

# Desarrollo de Aplicaciones Web

Arquitecturas Web

# Agenda

- Concepto
- Diseño de la arquitectura
  - Planear las capas de la aplicación
  - Diseñar una aplicación distribuida

# Diseño de la arquitectura de la aplicación

- La arquitectura de la solución debe satisfacer completamente los requerimientos además de crear una aplicación de alto rendimiento
- Consideraciones para construcción y despliegue de la aplicación
  - El despliegue se puede hacer a través de múltiples máquinas físicas
  - Los datos deben ser almacenados en una base de datos o el cliente necesita accederlos con regularidad en el servidor
  - La aplicación necesita ser distribuida en una granja de servidores, disponibilidad el 99.99999%
  - El servidor sirve miles de páginas y soporta cientos de usuarios concurrentes.

# Definición

“La arquitectura del software de un programa o un sistema de computación es la estructura o estructuras del sistema, las cuales comprenden componentes software, las propiedades visibles de estos componentes y las relaciones entre ellos”

Bass, Clements, and Kazman. Software Architecture in Practice, AddisonWesley 1997.

“Es la descripción de los subsistemas y componentes de un sistema software y las relaciones entre ellos, típicamente representado mediante vistas que muestran las propiedades funcionales y no funcionales más relevantes.

Frank Buschman, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture – A System of Patterns; John Wiley & Sons Ltd. Chichester, England, 1996

# Aspectos generales

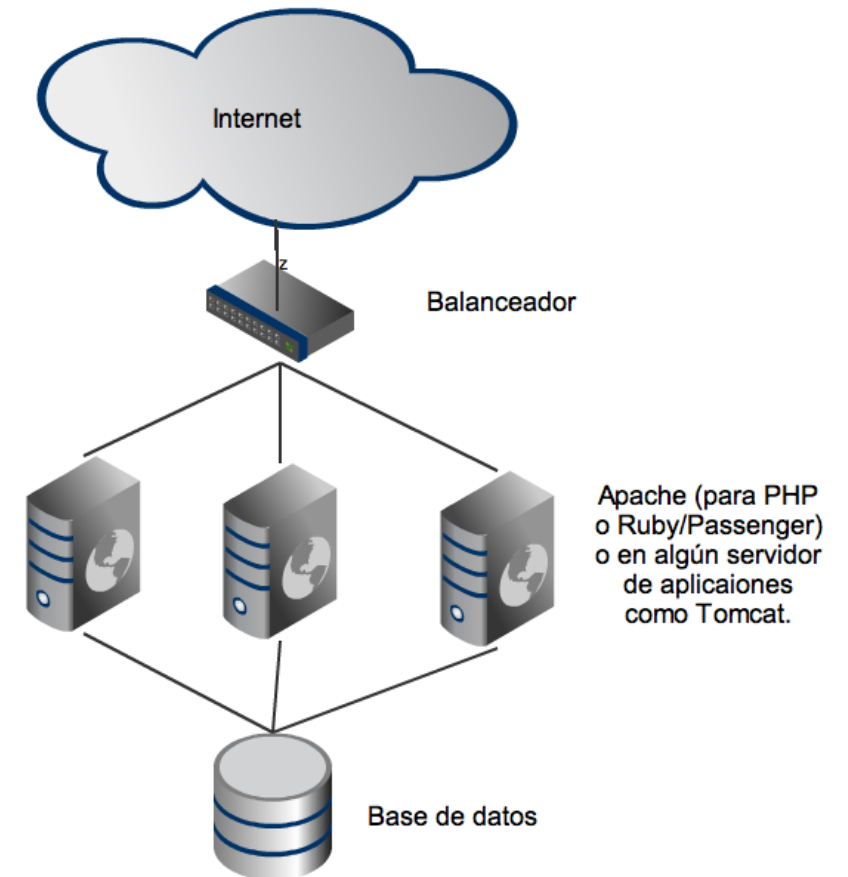
## Escalabilidad

- Posible incremento vertiginoso del número de usuarios
- Se debe considerar
  - Dimensionamiento correcto de la aplicación
  - Adaptabilidad del sistema ante el incremento de demanda.
- Opciones
  - Escalabilidad horizontal (Bases de Datos NoSQL)
  - Escalabilidad vertical
  - Clúster de servidores

# Aspectos generales

## Escalabilidad - Horizontal

- Balanceador por hardware
  - Clonar el sistema y balancear la carga
  - Distribución por algoritmos predeterminados para las peticiones HTTP entre los distintos clones del sistema
  - El clon es aleatorio y no se garantiza que diferentes peticiones de un usuario sean servidas por el mismo sistema
- Investigar: algoritmos



# Aspectos generales

## Escalabilidad - Horizontal

- Balanceador por software
  - Examinan el paquete a nivel del protocolo
  - HTTP para garantizar el mantenimiento de la sesión de usuario.
  - Distintas peticiones del mismo usuario son servidas por el mismo clon del servidor.
  - Más lentos que los balanceadores Hardware
  - Normalmente son soluciones baratas. Ej., módulo mod\_jk de apache
- Balanceador Hardware HTTP
  - Hardware que examinan la petición a nivel de paquete HTTP.
  - Término medio entre las dos anteriores.
  - Garantizan el mantenimiento de sesión.
  - Más rápidos que los de Software pero menos que los de Hardware

# Aspectos generales

## Escalabilidad - Vertical

- Separación lógica entre capas permite implementar la separación física de las mismas
- Middleware entre las capas para permitir la comunicación remota



# Aspectos generales

## Escalabilidad - Clusters de Servidores

- Habituales en los servidores de aplicaciones comerciales
- Dependiendo de su implementación son horizontales o verticales
- Distribuye y escala el sistema de modo transparente a usuario y administrador
- Garantiza que sea cual sea la máquina que sirva la petición http tendrá acceso a la sesión del usuario (Replicación de sesión)
- La replicación de sesión es MUY costosa, produce bajo rendimiento del sistema.

