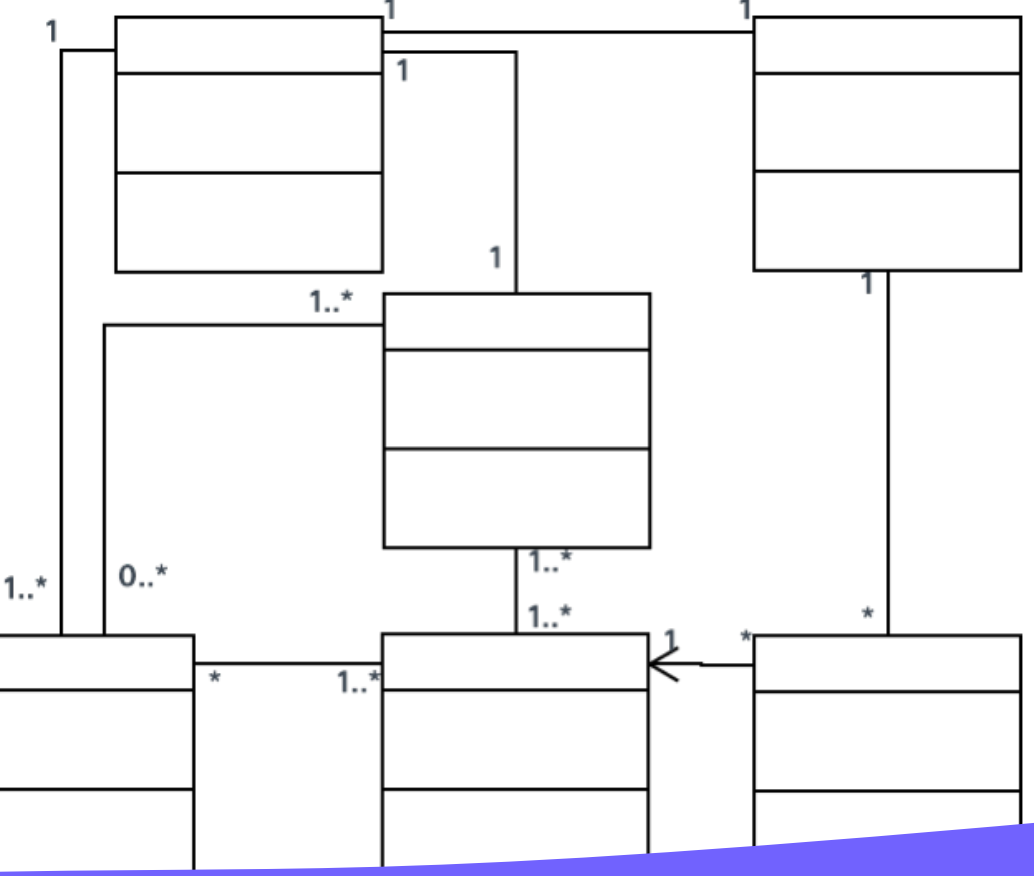
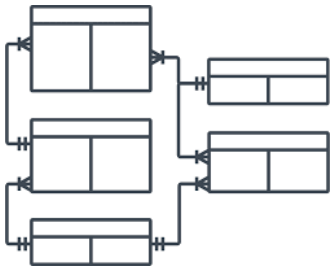




Diseño de Sistemas

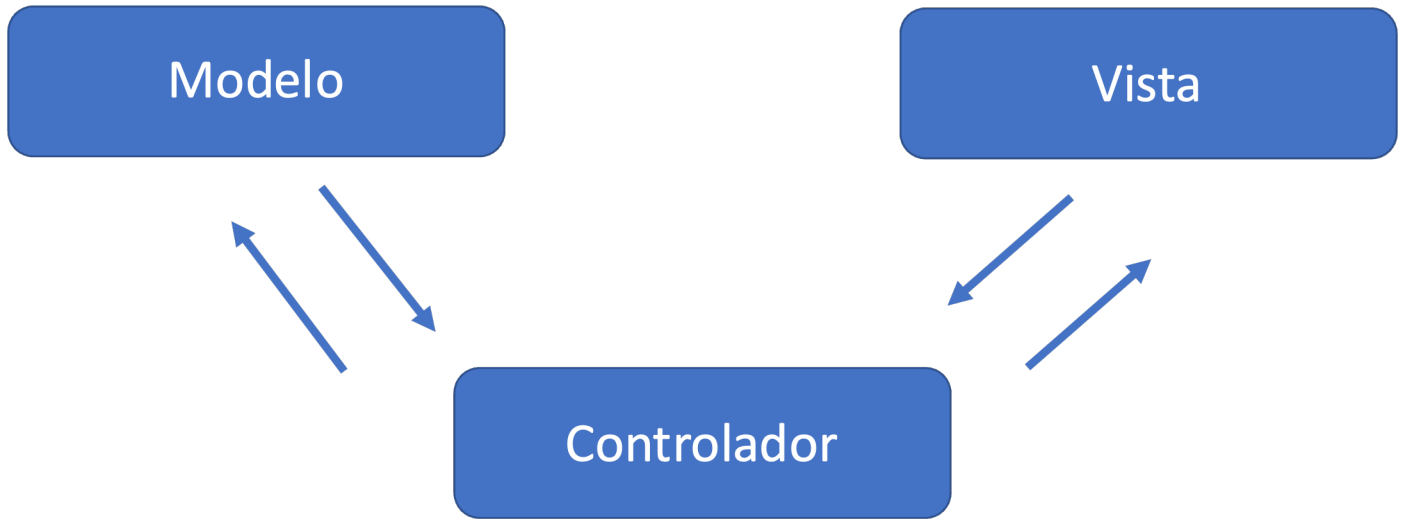
Agenda

- MVC: Modelo – Vista – Controlador
- Patrón Repositorio
- Modelado de Usuarios, Roles & Permisos

MVC (Patrón Arquitectónico)

MVC

Modelo-vista-controlador (MVC) es un **patrón** de arquitectura de software **de interacción** que separa la arquitectura en tres componentes: el modelo, la vista y el controlador.



MVC - Modelo

Está vinculado con la representación de los datos con los cuales el Sistema opera. Está relacionado a la lógica y las reglas del negocio.

