Projecte de Programació

Arquitectura 3 capas

Fase de Diseño - Arquitectura 3 capas

En esta etapa hay que empezar a pensar **CÓMO** hay que hacer lo que se ha propuesto en las etapas anteriores

Esta fase es *dependiente* de la tecnología

Partimos del resultado del Análisis de Requisitos y Especificación. En PROP:

- Casos de Uso
- Modelo conceptual de los datos (versión Especificación)

Fase de Diseño - Arquitectura 3 capas

La salida de la fase de diseño en PROP:

- Arquitectura de la aplicación
- Modelo conceptual de datos (versión Diseño)
- Estructuras de Datos
- Algoritmos

Predeterminado en PROP:

- Lenguaje de programación Java
- Persistencia mediante ficheros Java
- Modelo arquitectónico de Arquitectura en 3 capas (+ Domain Model en la capa de dominio)

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas

La arquitectura del *software* describe los sistemas y componentes computacionales del *software*, y las relaciones entre ellos

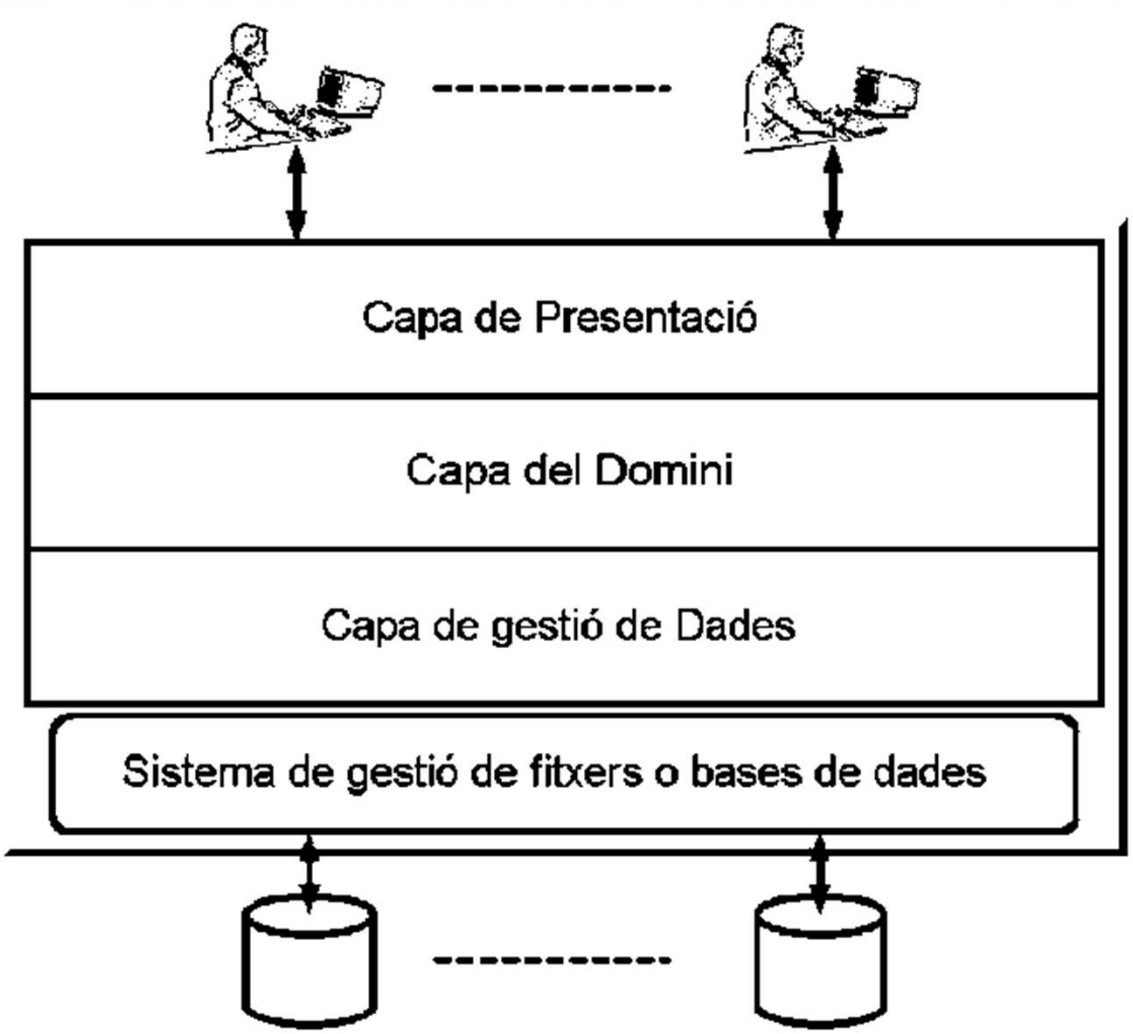
Existen diferentes *patrones arquitectónicos* en los que podemos basarnos para diseñar nuestro sistema. En PROP usaremos la **arquitectura en tres capas**.