# Aspectos generales

## Separación de responsabilidades

- Premisa base para la separación de capas
- Distintas Responsabilidades no deben ser delegadas en la misma clase (independencia)
- Arquitectura de n-capas
- Arquitectura de 3 capas como modelo básico
  - Capa de presentación
  - Capa de negocio
  - Capa de persistencia

# Aspectos generales

## Portabilidad

- Una aplicación web debe poder adaptarse a las distintas arquitecturas físicas posibles en el despliegue.
- Las tareas de adaptación a un nuevo entorno deben limitarse al ámbito de la configuración, no del desarrollo.
  - *Cliente reacio a las tecnologías de componentes J2EE (EJBs) por costes, rendimiento o simplemente, moda.*

# Diseño de la arquitectura de la aplicación

## Planear las capas de la aplicación

- Aplicación
  - Conjunto de funcionalidades
  - Pantalla o conjunto de pantallas que muestran información o la hacen persistente o permiten tomar decisiones de negocio
- Capa
  - Agrupación lógica de código que trabaja bajo una finalidad común.
  - Las capas trabajan en conjunto para producir la aplicación completa.

# Diseño de la arquitectura de la aplicación

## Planear las capas de la aplicación

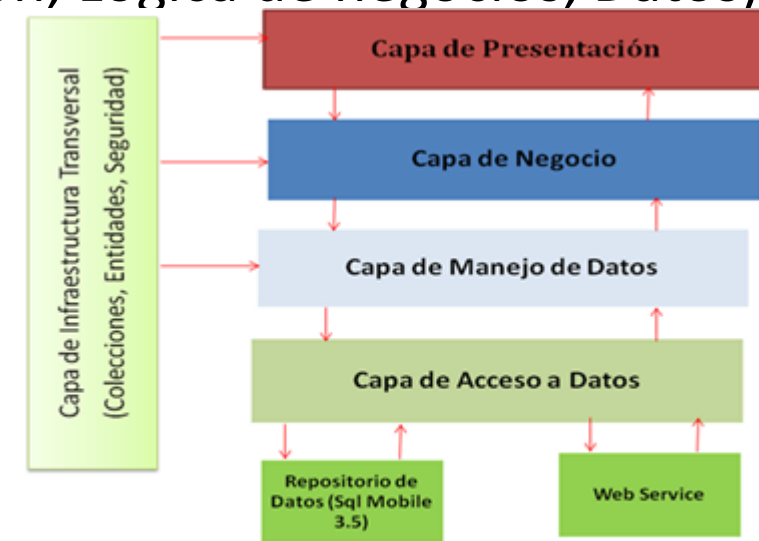
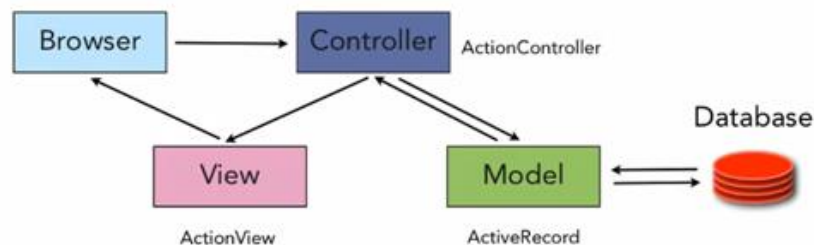
- Aspectos de la arquitectura de la aplicación que contribuyen con sus capas
  - Método de acceso a datos
  - Separación de responsabilidades

# Planear las capas de la aplicación

## Separación de responsabilidades

- Separar las parte de un programa en diferentes secciones o con diferentes responsabilidades y cada sección tiene un propósito diferente.
- Ejemplo
  - Desarrollo basado en N-Capas (Presentación, Lógica de negocios, Datos)
  - Patrón Modelo Vista Controlador (MVC)