- Encapsula el estado del sistema
- Gestiona todos los accesos a dichos datos (consultas, actualizaciones, etc.)
- Valida la especificación de la lógica detallada en el “negocio”
- Envía a la Vista, a través del controller, los datos solicitados para que sean visualizados

MVC - Controlador

Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud sobre los datos.

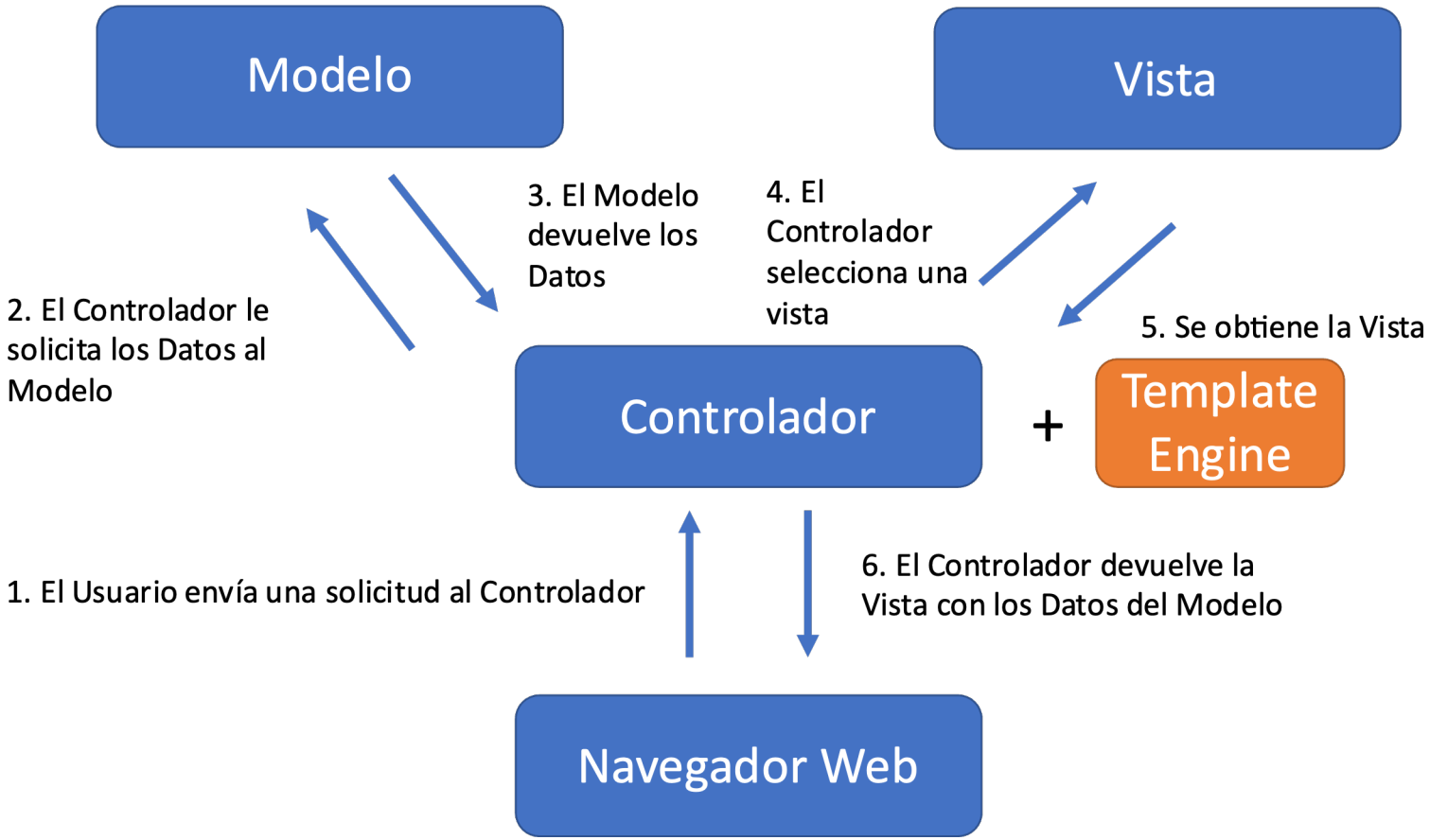
- Es el intermediario entre el modelo y la vista
- Define el comportamiento del sistema
- Traduce acciones del usuario a actualizaciones del modelo
- Es el gestor del ciclo de vida del sistema
- Responde a eventos
- Invoca peticiones al modelo

MVC - Vista

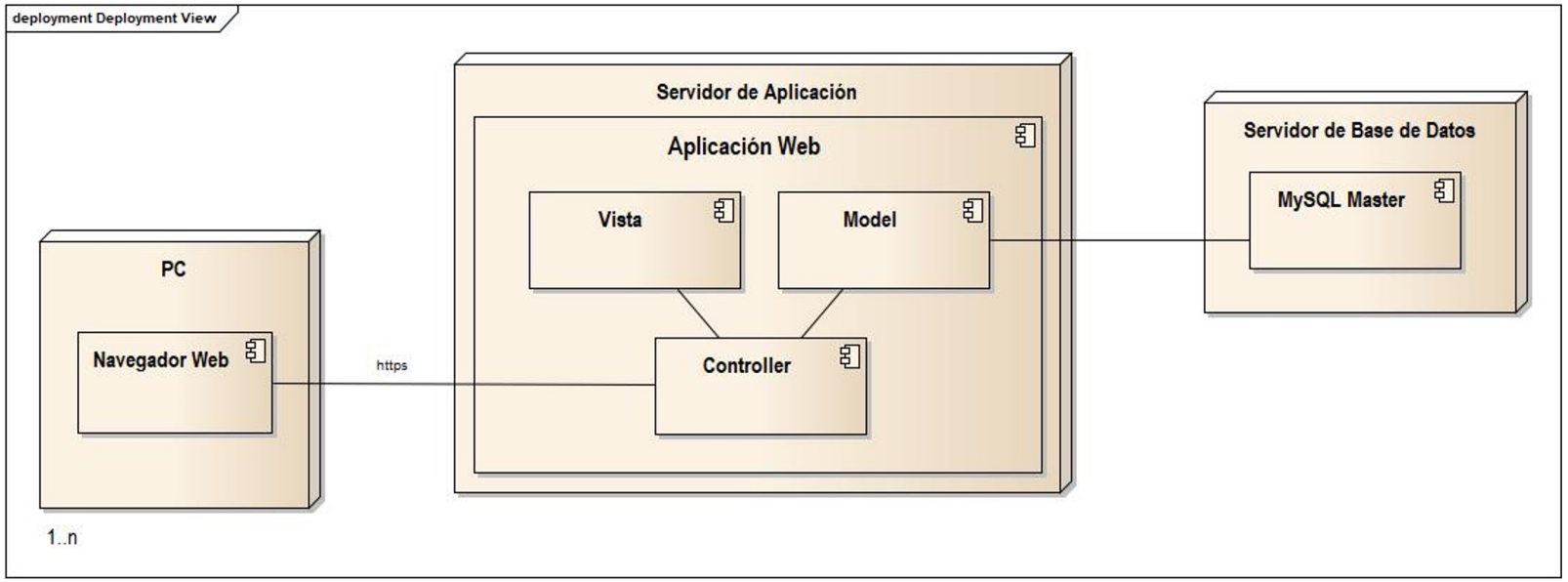
Presenta los datos y su forma de interactuar en un formato adecuado para el usuario.