Consta de:
- Capa de *Presentación*
- Capa de *Dominio*
- Capa de *Gestión de Datos* (o de *Persistencia*)

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas

Esencialmente es:



Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas

Capa de Presentación: Responsable de la interacción con el usuario

- Presenta los datos al usuario, pero no sabe hacer las transformaciones necesarias para poder obtener los resultados que el usuario quiere

- Se relaciona con los usuarios capturando los acontecimientos y presentando respuestas y resultados

- Se relaciona con la capa de dominio pasándole las peticiones externas y recogiendo los resultados que hay que presentar

- Gestiona la interficie de comunicación con el usuario (incluye comprobación de sintaxis de los datos, NO de sus valores)

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas

Capa de Dominio: Contiene el núcleo del programa. Es la capa que sabe transformar y manipular los datos del usuario en los resultados que espera (la capa no "sabe", sin embargo, ni de dónde vienen estos datos ni dónde están almacenados)

- Se relaciona con la capa de presentación, recibiendo las peticiones externas y retornando los resultados deseados

- Se relaciona con la capa de persistencia, pasando las operaciones de consulta y modificación de los datos que están en memoria secundaria, y recibiendo las respuestas y resultados

- En general, el estado del dominio se mantiene y cambia en esta capa, y aquí se chequea la validez de las peticiones de los usuarios

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas

Capa de Persistencia: Esta capa es la responsable de almacenar los datos (sabe dónde y cómo están almacenados). Sin embargo, ignora cómo tratarlos

- Se relaciona con la capa de dominio recibiendo las operaciones de consulta y actualización de los datos en memoria secundaria, y retornando respuestas y resultados de estas peticiones

- Se relaciona con el sistema de gestión de bases de datos o ficheros pasando las operaciones de consulta y/o actualización de los datos en el formato y lenguaje adecuados y recibiendo las respuestas y resultados

- Permite que determinados objetos del dominio sean persistentes y que el dominio ignore dónde se encuentran los datos

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Cada capa tiene responsabilidades cualitativamente diferentes:

- Presentación
  - Interacción con el usuario
  - Control de las peticiones del usuario
  - *Comunicación con la capa de dominio*

- Dominio
  - Mantenimiento básico de los datos (clases definidas en la especificación)
  - Implementación de una parte de las funcionalidades principales (casos de uso) que corresponde a la capa de dominio: cálculos, modificación del estado del sistema, etc.
  - *Comunicación con las capas de presentación y persistencia*

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

La tarea de comunicación entre capas es parte esencial de las responsabilidades de cada capa, principalmente en las capas de Presentación y Dominio

Por ello las clases que forman parte de cada capa se pueden dividir, a grandes rasgos, en:

- Capa de Presentación: Vistas y ***controladores***

- Capa de Dominio: Clases del modelo y ***controladores***

Así pues, qué son los controladores?

