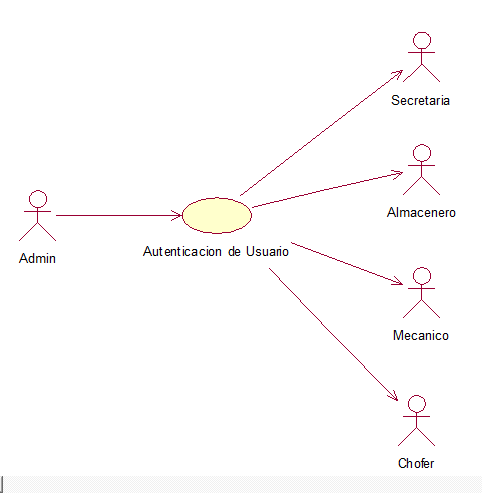
**Directrices Diseñando Visualmente**

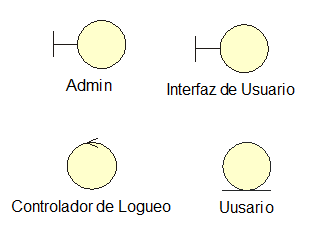
**Introducción**

Se usa las técnicas de modelado visual para diseñar software puede ayudar a descomponer problemas complejos, Compartir imágenes en lugar de documentos escritos o código fuente también ayuda a la comprensión y la comunicación de los conceptos difíciles. La adopción de las notaciones de modelado estándar como el UML aumenta esta capacidad, ayudando a hacer diagramas precisos y sin ambigüedades. El grado de formalidad utiliza cuando la producción y difusión de modelos deben variar de acuerdo a sus necesidades.



**Identificar los elementos**

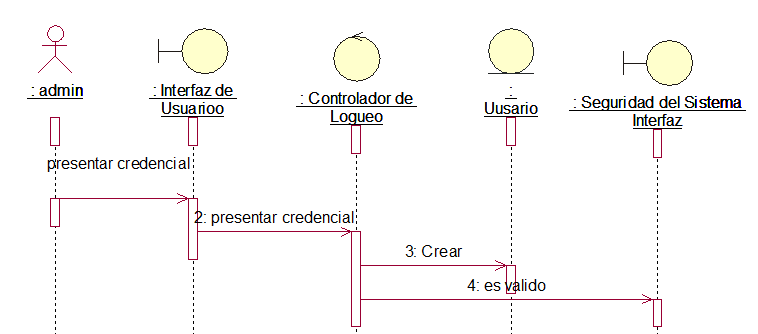
Los elementos de diseño identificados como clases en un diagrama UML. Aplican estereotipos apropiados y, hacen que la clase utilice un icono específico para el estereotipo para caracterizar la intención de la clase en el diseño.



**Determinar cómo los elementos colaboran para darse cuenta de la situación**

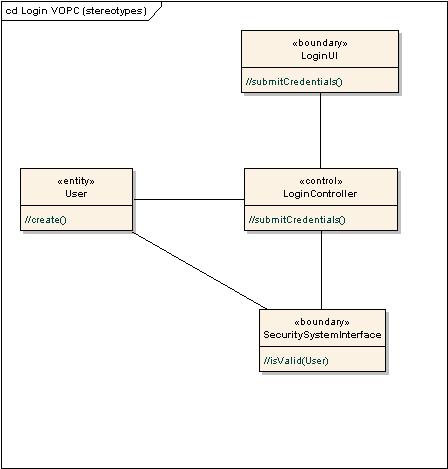
La determinación de los elementos que colaboran son dos tipos de diagramas:

* Diagrama de objeto dinámico, mostrando cómo los elementos de diseño colaboran para darse cuenta de las necesidades.
* Diagrama de clase estática, que muestra las clases involucradas en la realización de los requisitos



El diagrama de secuencia de arriba muestra las credenciales de usuario que se pasa a través del sistema de seguridad para la autenticación.

Puede ser útil para crear uno o más diagramas estáticas de clases que muestran las clases en el diseño que apoyan la realización. Estos diagramas de clases son a menudo llamados Vista del Participante Clases diagramas, que proporcionan una visión centrada en el diseño general sólo muestra las clases, relaciones, operaciones y atributos relevantes para la colaboración.



Este diagrama muestra las operaciones y las relaciones que fueron identificados por dibujar el diagrama de secuencia. Las relaciones en este ejemplo no se han refinado aún, por lo que sólo se muestran como simples asociaciones. Recuerde que debe examinar el diagrama para verificar que el diseño puede apoyar el comportamiento en el diagrama de secuencia.

Trabajar a este nivel de detalle en el modelo durante las primeras etapas de diseño puede ser útil. Mantiene los diagramas relativamente simple y fácil de entender. Esto los hace más fáciles de dibujar en un taller y más fácil de cambiar durante el debate. A menudo es más fácil añadir el detalle una vez que hay acuerdo sobre los fundamentos.