- Posee lógica para poder representar los datos de la forma más “amigable” para el usuario
- Envía acciones del usuario al controlador
- Solicita actualizaciones al modelo a través del controller
- La comunicación entre la vista y el controlador se realiza mediante objetos de transferencia (*DTO - Data Transfer Objects*)

MVC - Web


Modelo
Vista
3. El Modelo devuelve los Datos
4. El Controlador selecciona una vista
2. El Controlador le solicita los Datos al Modelo
5. Se obtiene la Vista
Controlador
+
Template Engine
1. El Usuario envía una solicitud al Controlador
6. El Controlador devuelve la Vista con los Datos del Modelo
Navegador Web

MVC - Web


deployment Deployment View
Servidor de Aplicación
Aplicación Web
Vista
Model
Controller
Servidor de Base de Datos
MySQL Master
PC
Navegador Web
https
1..n

MVC – Validación de Datos

¿Dónde se validan los datos?

- ***EN CADA CAPA***. Esto se lo denomina ***validación en profundidad***.
- En la vista se puede validar campos requeridos y consistencia (números y/o letras donde corresponda, entre otras cosas).
- En el controlador se debería volver a validar la consistencia de los datos y los campos requeridos. También se puede realizar una validación adicional contra un servicio externo.
- En el modelo se deberían realizar las validaciones explícitas del negocio.

MVC – Ventajas y Desventajas

Ventajas

- Buena separación de intereses (concerns)
- Reusabilidad de Vistas y Controladores
- Flexibilidad

Desventajas

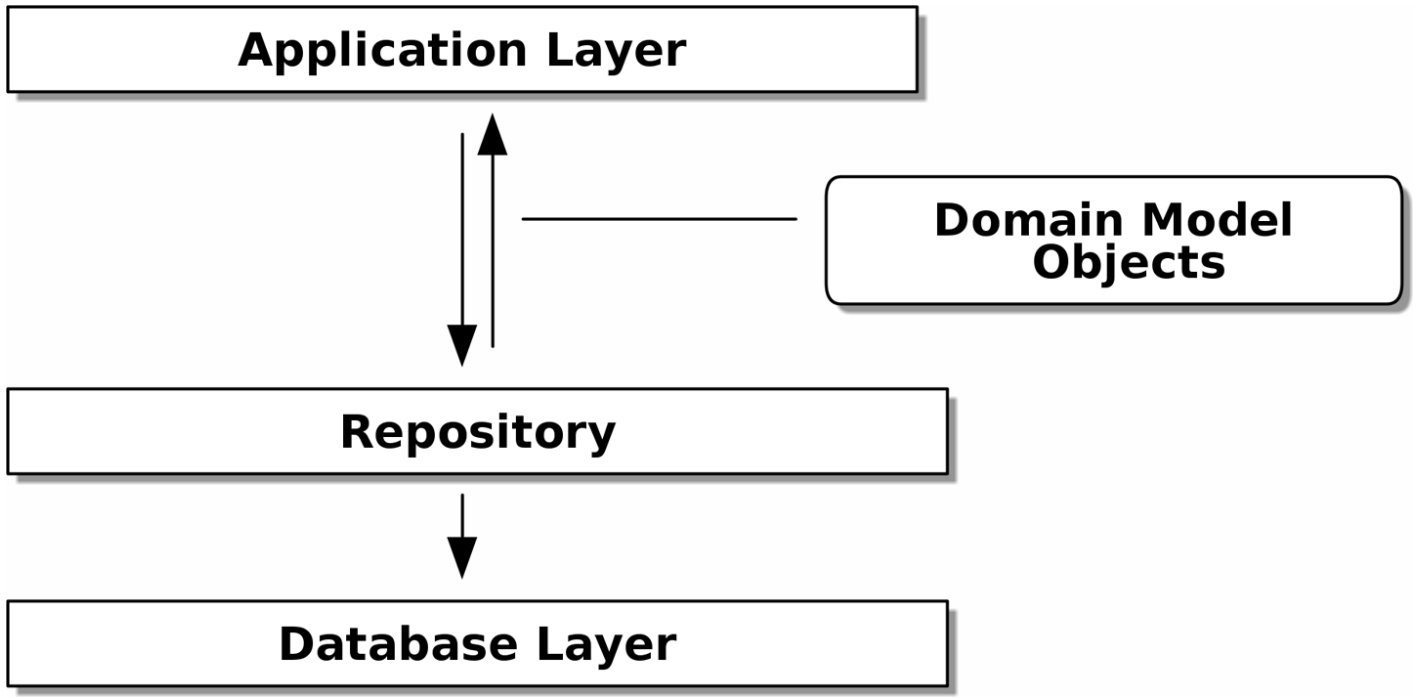
- Mayor complejidad de los Sistemas
- No siempre útil en aplicaciones con "*Poca Interactividad*" o con "*Vistas Simples*"
- Difícil de Testear "como un todo"

