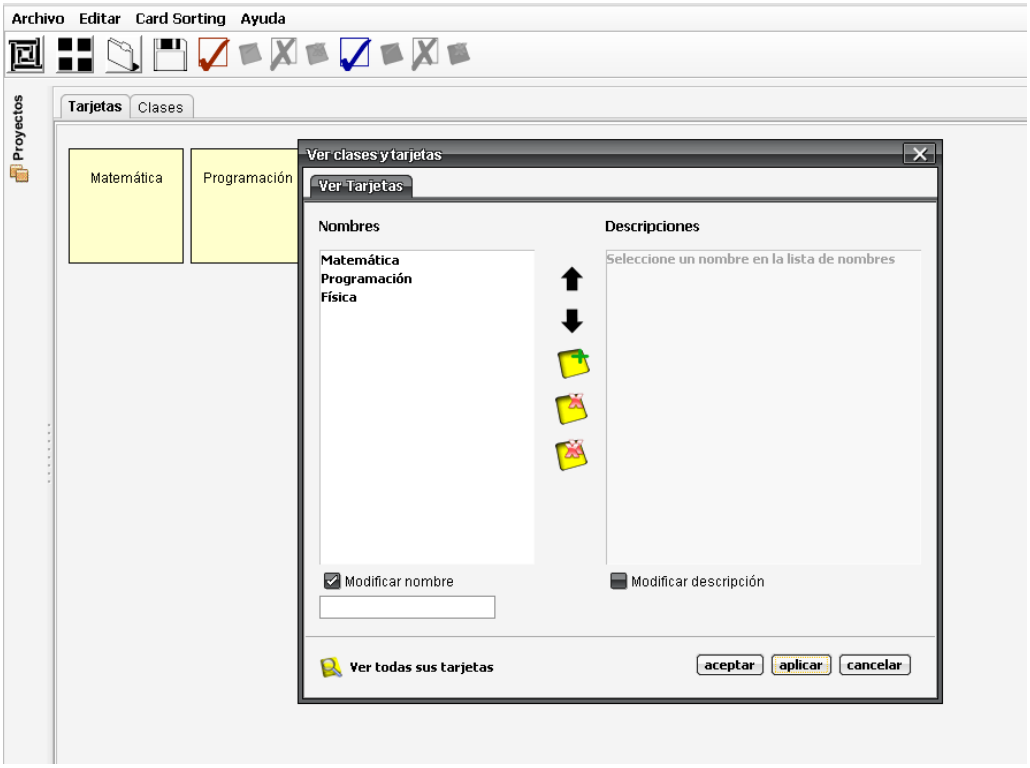
**Tarjetas**

Sub-tópicos de la aplicación dada (Ej: Matemática, Cuota de Internet ).

