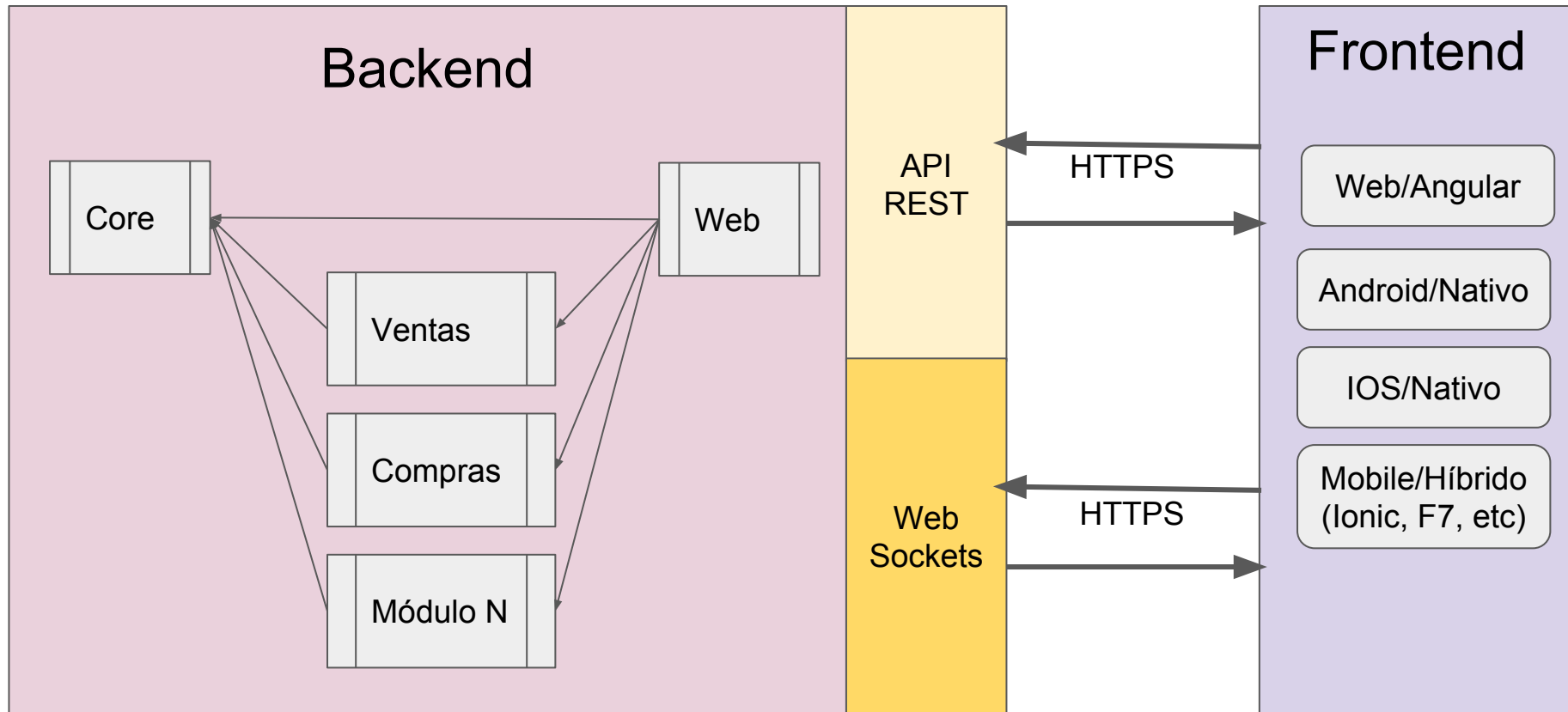


# Modularización e Implementación

# Big Picture (Módulos, Dependencias, UI, etc)



# Tecnologías Backend

## Base

- JSE 1.8
- JEE 1.7

## Ciclo de vida de SW

- Maven 3
- GIT
- Bower

## Frameworks

- SpringMVC
- JPA/Hibernate

## Test

- JUnit 4.11

## Logging

- Log4J 2 sobre SLF4J

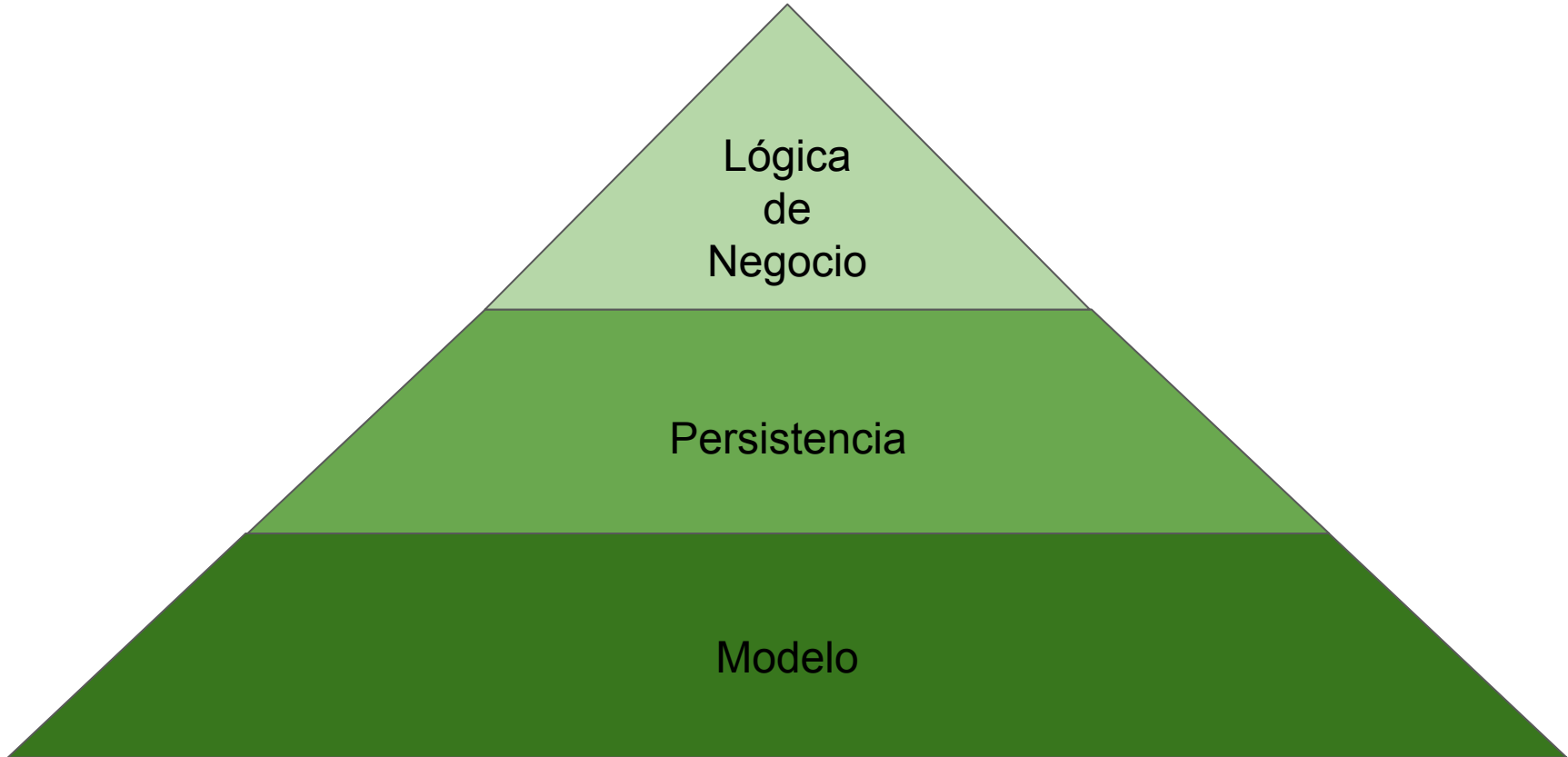
## Codificación

- UTF-8

## Orden de preferencia en la Configuración

- 1) Anotaciones, 2) XML, 3) JSON, 4) AD-HOC

# Capas mínimas de un Módulo



# Responsabilidades de los módulos

## Modelo:

- Definir las clases componentes del negocio.
- Herencia, Interfaces, encapsulación de los atributos y relaciones.
- Sobreescritura de métodos *hashCode()*, *toString()*, *equals()*, *compareTo()*
- Todos Serializables
- Anotaciones de persistencia JPA
- Tests Unitarios