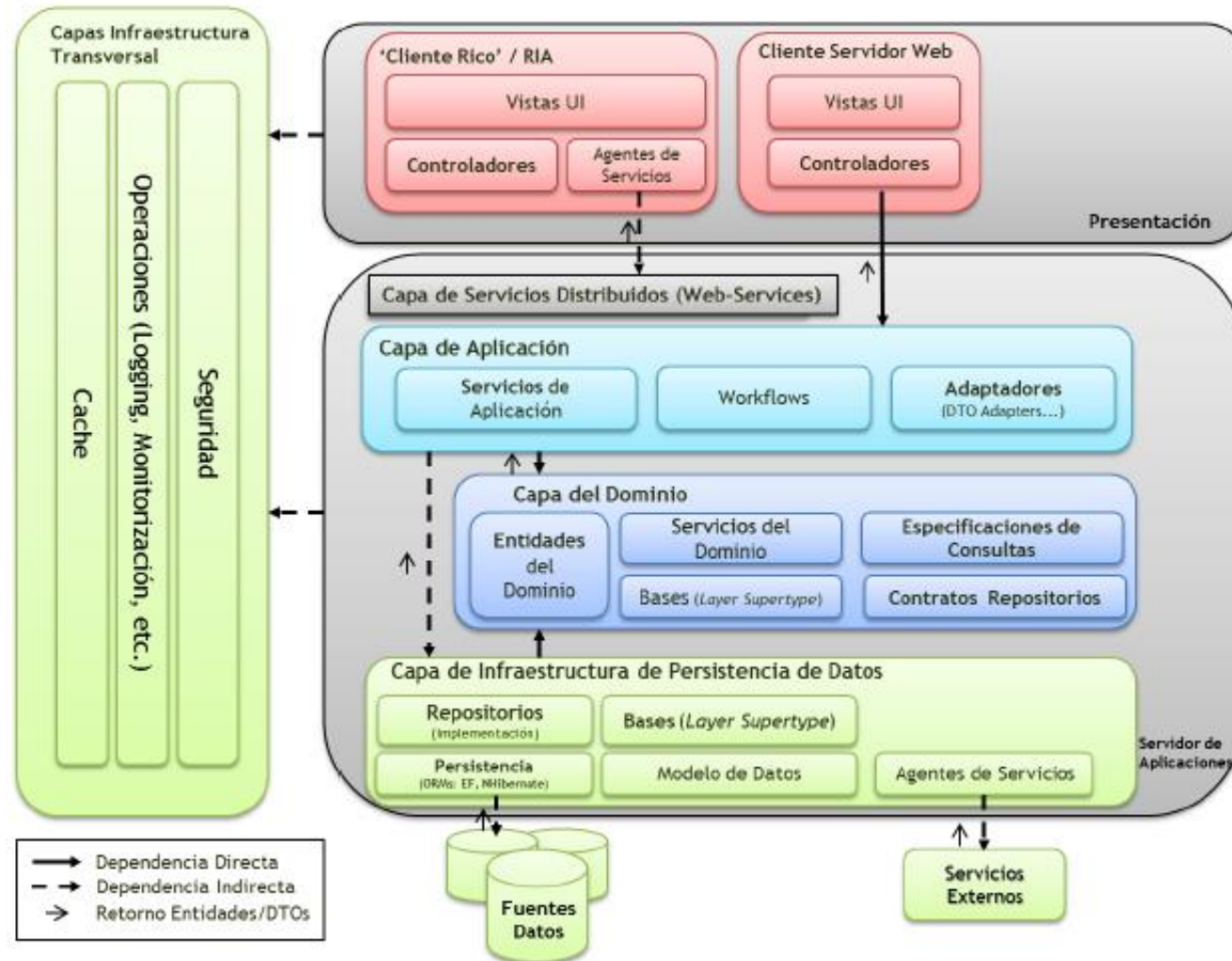
MVC WEB ARCHITECTURE



# Diseño de la arquitectura de la aplicación

## Planear las capas de la aplicación

### *Arquitectura N-Capas con Orientación al Dominio*



# Diseño de la arquitectura de la aplicación

## Separación de responsabilidades

- Evolución
  - Aplicaciones Mainframe
    - Única capa física y lógica
  - Aplicaciones Cliente – Servidor
    - Cliente: capas presentación y negocio
    - Servidor: capas de negocio y acceso a datos
  - Aplicaciones Web basadas en transaction scripts (CGI)
    - Presentacion: Interface Web + lenguajes scripting
    - Negocio y acceso a datos
  - Aplicaciones de 3 capas



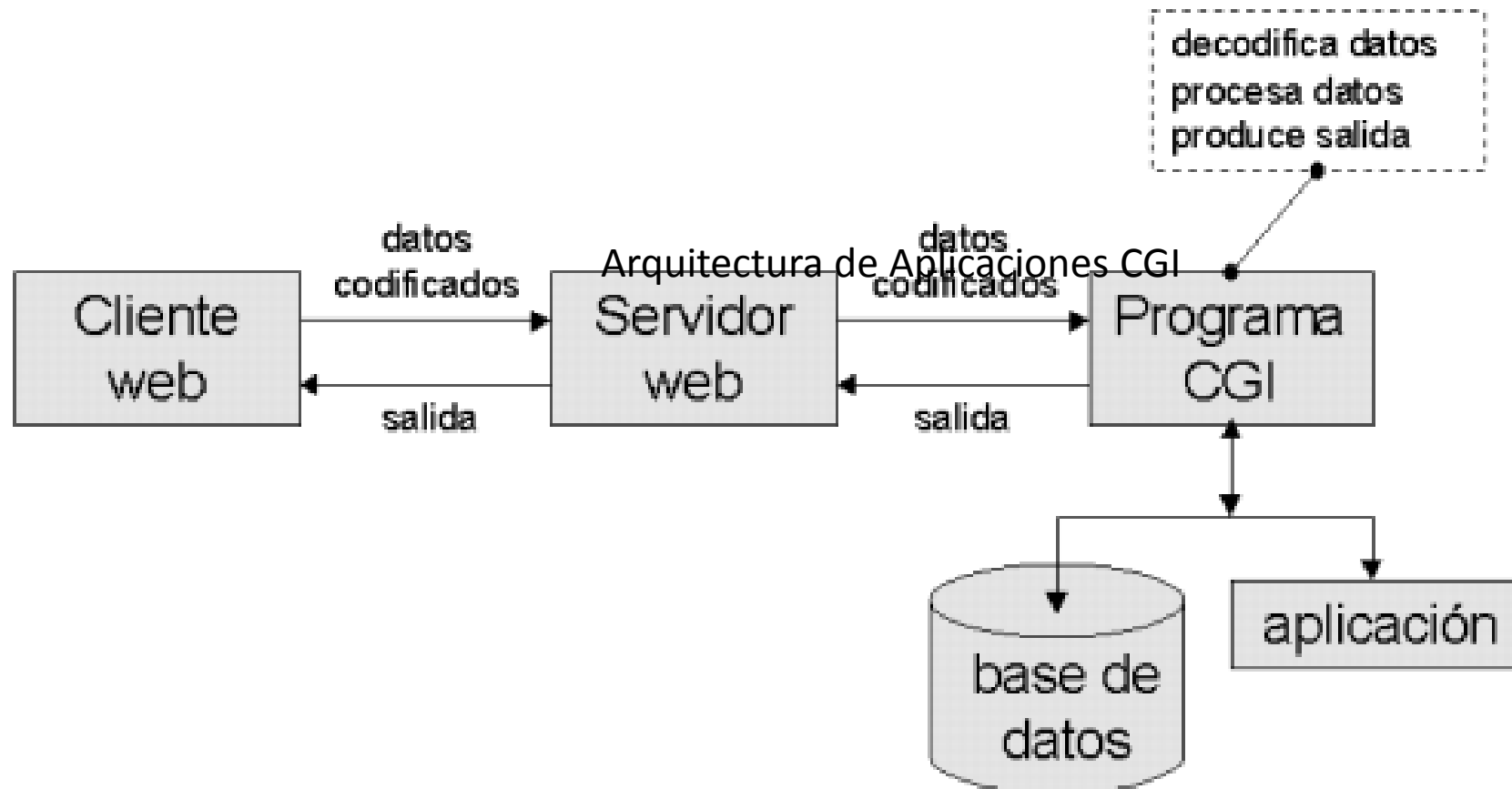
# Diseño de la arquitectura de la aplicación

## Aplicaciones CGI

- Las aplicaciones mas primitivas del WWW
- Aplicaciones Web Common Gateway Interface
- Permite al navegador solicitar datos de un programa ejecutado en un servidor web
  - El servidor recibe la solicitud del cliente, comprueba si es una petición CGI y la pasa a un programa externo
  - El programa puede estar escrito en cualquier lenguaje soportado por el servidor
  - La respuesta del programa se envía al cliente en lugar del archivo estático tradicional (MIME)
- Investigar: MIME