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Los controladores son clases propias de cada capa que tendrán la *responsabilidad* de:

- **Comunicación entre capas**, manteniendo la sincronización entre capas adyacentes

- Implementar los casos de uso y/o **aglutinar funcionalidades de los casos de uso** relacionados

Además, la presencia de controladores dentro de las capas permite **organizar mejor el código**, pensando en la reutilización

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Controladores de la capa de Presentación :

- Permiten separar el aspecto externo (vista) de los métodos que gestionan el comportamiento de la interfície (controladores)

- Ligados a los casos de uso, es habitual agrupar los casos de uso de una misma familia en el mismo controlador (ej: ABMC de una clase) -> REUTILIZACIÓN (al menos de la estructura del código)

Controladores de la capa de Persistencia :

- No son estrictamente necesarios, pero podrían encargarse de la conversión entre formatos diferentes de la información, por ejemplo.

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Controladores de la capa de Dominio:

- Permiten descargar de funcionalidades a las clases del modelo, que pueden encargarse más de la gestión de los datos

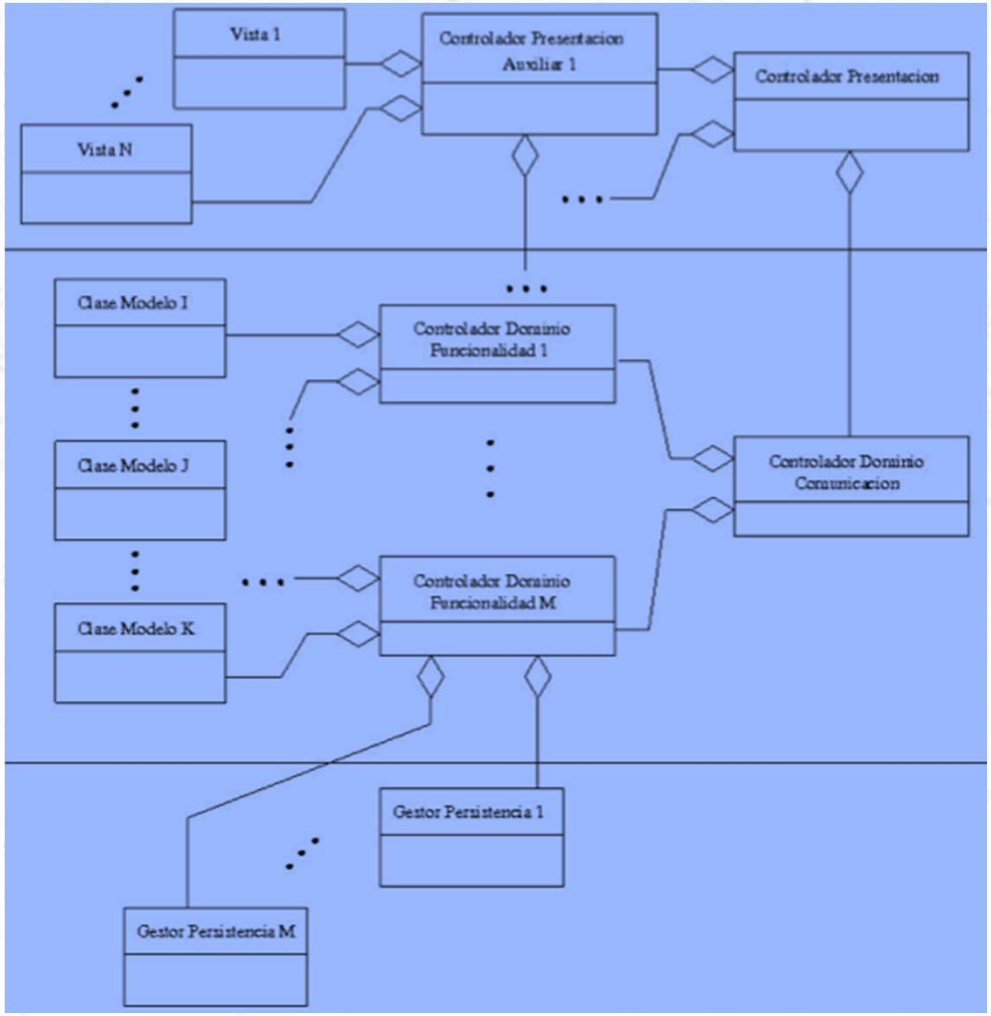
- Permiten que las funcionalidades concretas de la aplicación (los algoritmos ligados a los casos de uso) se implementen dentro de los controladores -> REUTILIZACIÓN (de algoritmos y clases del modelo)