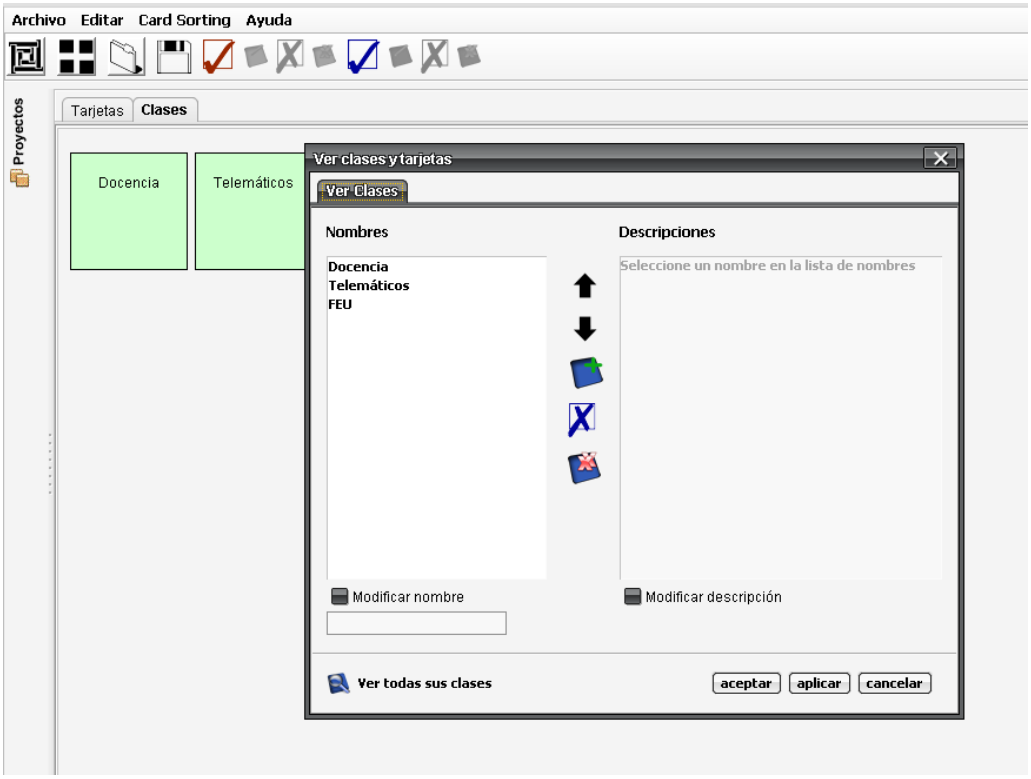
**Anexos**

Caso de Uso Gestionar Proyecto.

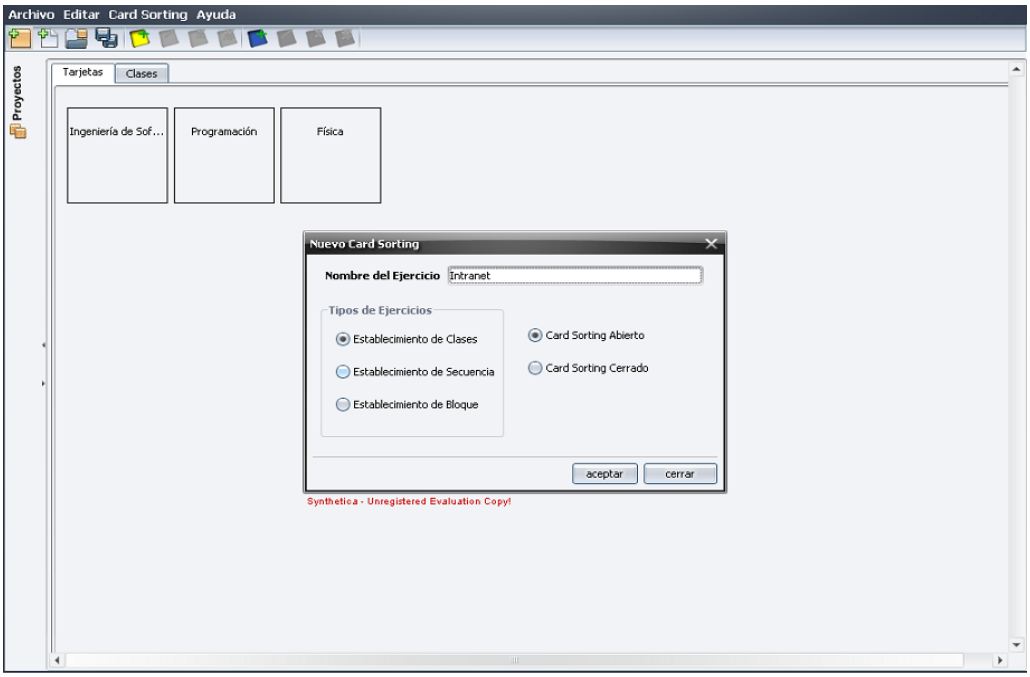

Caso de Uso Gesionar Tarjetas.