# Diseño de la arquitectura de la aplicación

## Arquitectura de Aplicaciones CGI



# Diseño de la arquitectura de la aplicación

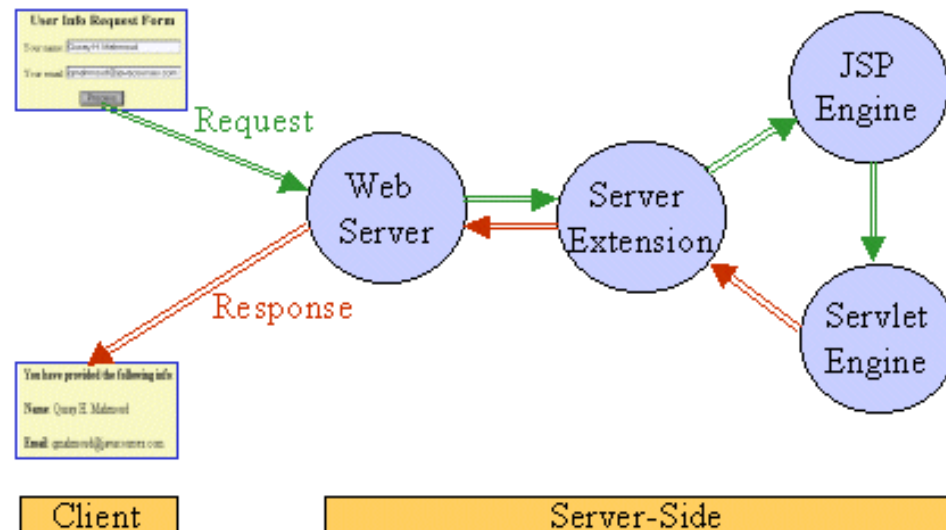
## Aplicaciones 3 Capas

Capa	Responsabilidad
Presentación	Navegabilidad del sistema, Validación de datos de entrada Formateo de los datos de salida, Internacionalización Renderizado de presentación
Lógica de negocios	Conjunto de reglas de negocio que abstraen el mundo real Independiente de la capa de presentación y viceversa
Acceso a datos	Persistencia de las entidades que maneja el sistema (inserción, eliminación, actualizaciones, búsquedas, etc.) No tiene porqué tratarse necesariamente de una base de datos relacional.

# Diseño de la arquitectura de la aplicación

## Arquitectura 3 Capas

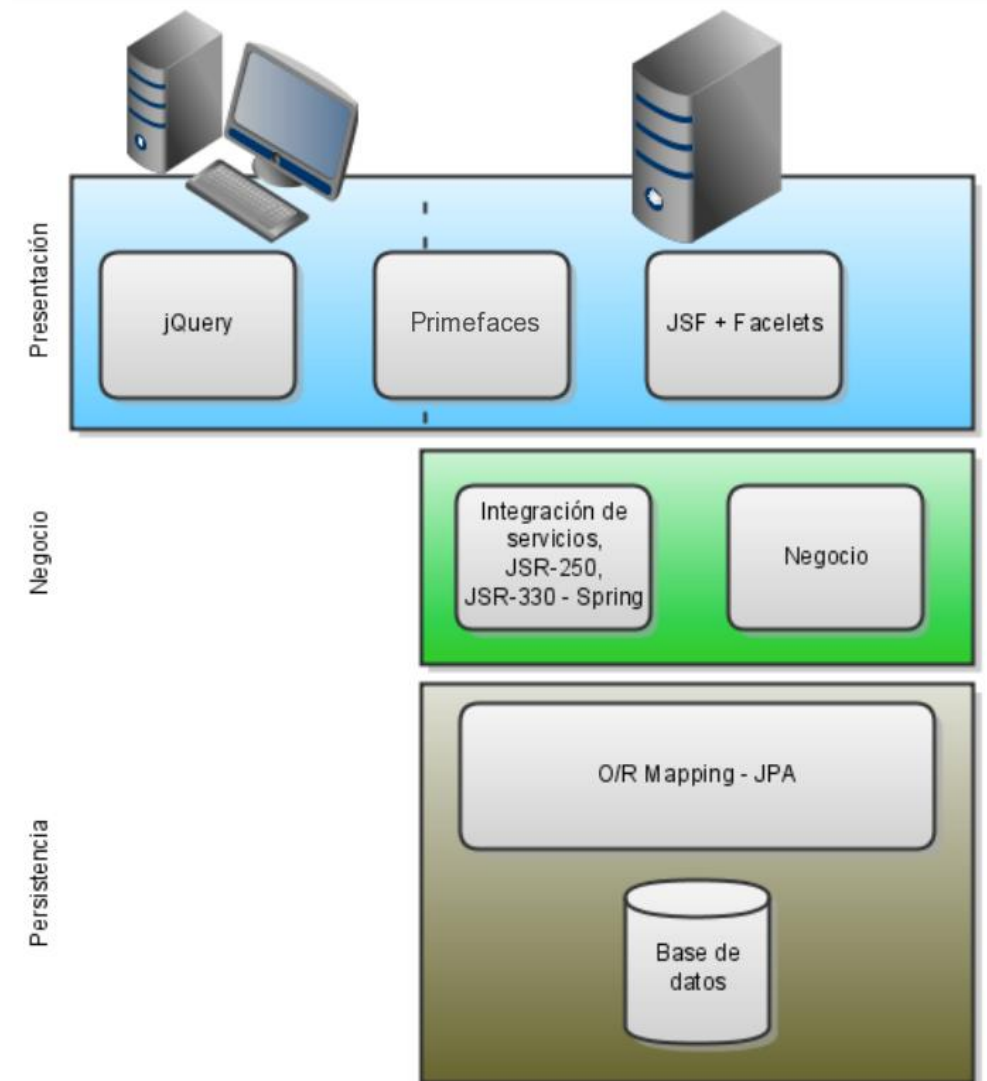
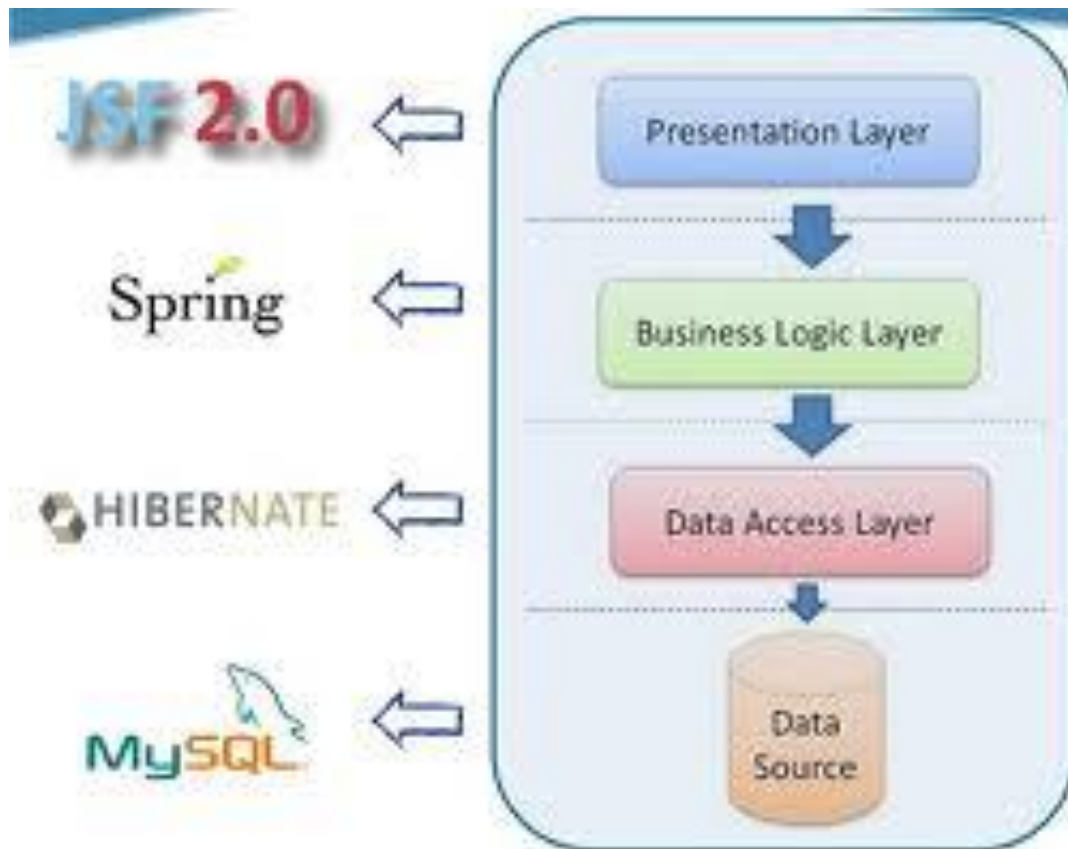
- JSP y Servlets
  - Java Server Pages
  - JSP Capa de presentación
    - Navegabilidad
    - Visualización
  - Beans incrustados asumen la responsabilidad de la capa de negocios y datos