Puede haber uno o más controladores por capa, dependiendo de la **granularidad** que se quiera tener

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Esquema general:



Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

- Este esquema general es una versión **muy GENERAL**: No hace falta que tengamos tantos controladores

- En particular, necesitaremos Controlador Dominio Comunicación si hay mucha carga de ello en el proyecto. Si no, Controlador de Presentación puede usar directamente los controladores de dominio

- Puede haber uno o más controladores por capa

- La manera de distribuir los controladores en cada capa **NO es única**, pero suele ir guiada por los casos de uso: un controlador de la capa de dominio integrará tantas clases del modelo como necesite, junto con los gestores de disco correspondientes, para implementar los casos de uso asociados

- Suele haber un gestor de persistencia por cada clase persistente del modelo (aunque se pueden agrupar varias clases en uno)

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Controladores

Así pues, un programa principal (aquel que pone en marcha la aplicación, *usualmente el controlador de presentación*) podría seguir un esquema similar a:

- El controlador de presentación, al crearse/inicializarse, crea una instancia de controlador de dominio y del resto de controladores de presentación (si hay)

- El controlador de dominio, al crearse/inicializarse, crea una instancia del resto de controladores de dominio (si hay)

- El resto de controladores de presentación crean una instancia de su(s) vista(s) asociada(s)

- El resto de controladores de dominio crean las clases del modelo que necesiten –> si estas clases son persistentes, también los Gestores de Disco asociados para cargar su contenido desde memoria secundaria

No es necesario hacerlo todo al principio, puede hacerse bajo demanda

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: versión PROP

- La comunicación (relaciones entre clases y mensajes entre objetos) sólo se produce **entre elementos de la misma capa o entre capas adyacentes**.

- Si es entre capas adyacentes, la comunicación se ha de hacer vía controladores. Única excepción: un controlador de presentación puede pasar un controlador de dominio como parámetro a una vista para que ésta trabaje directamente con él (**sólo** en casos **muy excepcionales** de interacción continuada: juegos, …)

- La comunicación entre capas, al realizarse entre controladores, se hace vía tipos de datos generales (por ejemplo, vectores de *Strings*), NUNCA vía instancias de datos del modelo -> puede ser una buena idea añadir en las clases del modelo 2 operaciones:
  - *vector<String> toString()*
  - *constructor (vector<String>)*

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: versión PROP

- El esquema conceptual de los datos se implementa en la capa de dominio: **patrón *DOMAIN MODEL*** (versus patrón *Transaction Script*)

- Para la comunicación entre las capas de dominio y gestión de datos, se relaja la restricción, permitiendo que una operación de la capa de gestión de datos reciba un objeto del modelo de datos para guardarlo o retorne un objeto del modelo al cargarlo. Sería equivalente a implementar la interficie *Serializable* en las clases del modelo y proporcionar las operaciones *writeObject* y *readObject*

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Ventajas

- *Intercambiabilidad*: Un cambio en la interfície del programa no afectará al resto, sólo a la capa de presentación. Un cambio en algoritmos o estructuras de datos sólo afectará a la capa de dominio. Un cambio en la representación de los datos (cambio en el SGBD, o entre BD y ficheros) sólo afectará a la capa de persistencia

- *Reusabilidad*. Cualquier capa se puede reusar "fácilmente"

- *Portabilidad*. La capa de dominio, que encapsula la lógica del programa, es bastante independiente de cambios de plataforma, sistema operativo, etc.