**REFINE DECISIONES DE DISEÑO**

Una vez que los fundamentos del diseño son relativamente estables, podemos comenzar a agregar detalles al diseño. Parte de este se puede realizar en código o en el modelo. Si se elige el modelado, entonces refinar atributos, responsabilidades y relaciones.

**DESCRIBIR LAS RESPONSABILIDADES**

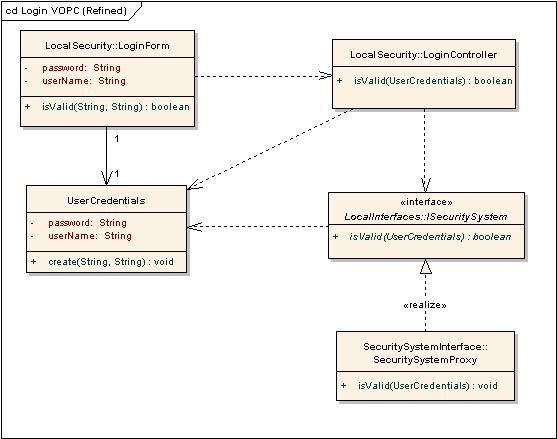
Las responsabilidades de clase son acciones a realizar por un objeto o conocimiento mantienen y proporcionan a otros objetos. Cada clase tendrá típicamente varias responsabilidades; cada responsabilidad se convertirá en una o más operaciones durante el diseño.

Las responsabilidades se derivan de los mensajes en los diagramas de interacción o de los requisitos no funcionales que una clase tiene que apoyar. Documentar una responsabilidad, dándole un nombre y, opcionalmente, una descripción breve.

**Describir los atributos y asociaciones**

Una clase puede tener que almacenar información de datos simple, como: cadena, entero, y similares. Para tal tipo simple de información, atributos se definen para las clases. Para un atributo de "comportamiento" más complejo o, considerar la creación de una clase extra y establecer una asociación a la misma.

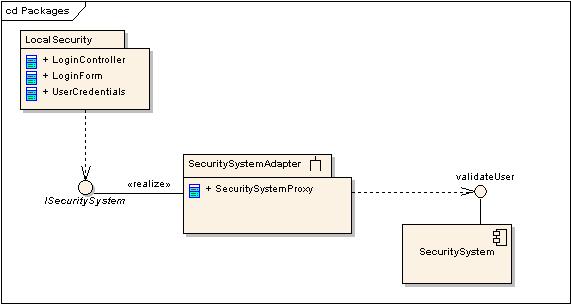
Para llevar a cabo sus responsabilidades, las clases pueden depender de otras clases para abastecer la conducta necesaria. Estas otras clases podrían ser los ya identificados en esta sesión de diseño, podrían ser clases extraídos de la arquitectura, o la necesidad de nuevas clases existentes puede ser concebida. Asociaciones en un diagrama de clases se pueden utilizar para representar relaciones entre clases.



Este diagrama muestra una serie de refinamientos. La clase LoginUI ha sido reemplazado por LoginForm. La clase de usuario ha sido UserCredentials renombrados y es creado por la clase LoginForm lugar de LoginController. Se utiliza entonces como un parámetro para los mensajes posteriores en lugar de pasar los atributos individuales. La clase SecuritySystemInterface se ha refinado en dos elementos, ISystemSecurity, que ofrece una fachada sencilla para la interacción con los restos de la concepción; y SecuritySystemProxy, que se ocupa de la interacción con el sistema de seguridad externa.

**Sistema de funcionamiento interno de diseño.**

Las clases en el diseño es probable que tenga que ser distribuidos entre los diferentes paquetes y subsistemas o componentes.



En este ejemplo, el LoginForm, LoginController y UserCredentials elementos han sido colocados en un paquete llamado LocalSecurity. El SecuritySystemProxy es una parte de un subsistema llamado SecuritySystemAdapter que da cuenta de la interfaz ISecuritySystem. El SecuritySystemAdapter envuelve el SecuritySystem legado, expresa aquí como un componente que ofrece una interfaz ValidateUser.

Cada uno de estos elementos empaquetados pueden ser distribuidos entre el equipo para su posterior trabajo de desarrollo.

**Conclusión**

Esta directriz caminó a través de las técnicas de una manera concreta comenzó con un escenario de un caso de uso a través de la distribución de las clases identificadas en un conjunto de paquetes. Este ejemplo demuestra una técnica para diseñar visualmente, pero debe ser considerado como una sola pasada conceptual de diseño. Se podría aplicar la misma facilidad con esta técnica en la definición de los detalles internos de cómo la clase SecuritySystemProxy colaborará con un conjunto de clases para validar las credenciales.

Al aplicar esta directriz, trabajar en pequeños trozos y tener en cuenta el objetivo de la distribución de software a los usuarios que proporciona valor. Para entregar software de alta calidad requiere la consideración de cómo las piezas trabajarán juntos para ofrecer ese valor. Pero tan pronto como las decisiones fundamentales se han hecho y las decisiones se han comunicado a los miembros del equipo adecuados, el equipo debe pasar a la implementación del código fuente para verificar el diseño y entregar el valor.