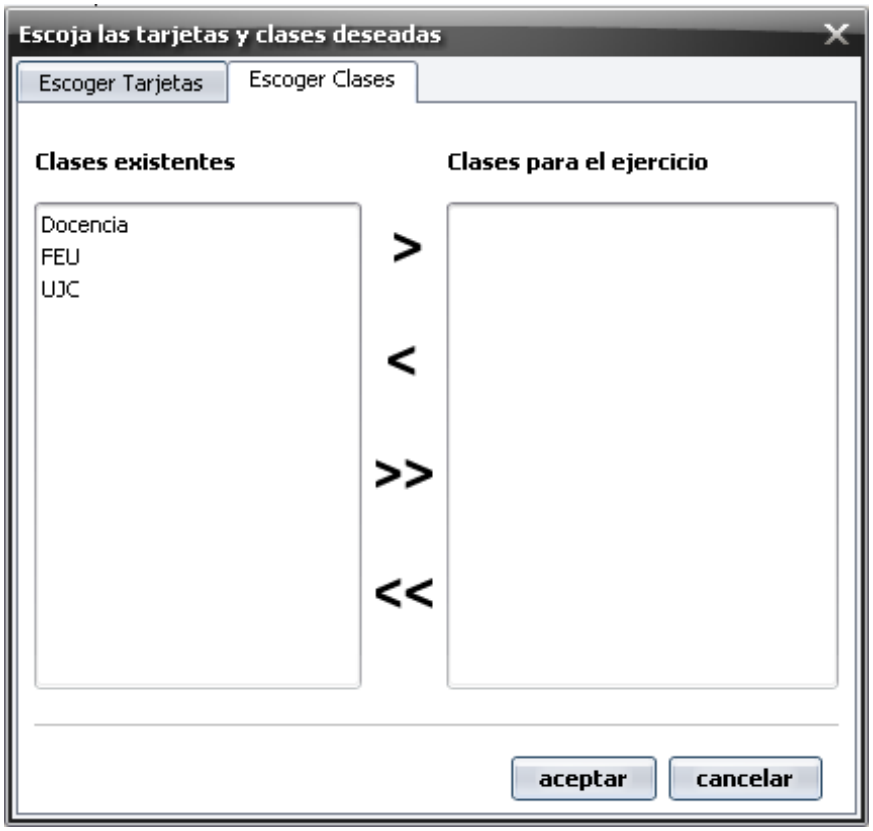
Caso de Uso Gestionar Clases.



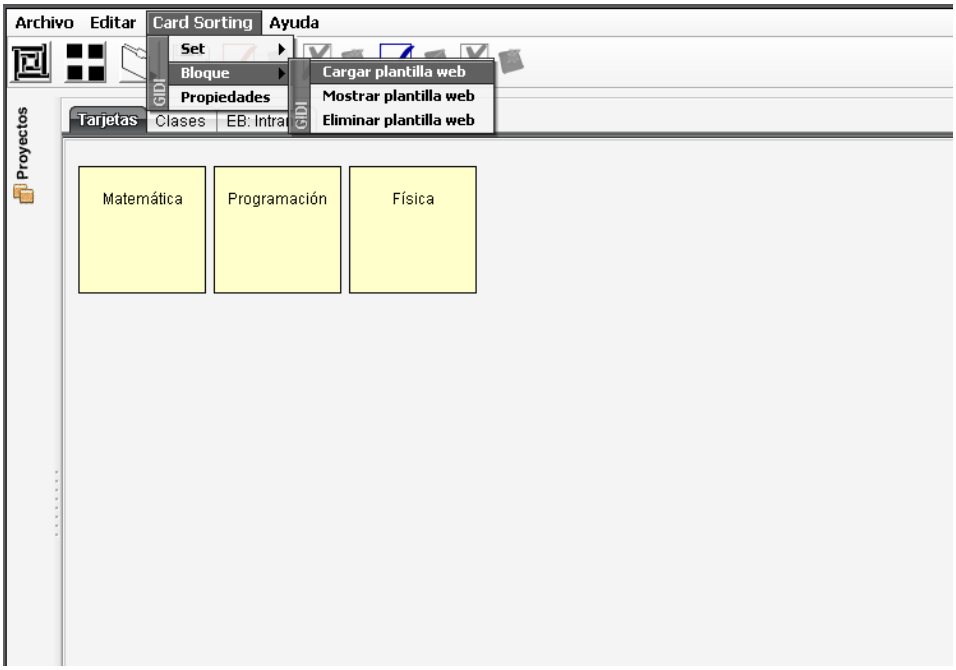

Adicionar Ejercicio:

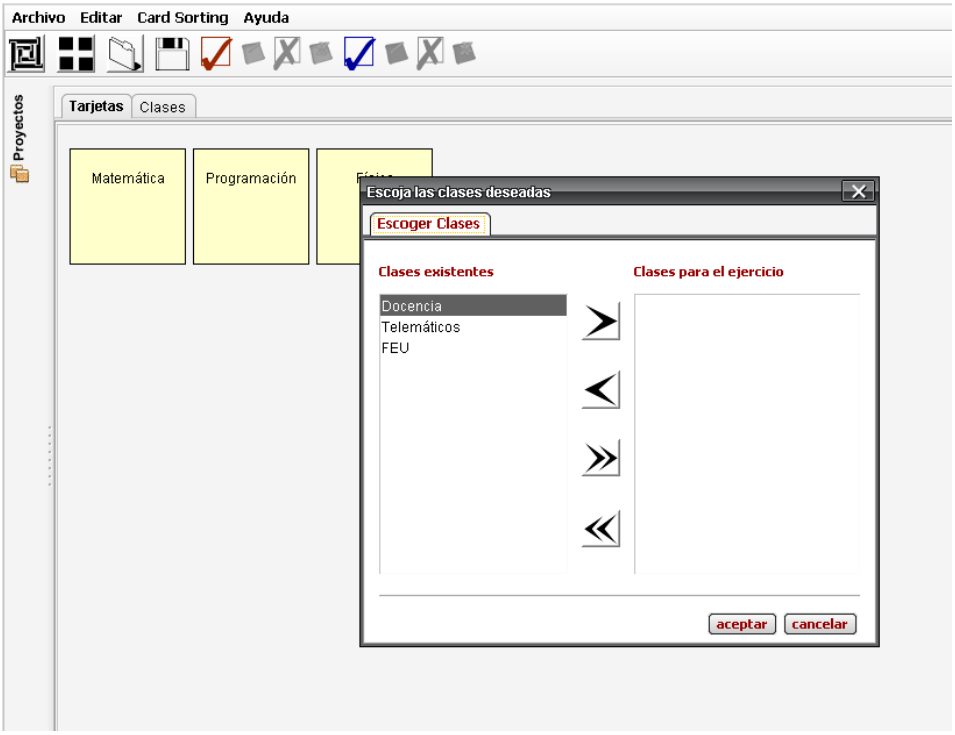


Caso de Uso Ejercicio de Establecimiento de Clases y Secuencia.

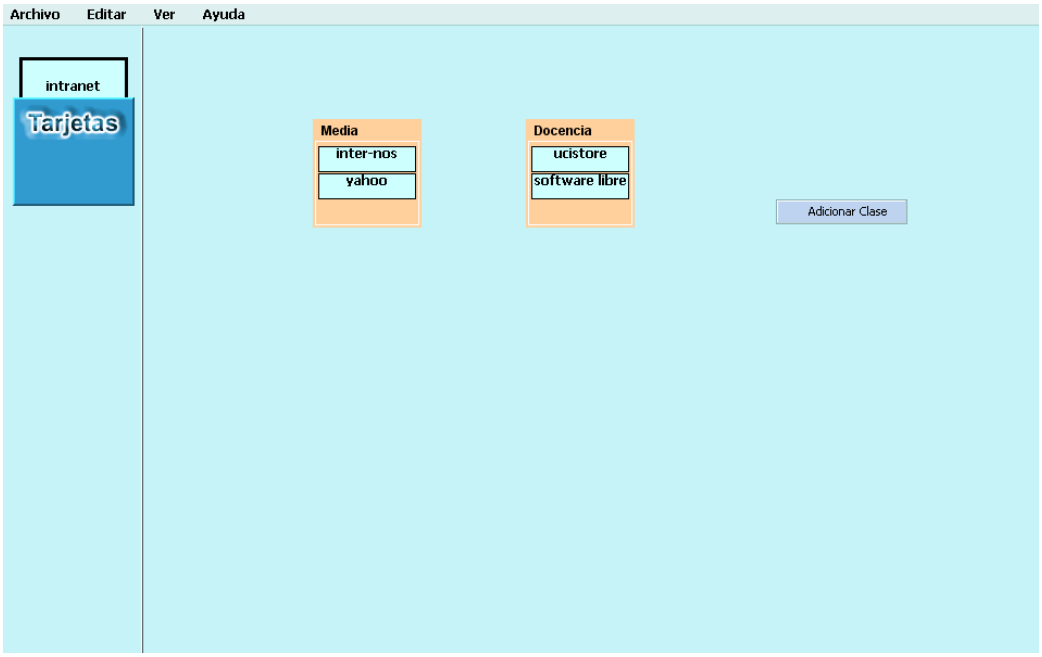



Caso de Uso Ejercicio Establecimiento de Bloque.