Fase de Diseño - Arquitectura 3 capas

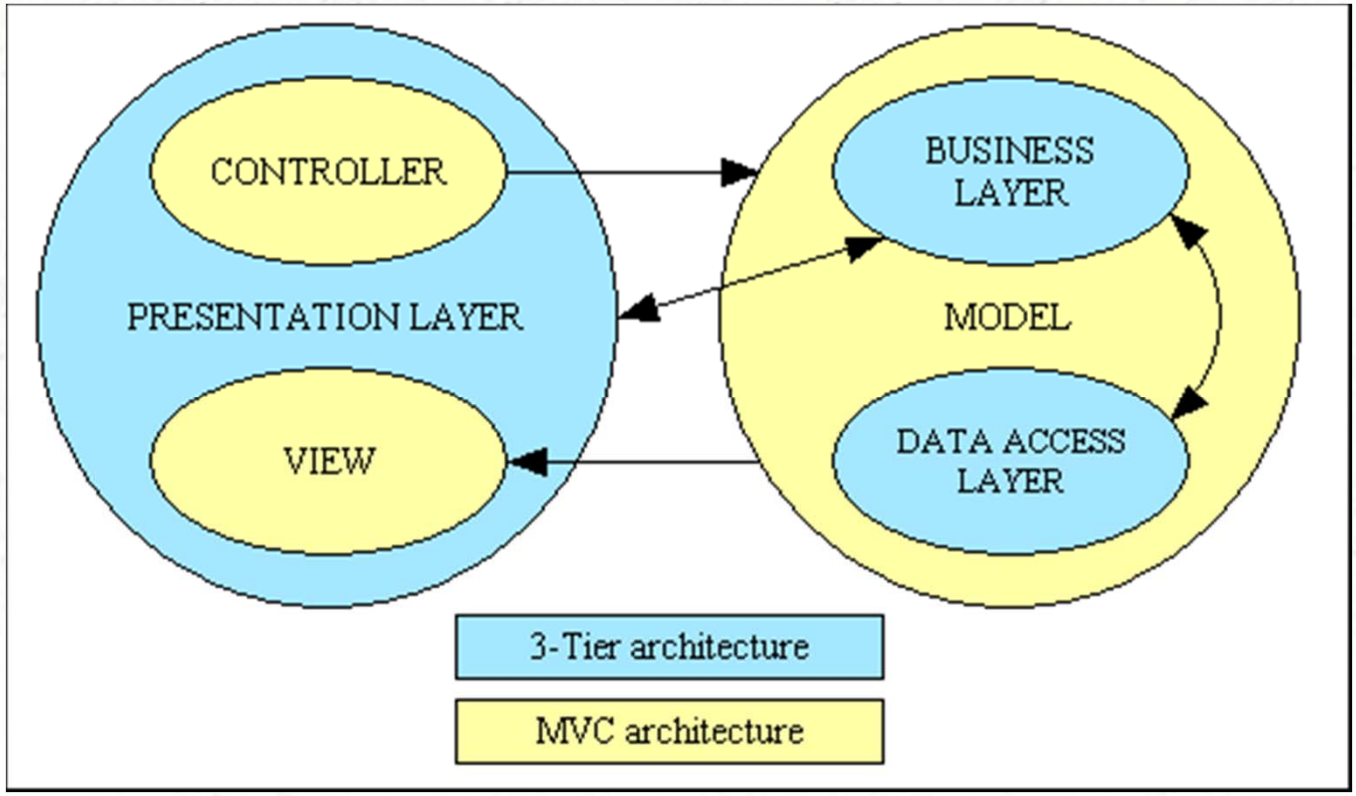
Arquitectura en tres capas: Inconvenientes

- *Poco eficiente*: Demasiados mensajes, estructura rígida

- *Redundante*: A veces se hace lo mismo (o similar) en diferentes capas. Por ejemplo, validar que un campo está dentro del rango correcto

Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas y MVC



(figura 8 de http://www.tonymarston.net/php-mysql/3-tier-architecture.html)

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (1)