# Diseño de la arquitectura de la aplicación

## Arquitectura de 3 capas

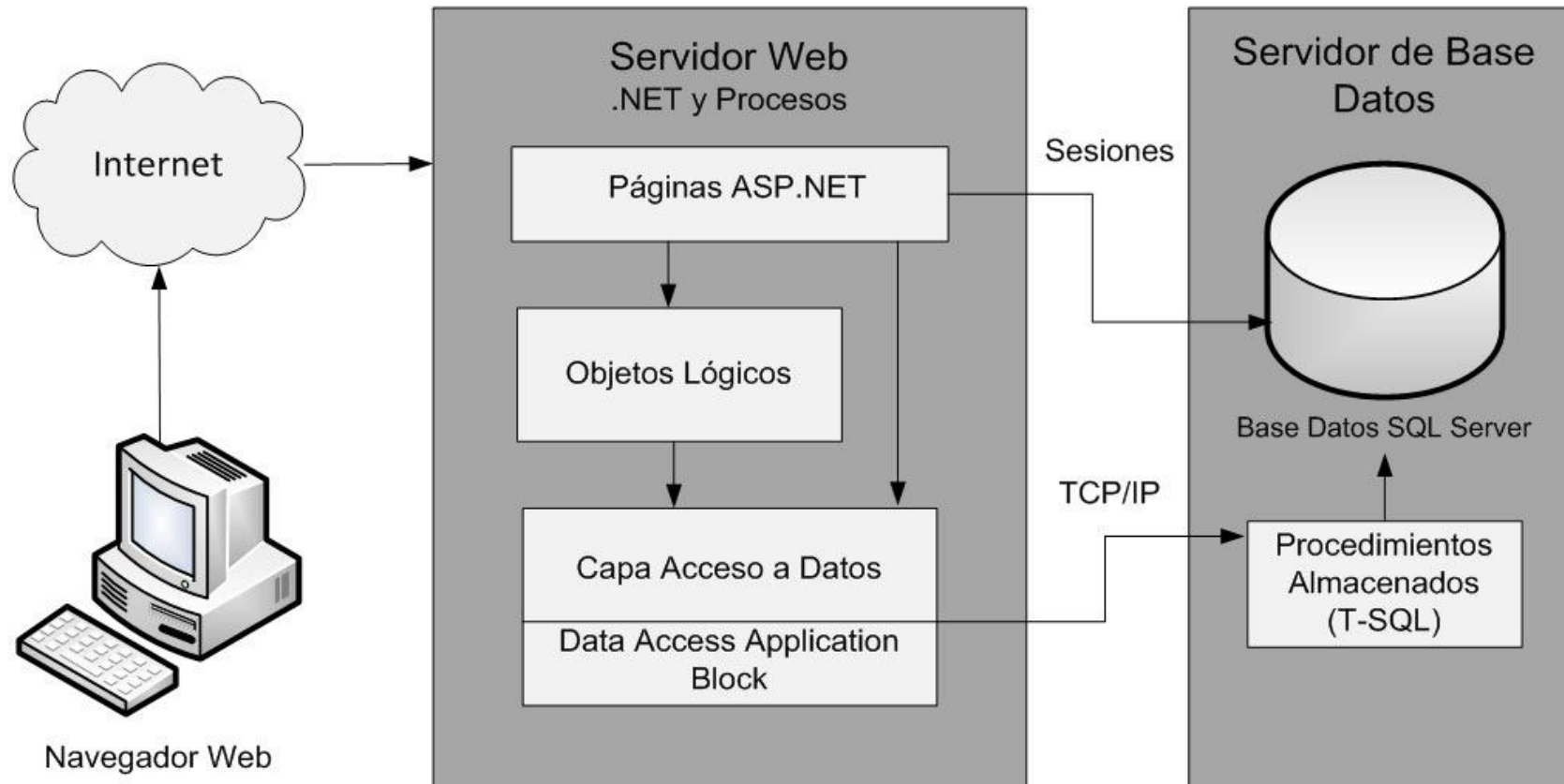
Plataforma J2EE



# Diseño de la arquitectura de la aplicación