Patrón Repositorio

Patrón Repositorio

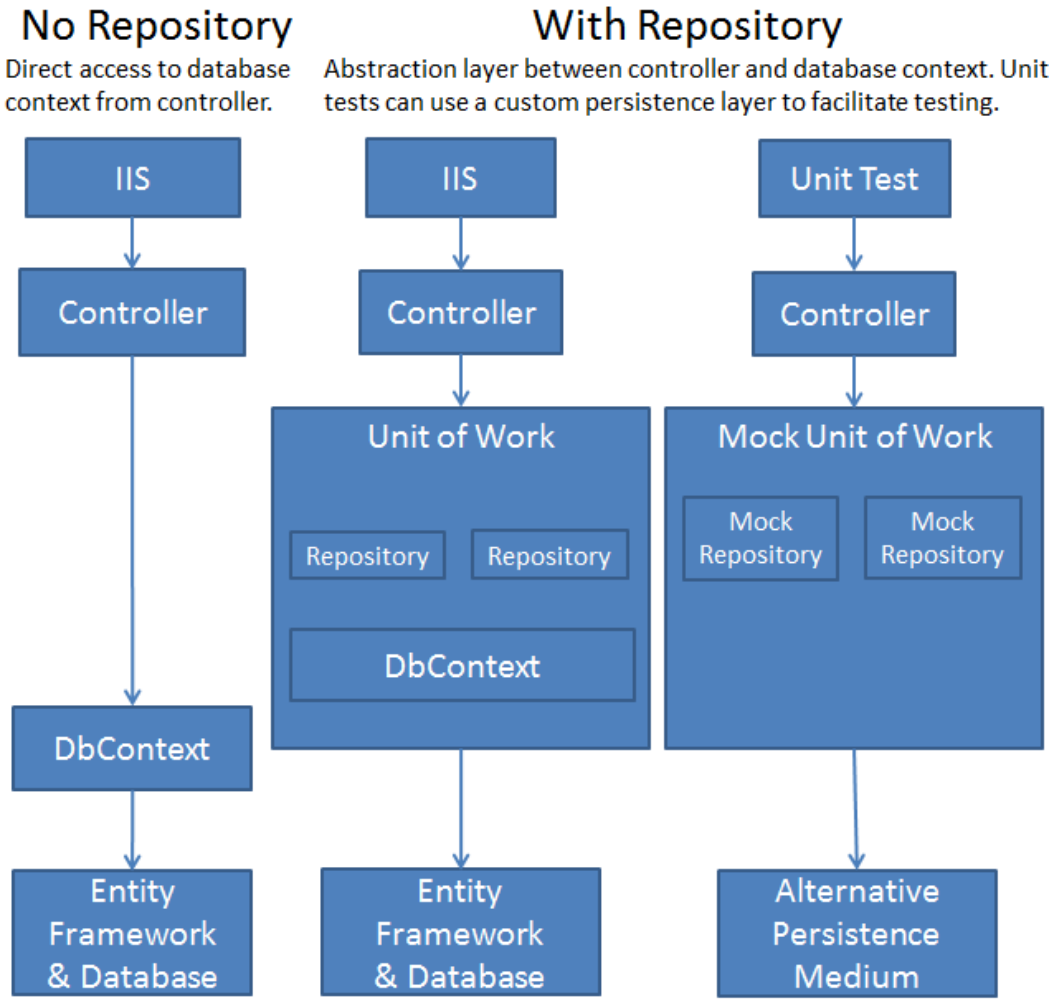
- Es un ***patrón arquitectónico***
- Nos ayuda a estructurar el aplicativo para lograr una buena separación de concerns
- *El patrón repositorio se apoya sobre el **estilo en capas** para estructurar el aplicativo.*
- *Propone crear una **capa de persistencia** para que la misma se encargue del **acceso a los datos***

Patrón Repositorio



Fuente [En línea]: https://www.cosmicpython.com/book/chapter_02_repository.html

Patrón Repositorio



Fuente [En línea]: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

Patrón Repositorio

- *Propone que en la capa de persistencia existan objetos "Repository" .*
- *Cada Repositorio debería poder:*

```
agregar(unObjeto)
modificar(unObjeto)
eliminar(unObjeto)
buscar
buscarTodos
…
```

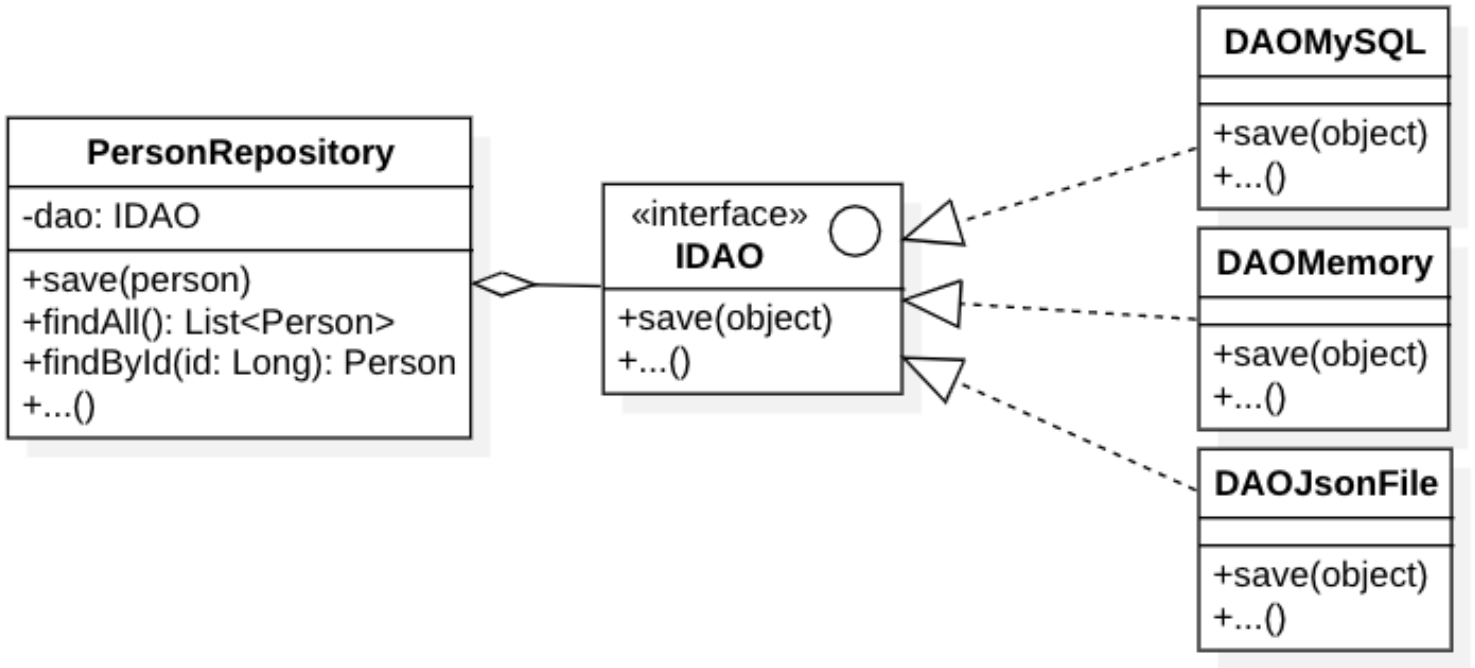
Patrón Repositorio & DAOs

- *El patrón repositorio puede implementarse haciendo uso de los objetos DAOs.*
- *DAO es la abreviatura de Data Access Object*
- *Los DAO son los responsables reales de acceder a los datos. Pueden acceder a una base de datos relacional, a una base de datos no relacional, a la memoria, a archivos, etc.*