Queremos implementar el *login* de un usuario. Una traducción algorítmica directa podría ser*:

```
void loginUsuario() {
  (user,pass) = pedirDatosLoginUsuario(); // petición de datos al usuario
  if (!loginUsuarioCorrecto(user,pass))
    mensajeError("Login incorrecto");
  else
    informarDatosUsuario(user);
}

boolean loginUsuarioCorrecto (user,pass) {
  f = abrirFicheroUsuarios();
  fuser = "";
  while (!finFichero(f) && (fuser != user))
    (fuser,fpass) = leerFichero(f); // acceso al fichero de claves
  if (fuser == user && fpass == pass) return true;
  return false;
}
```

*Hay licencias de pseudocódigo: p.ej., Java no permite retornar más de 1 objeto

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (2)

```
void informarDatosUsuario (user) {
  datos = obtenerDatosUsuarioPersist(user); // acceso a ficheros del usuario
  informarDatosUsuarioMemoria(datos); // modifica objetos en memoria
}

datos  obtenerDatosUsuarioPersist (user) {
  ... leer los fichero asociados a user y guardarlos en datos
}

void informarDatosUsuarioMemoria (datos) {
  ... modificar los objetos en memoria con la información que hay en datos
}
```

Esta manera de pensar NO es compatible con la Arquitectura en 3 capas: mezcla la petición de datos al usuario (capa de presentación) con el acceso a los ficheros de datos (capa de persistencia) y con la modificación de los objetos en memoria (capa de dominio).

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (3)

CÓMO SE IMPLEMENTARIA EN ARQUITECTURA EN 3 CAPAS?
- Asignar cada tarea a cada una de las capas
- Distribuir los métodos implicados en las diferentes clases
- Usar los controladores para pasar la información entre capas

Por ejemplo, en el Controlador de Presentación tendríamos:

```
private void loginUsuario() {
  (user,pass) = VISTALOGIN.pedirDatosLoginUsuario();
  if (!CTRLDOMINIO.loginUsuarioCorrecto(user,pass))
    VISTALOGIN.mensajeError("Login incorrecto");
  else
    CTRLDOMINIO.informarDatosUsuario(user);
}
```

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (4)

En la Vista de Login tendríamos:

```
public (u,p) pedirDatosLoginUsuario() {
  user = readString();
  pass = readString();
  return (user,pass);
}

public void mensajeError (String mensaje) {
  writeString(mensaje);
}
```

En el Controlador de Dominio tendríamos:

```
public boolean loginUsuarioCorrecto (String user, String pass) {
  fpass = GESTORUSUARIO.getPasswordUsuario(user);
  if (fpass == pass) return true;
  return false;
}
```

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (5)

En el Controlador de Dominio tendríamos:

```
public void informarDatosUsuario (String user) {
  Estructura datos = GESTORUSUARIO.obtenerDatosUsuario(user);
  informarUsuario(datos);
  para (cada ClaseDominio relacionada con la clase Usuario) {
    datos = GESTORCLASEDOMINIO.obtenerDatosClaseDominio(user);
    informarClaseDominio(datos);
  }
}

private void informarUsuario (Estructura datos) {
  crear instancias de Usuario e informar via setAtributo
  ...
}

private void informarClaseDominio (Estructura datos) {
  crear instancias de ClaseDominio e informar via setAtributo
  ...
}
```

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (6)

En el Gestor de Usuario tendríamos:

```
public String getPasswordUsuario (String user) {
  f = abrirFicheroUsuarios();
  fuser = "";
  while (!finFichero(f) && (fuser != user))
    (fuser,fpass) = leerFichero(f);
  if (fuser == user) return fpass;
  return "";
}
public Estructura obtenerDatosUsuario (String user) {
  f = abrirFicheroUsuarios();
  buscar y devolver los datos del usuario user
}
```