## Arquitectura de 3 capas

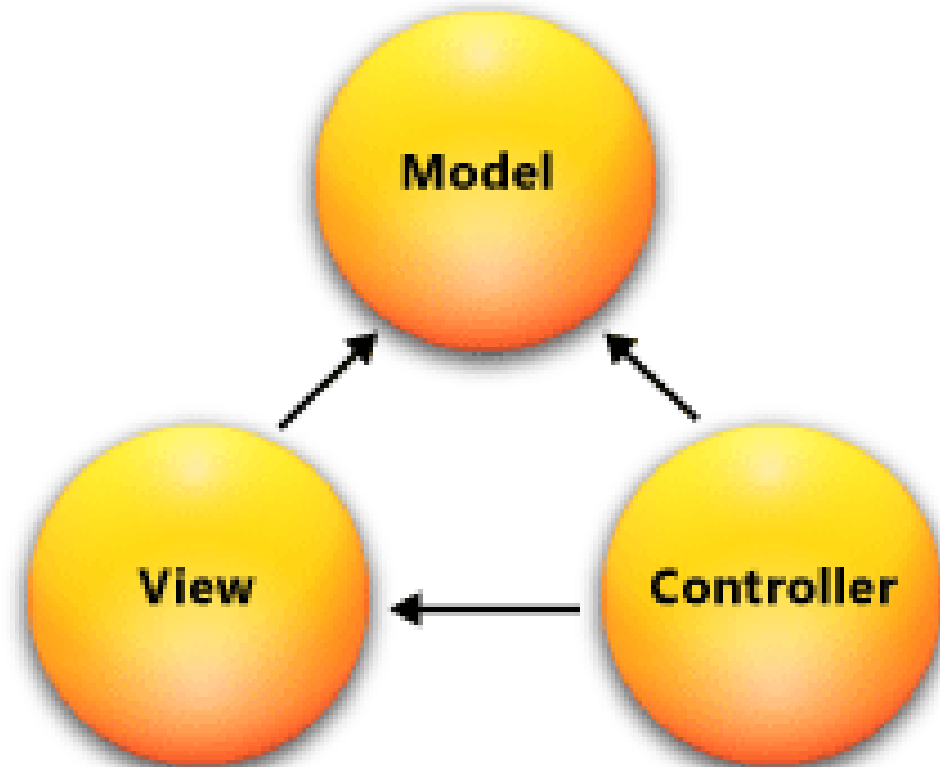
Plataforma .NET



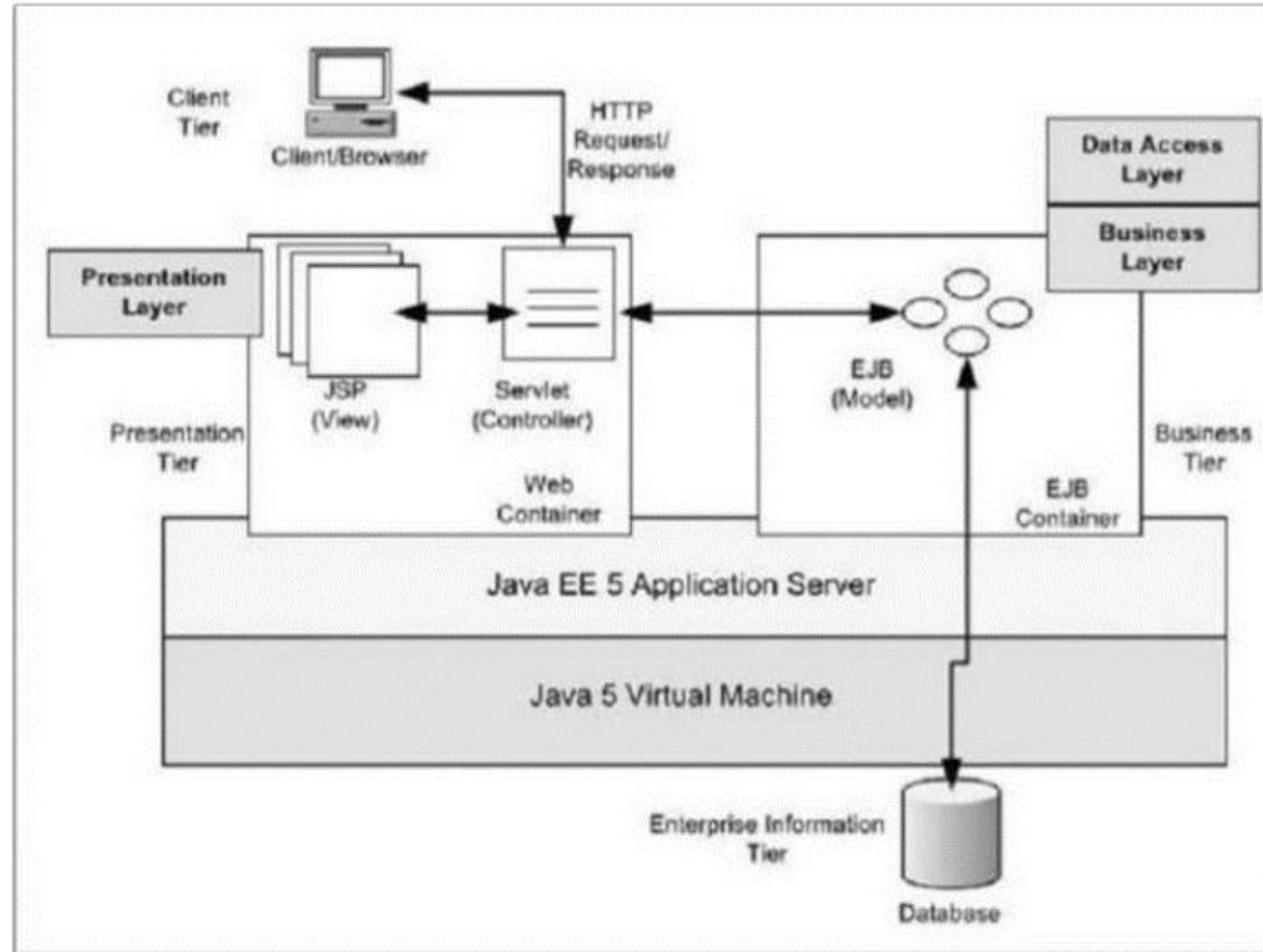
# Diseño de la arquitectura de la aplicación