Patrón Repositorio & DAOs

- *El patrón repositorio puede combinarse con el patrón de diseño Strategy para utilizar los DAOs.*
- *Cada repositorio podría delegar su responsabilidad en un DAO concreto.*
- *Para clientes de los repositorios, es indistinto el DAO concreto que se utiliza. En otras palabras, los clientes de los repositorios no deberían enterarse cuál es el medio persistente.*

Patrón Repositorio & DAOs



Modelado de Usuarios, roles & permisos – Parte I

Responsabilidades de Capas de un Sistema

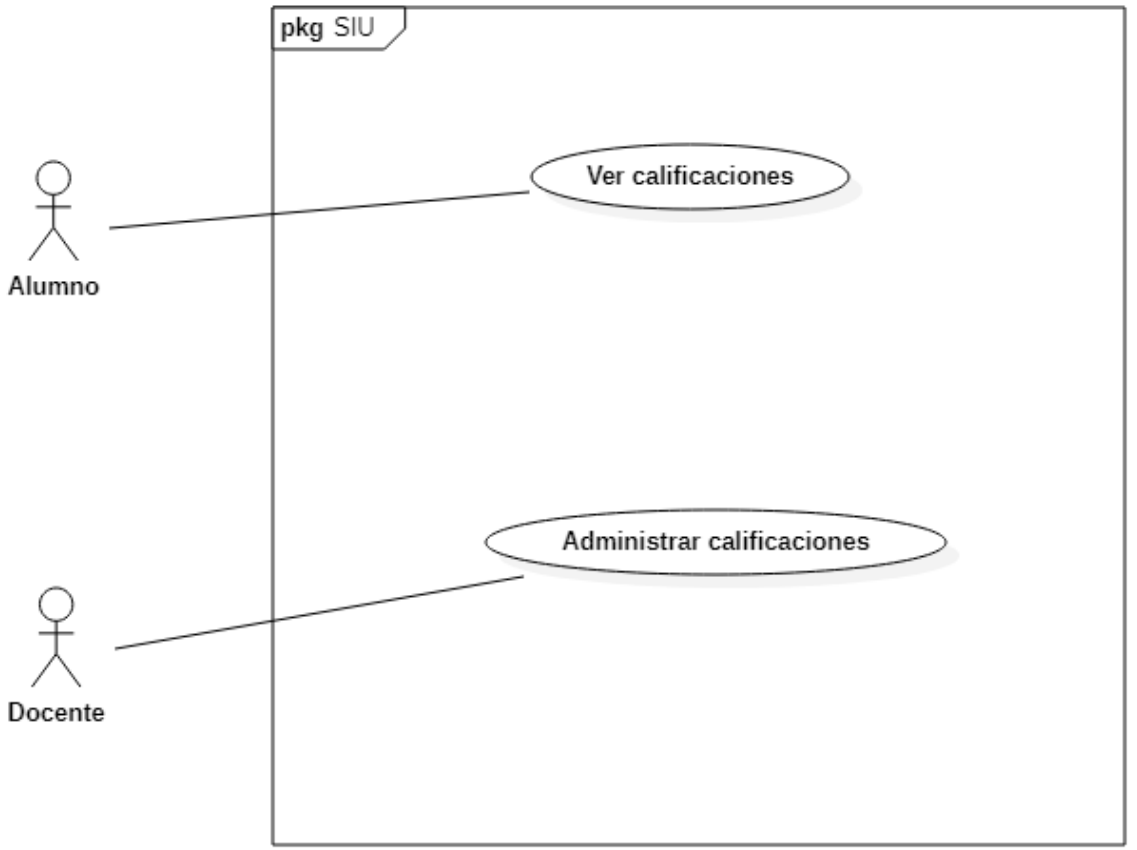
Muchas veces el Paradigma Orientado a Objetos nos puede llevar a confundir la asignación de responsabilidades entre las distintas capas que puede tener un Sistema. Esto lo podemos ver reflejado al querer asignar, de forma incorrecta, métodos que tienen nombre de "*funcionalidades completas*", de "*botones de interfaz de usuario*" o de "*Casos de Uso*", a hipotéticas clases que representan distintos Actores (generalmente una clase por cada posible actor).

Resulta que este diseño es incorrecto porque no debemos mezclar la lógica de negocio con los hipotéticos permisos que tiene un actor/rol en particular sobre nuestro Sistema.

Responsabilidades de Capas de un Sistema

Por ejemplo, si consideramos el Sistema de Gestión Educativa SIU Guaraní, el diagrama de Casos de Uso expuesto y los siguientes requerimientos:

- El Sistema debe permitir que los alumnos visualicen sus calificaciones
- El Sistema debe permitir la administración (alta, baja y modificación) de calificaciones por parte de los docentes



Responsabilidades de Capas de un Sistema

Sería **incorrecto** plantear:

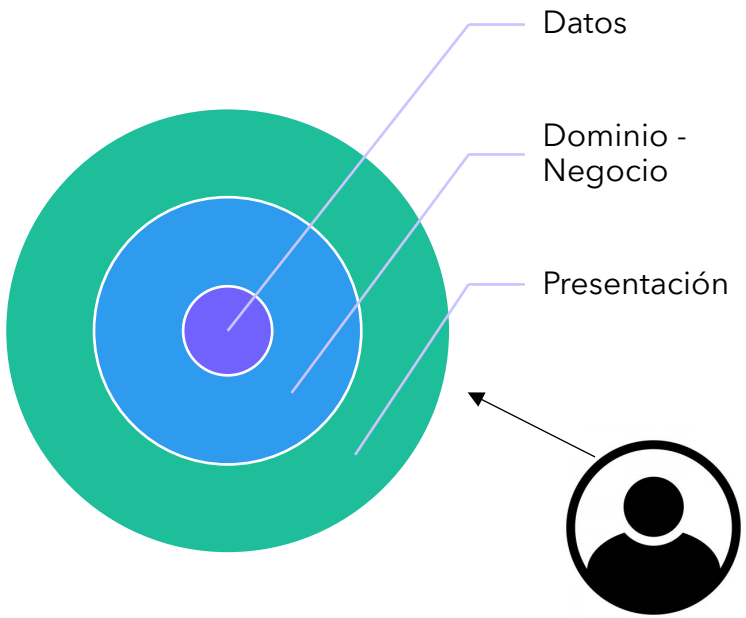
- Una clase `Alumno` con un método "`verCalificaciones`": ¿qué haría ese método?
- Una clase `Docente` con un método "`administrarCalificaciones`": nuevamente, ¿qué haría ese método?

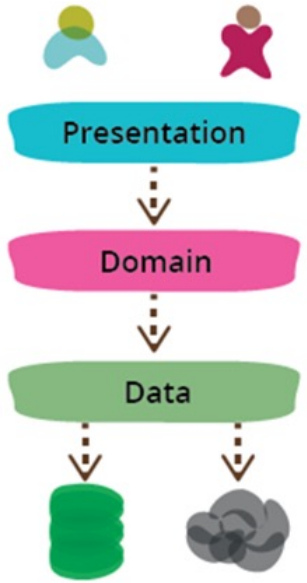
Responsabilidades de Capas de un Sistema

La discusión no pasa por entender que, eventualmente, solamente los docentes pueden administrar las calificaciones y los alumnos visualizarlas, sino que pasa por estar mezclando responsabilidades de distintas capas del Sistema.

Responsabilidades de Capas de un Sistema

Un Sistema puede dividirse, *mínimamente*, en tres capas.





Capa de Presentación

Capa de Presentación

- Es la capa encargada de presentar los datos al usuario, con la cual éste interacturará.

- La presentación de datos no necesariamente es visual, es decir mediante interfaz gráfica, sino que puede ser de distintas formas.

- La forma de presentación de datos la podemos dividir al menos en dos formas:

    - Presentación de datos mediante interfaz gráfica (interfaz desktop, interfaz Web, aplicación móvil, etc.)
    - Presentación de datos mediante APIs (ejemplo, mediante API REST).

Capa de Dominio/Negocio

Capa de Dominio/Negocio

- Es la capa encargada de modelar las reglas de negocio, las entidades del dominio.
- Contiene la parte estructural y de comportamiento que brinda sostén a todos los posibles Casos de Uso del Sistema.

Capa de Datos

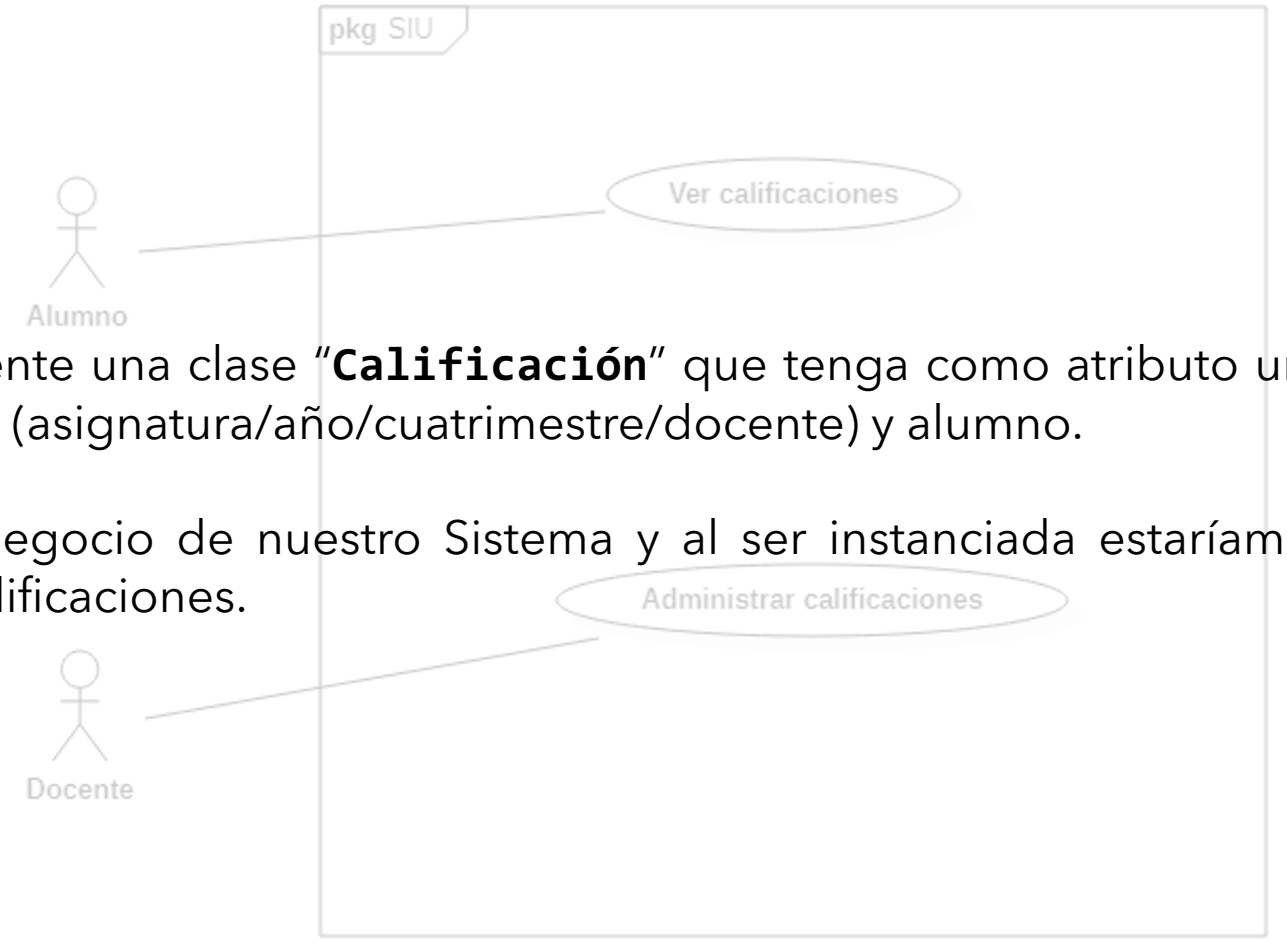
Capa de Datos

- Es la capa encargada de la manipulación de la Persistencia de los datos del Sistema ("*guardado de los datos*").