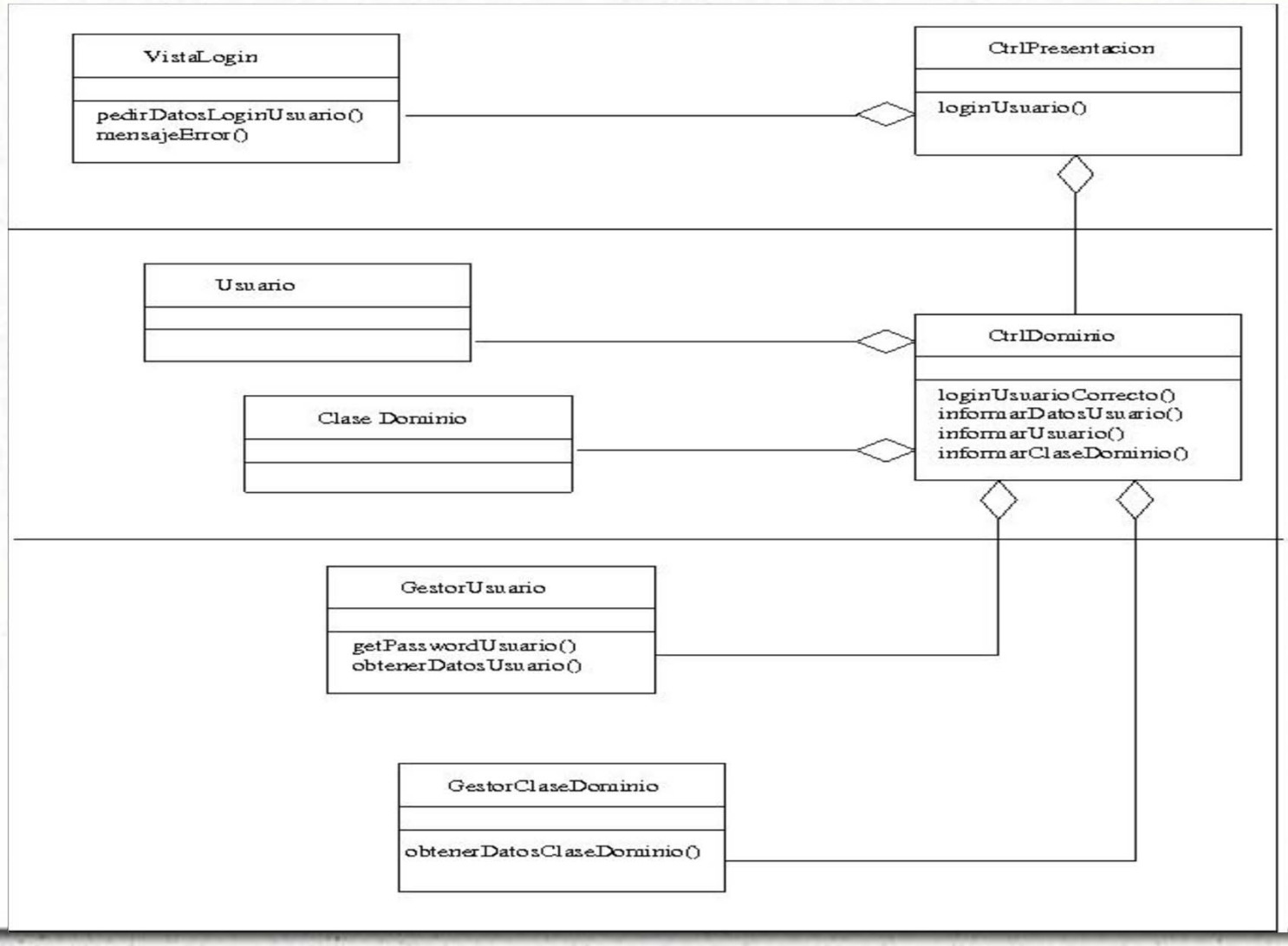
Y en cada Gestor de ClaseDominio tedríamos:

```
public Estructura obtenerDatosClaseDominio (String user) {
  f = abrirFicheroClaseDominio();
  buscar y devolver los datos relacionado con el usuario user
}
```

Fase de Diseño - Arquitectura 3 capas

Ejemplo traducción de caso de uso a Arquitectura en tres capas (7)