## Patrón de diseño MVC

- Incorpora el patrón de diseño MVC
  - Modelo
  - Vista
  - Controlador
- Frameworks
  - Struts, Spring, JSF
  - ASP.NET MVC, Spring .NET
  - Symfony
  - Django



# Diseño de la arquitectura de la aplicación MVC - J2EE





# Diseño de la arquitectura de la aplicación

## Planear las capas de la aplicación

- Aspectos de la arquitectura de la aplicación que contribuyen con sus capas
  - Método de acceso a datos
    - La aplicación accederá a una base de datos existente?
    - El diseño de la base de datos será realizado a lo largo del diseño de la aplicación?
  - Separación de responsabilidades
    - Separar las parte de un programa en diferentes secciones o con diferentes responsabilidades y cada sección tiene un propósito diferente.
    - Desarrollo basado en N-Capas
      - Presentación
      - Lógica de negocios
      - Datos

# Planear las capas de la aplicación

## Método de acceso a datos

- Opciones para acceder a los datos
  - Object relational mapper (ORM)
    - Asiste en la conversión de datos de un RDBMS a el modelo de objetos que es necesario para usar dentro de la POO
    - Carga objetos con datos de la base de datos
    - Crea sentencias SQL para hacer persistente los datos

# Object relational mapper (ORM)

## Python - Peewee

Peewee is a simple and small ORM. It has few (but expressive) concepts, making it easy to learn and intuitive to use.

- A small, expressive ORM
- Written in python with support for versions 2.6+ and 3.2+.
- built-in support for sqlite, mysql and postgresql
- tons of extensions available in the *Playhouse, a collection of addons* (postgres hstore/json/arrays, sqlite full-text-search, schema migrations, and much more).



Peewee's source code hosted on [GitHub](#).

# Object relational mapper (ORM)

## Python - Pony

### Declarative queries

Pony allows you to interact with databases in pure Python in the form of generator expressions, which are then translated into SQL. Python generator:

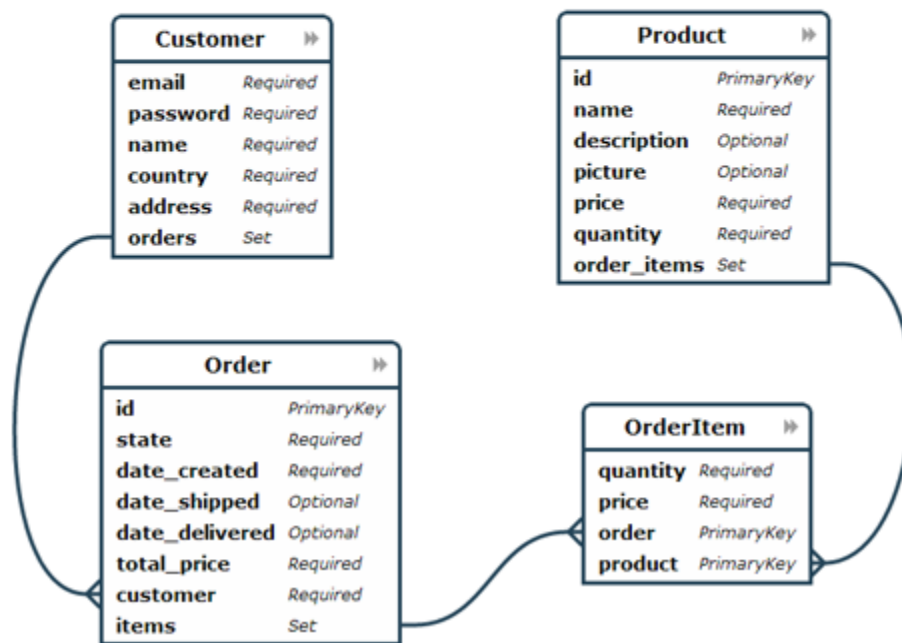
```
select(c for c in Customer
       if sum(c.orders.price) > 1000)
```

is translated to the following SQL:

```
SELECT "c"."id"
FROM "Customer" "c"
  LEFT JOIN "Order" "order-1"
    ON "c"."id" = "order-1"."customer"
GROUP BY "c"."id"
HAVING coalesce(SUM("order-1"."total_price"), 0)
> 1000
```

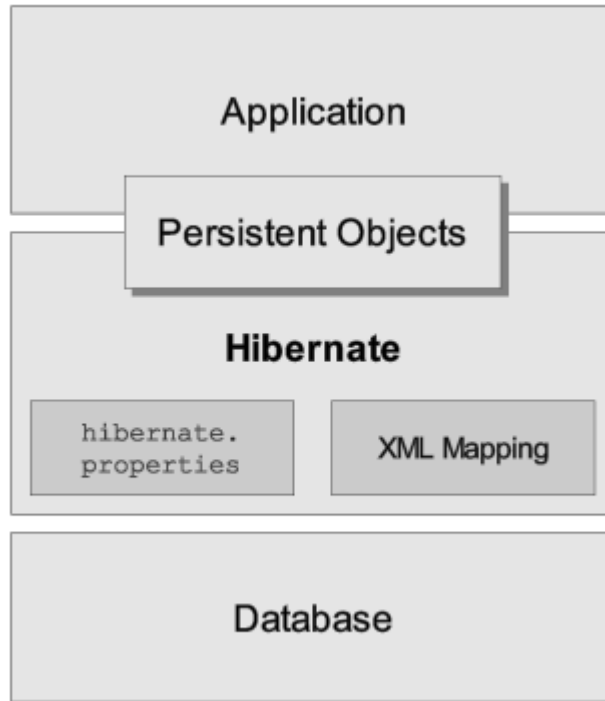
### Diagram editor

You can use the graphical ER diagram editor which generates Python code for models