- Preguntas como *"¿dónde obtengo todos los alumnos del Sistema?"*, *"¿de dónde recupero todos los docentes del Sistema?"*, y similares, corresponden a esta capa.

- Si existe persistencia en una base de datos, esta capa es la encargada de comunicarse con ella.

Responsabilidades de Capas de un Sistema

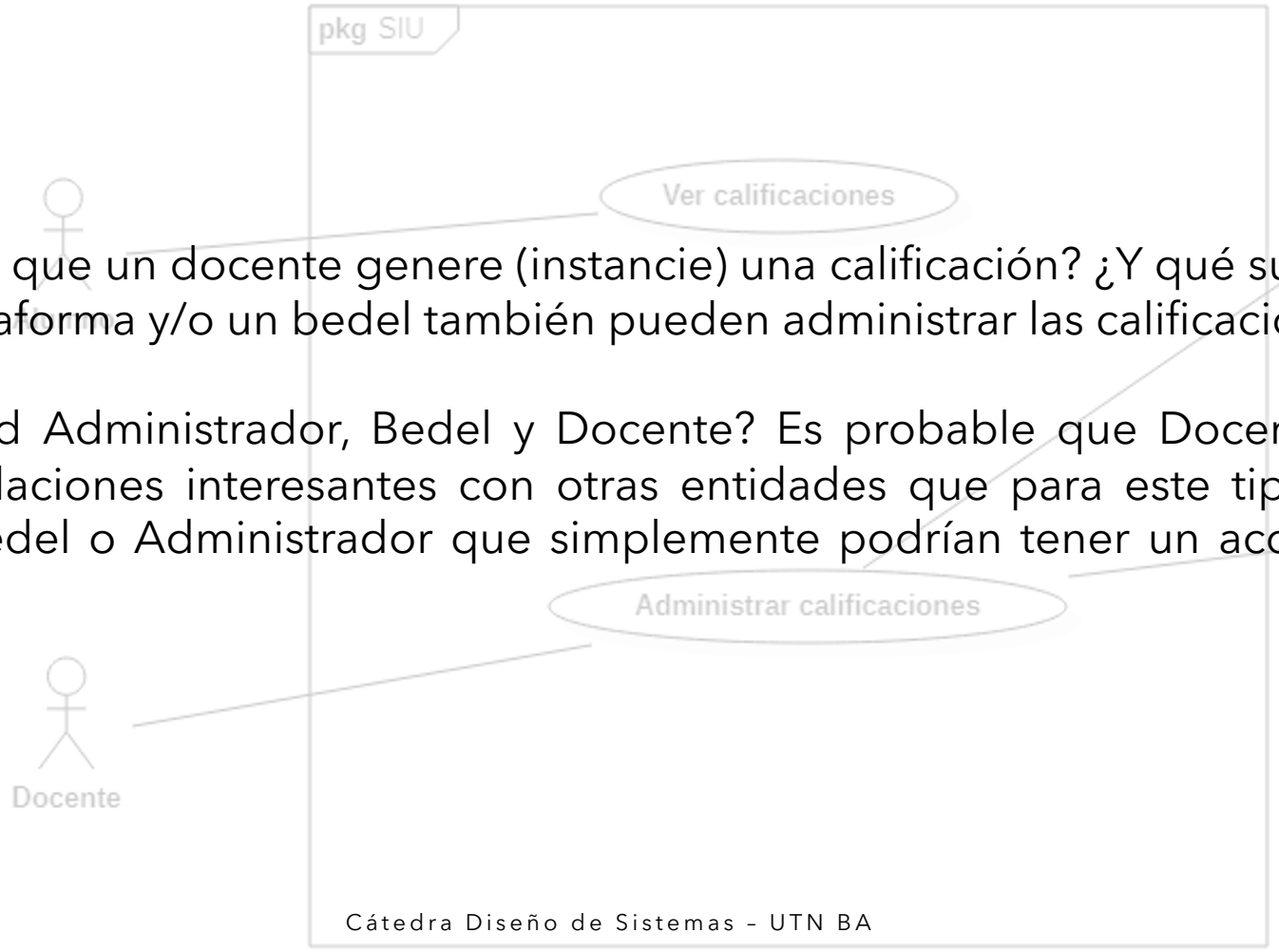


Retomando el ejemplo del SIU…

- En este caso sería correcto plantear, mínimamente una clase "**`Calificación`**" que tenga como atributo una "nota" numérica, por ejemplo, fecha/hora, curso (asignatura/año/cuatrimestre/docente) y alumno.

- Esta clase pertenece a la capa de Dominio/Negocio de nuestro Sistema y al ser instanciada estaríamos permitiendo que "alguien" dé de alta nuevas calificaciones.

Responsabilidades de Capas de un Sistema



- Pero, ¿cómo permitimos que un docente genere (instancie) una calificación? ¿Y qué sucedería si, además, un administrador de la plataforma y/o un bedel también pueden administrar las calificaciones?

- ¿Nos interesa la entidad Administrador, Bedel y Docente? Es probable que Docente sí, pues de ella se desprenden algunas relaciones interesantes con otras entidades que para este tipo de Sistema pueden servir; no así para el Bedel o Administrador que simplemente podrían tener un acceso al Sistema y nada más.

Modelado de Usuarios, roles & permisos

- Llamaremos "***Usuario***" a aquella entidad que ***contiene los datos de Acceso*** para que una persona física u otro Sistema pueda identificarse en nuestro Sistema al usarlo.