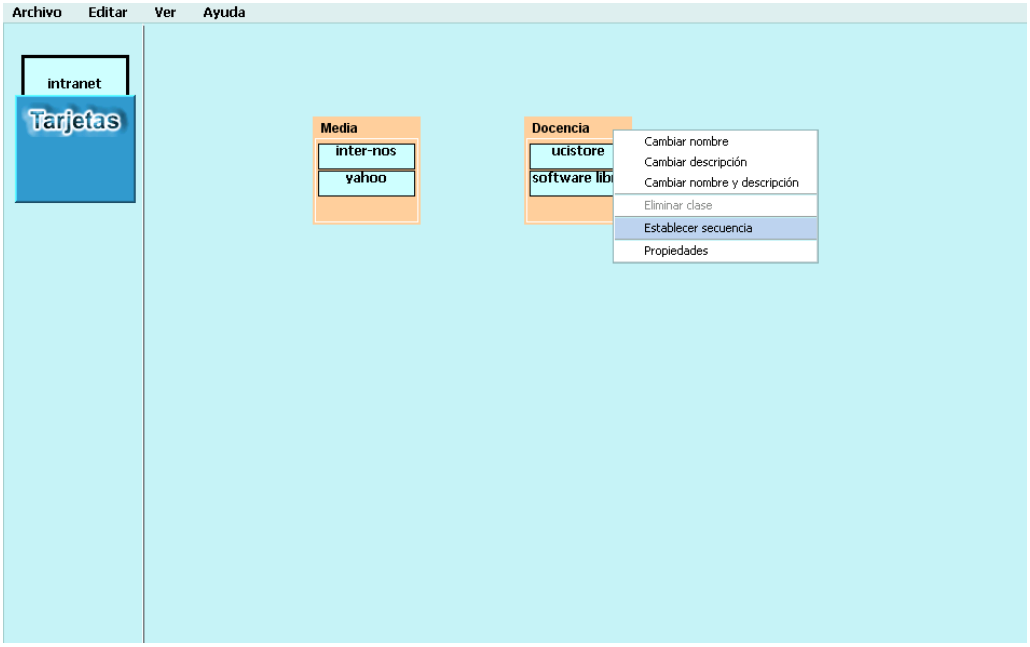


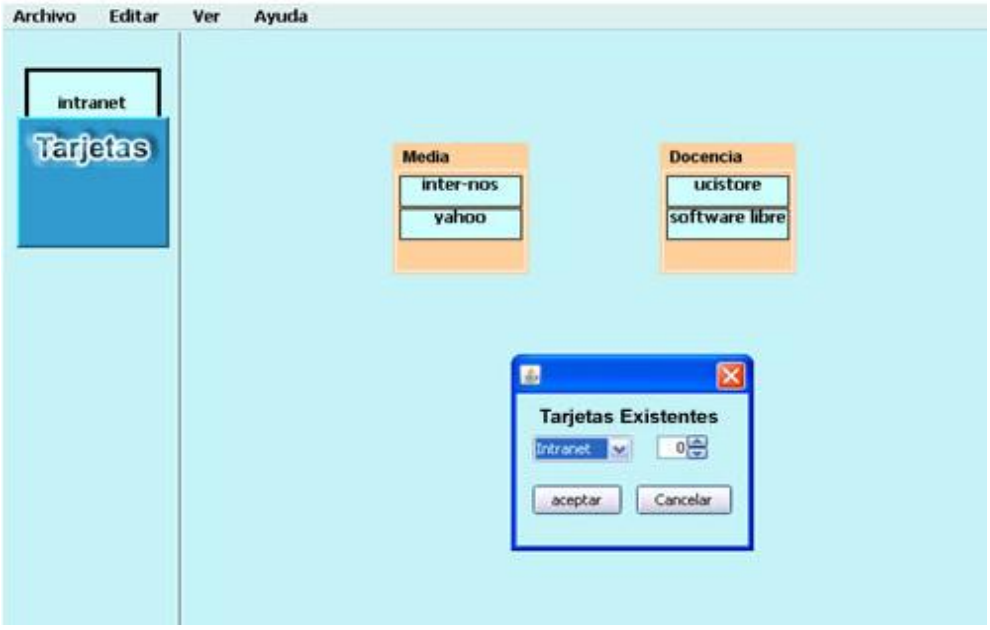



Caso de Uso Resolver Establecimiento de Clase.



Caso de Uso Resolver Establecimiento de Secuencia.



Caso de Uso Resolver Ejercicio de Bloque.