# Object relational mapper (ORM)

## Java - Hybernate



```
Cat fritz = (Cat) sess.load(Cat.class, generatedId);
```

```
// you need to wrap primitive identifiers  
long id = 1234;  
DomesticCat pk = (DomesticCat) sess.load( DomesticCat.class, new Long(id) );
```

```
Cat cat = new DomesticCat();  
// load pk's state into cat  
sess.load( cat, new Long(pkId) );  
Set kittens = cat.getKittens();
```

```
Cat cat = (Cat) sess.get(Cat.class, id);  
if (cat==null) {  
    cat = new Cat();  
    sess.save(cat, id);  
}  
return cat;
```

```
// fetch ids  
Iterator iter = sess.createQuery("from eg.Qux q order by q.likeliness").iterate();  
while ( iter.hasNext() ) {  
    Qux qux = (Qux) iter.next(); // fetch the object  
    // something we couldnt express in the query  
    if ( qux.calculateComplicatedAlgorithm() ) {  
        // delete the current instance  
        iter.remove();  
        // dont need to process the rest  
        break;  
    }  
}
```

# Object relational mapper (ORM) .NET – Entity Framework

## Insert a new record

To insert a new record:

1. Add an object of **Employee** class. e.g.

```
Employee objEmp = new Employee();
```

[Collapse](#) | [Copy Code](#)

2. Set the value of the properties like:

```
objEmp.FirstName = txtFirstName.Text;
```

[Collapse](#) | [Copy Code](#)

3. Add object to the collection in **ObjectContext** and call **SaveChanges**:

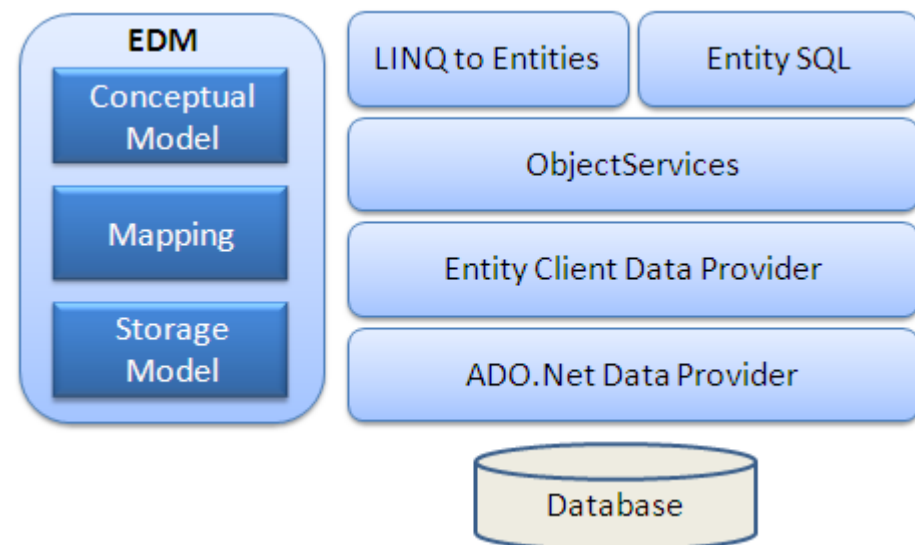
```
db.Employees.AddObject(objEmp);  
db.SaveChanges();
```

[Collapse](#) | [Copy Code](#)

Here is a complete code snippet:

```
LearnEFEntities db = new LearnEFEntities();  
  
Employee objEmp = new Employee();  
objEmp.HREmpId = txtHREmpId.Text;  
objEmp.FirstName = txtFirstName.Text;  
objEmp.LastName = txtLastName.Text;  
objEmp.Address = txtAddress.Text;  
objEmp.City = txtCity.Text;  
  
db.Employees.AddObject(objEmp);  
db.SaveChanges();
```


[Collapse](#) | [Copy Code](#)



# Object relational mapper (ORM)

## PHP – Propel

```
1  <?php
2
3  use Propel\Tests\Bookstore\Author;
4  use Propel\Tests\Bookstore\Book;
5  use Propel\Tests\Bookstore\BookQuery;
6  use Propel\Runtime\ActiveQuery\Criteria;
7
8  $book = new Book();
9  $book->setTitle('Lord of Propel');
10 $book->setPrice(25);
11
12 $author = new Author();
13 $author->
```



### Prerequisites

Propel just requires:

- [PHP 5.4](#) or newer, with the DOM (libxml2) module enabled
- A supported database (MySQL, MS SQL Server, PostgreSQL, SQLite, Oracle)

Propel also uses some Symfony2 components to work properly:

- [Config](#) : uses in the source code to manage and validate configuration.
- [Console](#) : which manage the generators propel uses.
- [Yaml](#)
- [Validator](#) : a way you manage validations with Propel.
- [Finder](#) : uses in the source code to manage the files.

# Planear las capas de la aplicación

## Método de acceso a datos

- Opciones para acceder a los datos
  - Object relational mapper (ORM)
    - Asiste en la conversión de datos de un RDBMS a el modelo de objetos que es necesario para usar dentro de la POO
    - Carga objetos con datos de la base de datos
    - Crea sentencias SQL para hacer persistente los datos
    - Permite generar la base datos a partir del modelo o el modelo a partir de la base de datos
  - Componentes propios
    - Programar componentes que realicen conversiones de objetos a tablas (sentencias insert, update, delete o procedimientos almacenados)
    - Recomendado cuando:
      - El modelo de datos no es cercano al modelo de objetos
      - El formato de la base de datos no es puramente relacional (bases NoSQL)



# Taller: Uso de un ORM