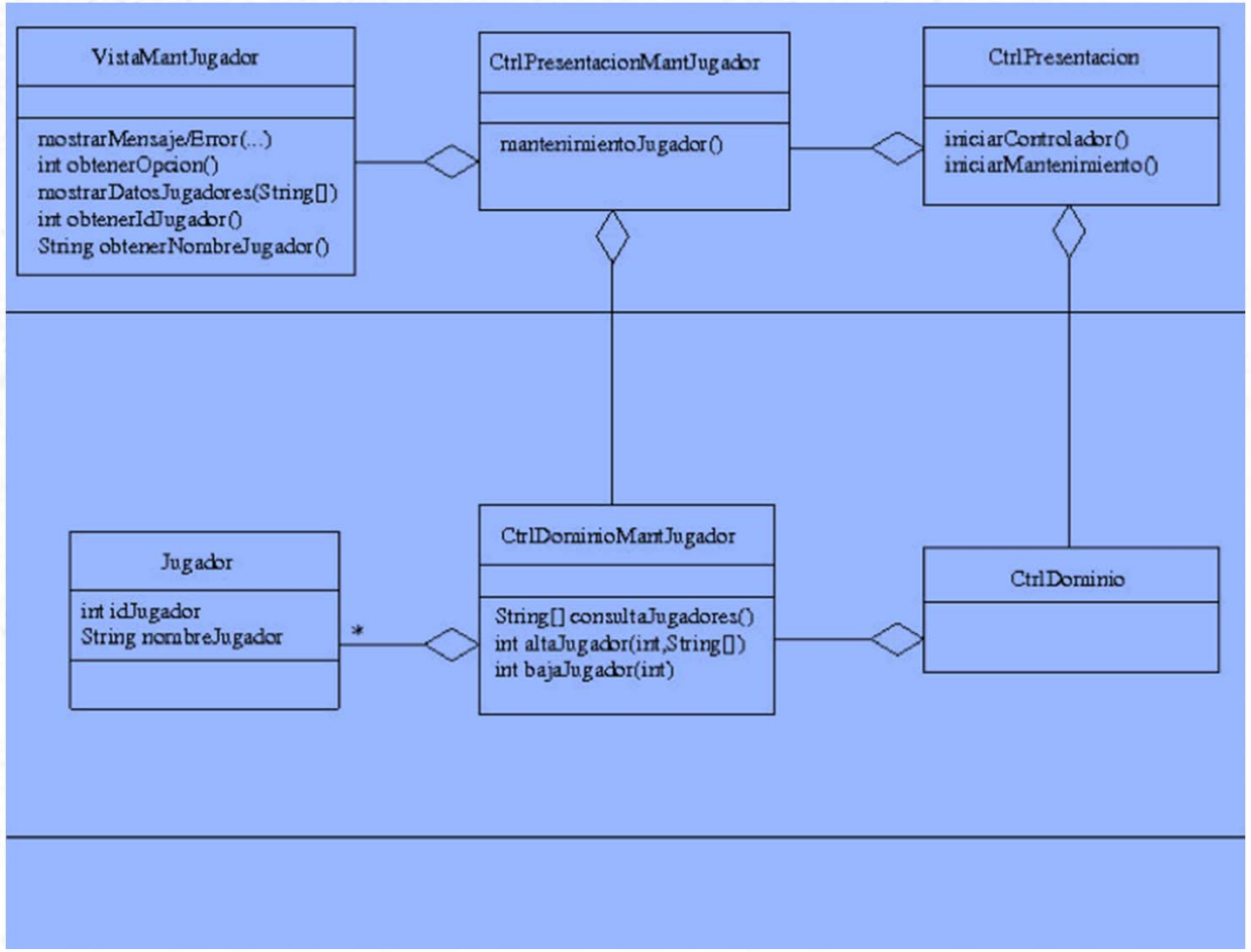
El diagrama resultante sería algo así:



Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Ejemplos

Mantenimiento Jugador



Fase de Diseño - Arquitectura 3 capas

Arquitectura en tres capas: Ejemplos

Mantenimiento Genérico