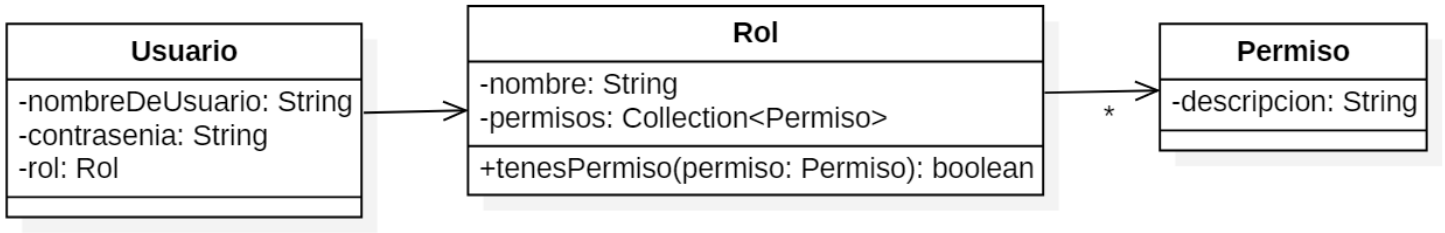
- Los datos de Acceso están compuestos, mínimamente, por nombre de usuario y contraseña.

- Una misma persona física o Sistema podría cumplir varios roles dentro de nuestro Sistema y ejecutar diferentes casos de uso.

- Distintos roles podrían tener la facultad de ejecutar el mismo caso de uso:
  - Cada rol podría ejecutar muchos casos de uso.
  - Para ejecutar un caso de uso se necesita tener uno o varios permisos.
  - Un mismo permiso podría ser adjudicado a varios roles distintos.

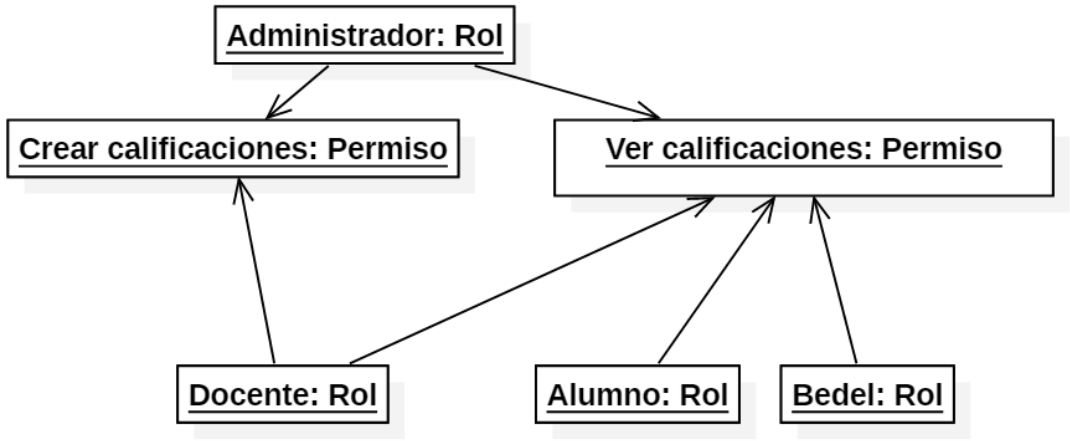
Modelado de Usuarios, roles & permisos

Una propuesta, considerando que un usuario solamente puede tener un único rol, podría ser…



Modelado de Usuarios, roles & permisos

Ejemplos de instancias sobre SIU…



Modelado de Usuarios, roles & permisos

No es la única propuesta de diseño para los Usuarios, roles y permisos:

- Los permisos podrían estar modelados con enumerados, entendiendo que esto trae aparejado ciertas limitaciones.

- Los usuarios podrían tener múltiples roles (colección de roles).

Modelado de Usuarios, roles & permisos

¿Cómo relacionamos nuestra entidad Docente, del ejemplo del SIU, con la entidad Usuario?

- La entidad Docente podría tener como atributo un Usuario.
- Análogamente, un Alumno podría tener como atributo un Usuario.

¿Qué sucedería si un Alumno puede ser Docente al mismo tiempo? (de asignaturas que ya cursó en el pasado)

- Tanto la instancia de la clase Alumno como la instancia de Docente que están representando a esa misma persona, podrían estar referenciando a la misma instancia de Usuario.

Responsabilidades de Capas de un Sistema

Finalmente, ¿dónde podríamos instanciar nuestros objetos? ¿Dónde los podríamos configurar? ¿Dónde podríamos verificar que quien esté intentando realizar esa acción tiene los permisos necesarios para hacerlo?

Las respuestas a estas preguntas es la ***Capa de controladores*** *(o capa de "Servicios" si quisiéramos separar la lógica de interacción con la capa de Presentación)*

Responsabilidades de Capas de un Sistema

Capa de Controladores

- Esta capa se encuentra más "arriba" de la capa de Dominio/Negocio, más precisamente entre ésta y la capa de Presentación.

- En esta capa se suelen agregar métodos que "orquestan" las funcionalidades; además de verificar si el usuario que está intentando realizar dicha acción posee los permisos necesarios.

- En algunos Sistemas se toma la decisión de delegar en otra capa la verificación de Permisos (una capa llamada "*Middleware*").

Responsabilidades de Capas de un Sistema

Retomando el ejemplo del SIU, podríamos tener algo como:

```
public class CalificacionesController {

        public void crearCalificacion(DataCalificacion data, Usuario usuario) {
                Permiso permisoCrearCalificaciones = RepositorioDePermisos.buscar("CREAR_CALIFICACIONES");

                if(!usuario.getRol().tenesPermiso(permisoCrearCalificaciones)) {
                        throw new PermisoInsuficienteException(permisoCrearCalificaciones);
                }

                Calificacion unaCalificacion = new Calificacion();
                //...

                RepositorioDeCalificaciones.guardar(unaCalificacion);
        }
}
```

Responsabilidades de Capas de un Sistema

Algunas consideraciones:

- El objeto “DataCalificacion” podría ser un Value Object que está representando los datos ingresados por el usuario (indistintamente del rol).
- Los repositorios son los objetos encargados de la persistencia de los demás objetos (guardar, buscar, eliminar, entre otros).
- La instanciación y configuración de la calificación podría realizarse mediante la ayuda de algún patrón creacional.

Responsabilidades de Capas de un Sistema

Ubicando gráficamente a nuestros objetos en las distintas capas…

Presentación

- HTML, CSS, JS/Tecnologías de Aplicaciones Nativas/Tecnologías Desktop/APIs

Controladores

- CalificacionesController

Dominio/Negocio

- Calificación
- Alumno
- Docente

Datos

- RepositorioDePermisos
- RepositorioDeCalificaciones

Gracias