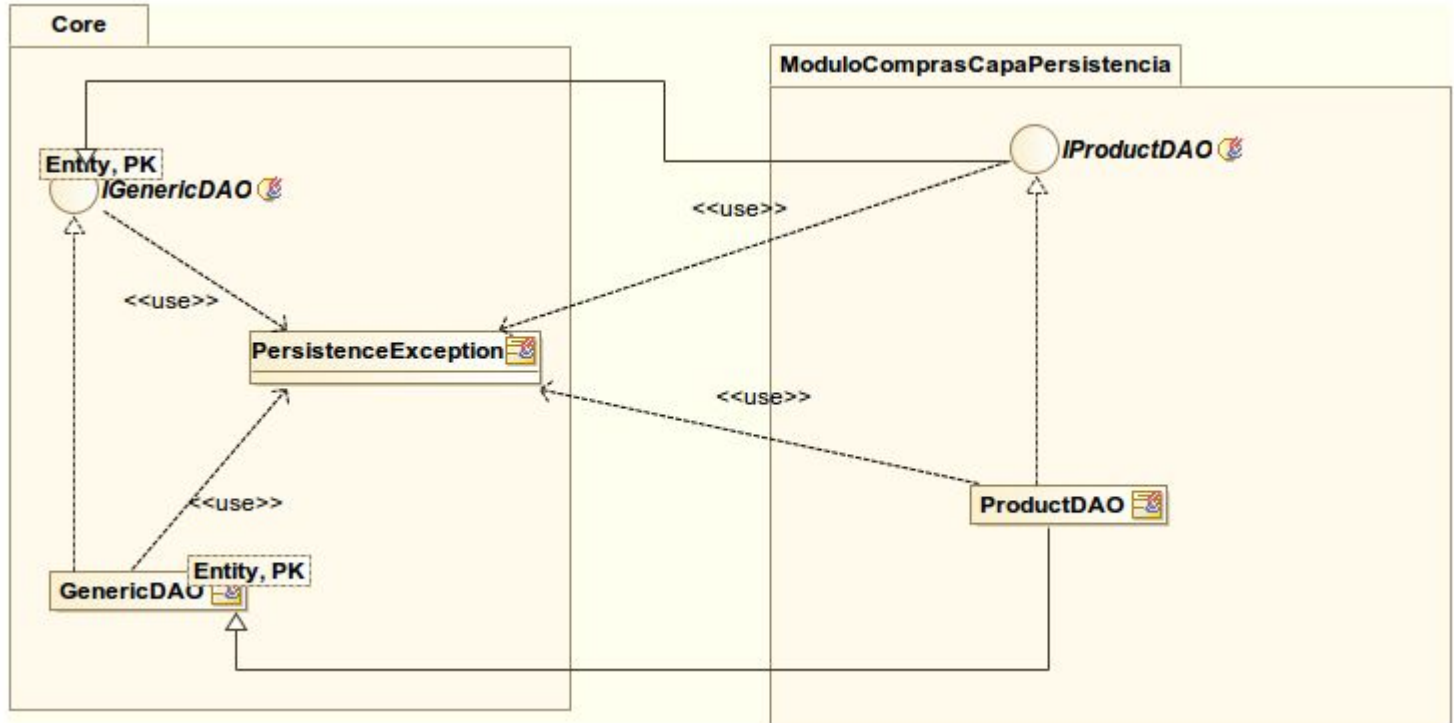
# Responsabilidades de los módulos

## Persistencia:

- Implementar los servicios de persistencia genéricos definidos en **Core**, interface *IGenericDAO<Entity, PK>* extendiendo a *GenericDAO<Entity, PK>*, para el **Modelo**.
- Extender los servicios definidos en **Core** de ser necesario.
- Encapsular las excepciones a *PersistenceException* (definida en **Core**)
- Logging de ERROR y DEBUG por método/servicio implementado
- Tests Unitarios

# Responsabilidades de los módulos

## Persistencia



# Responsabilidades de los módulos

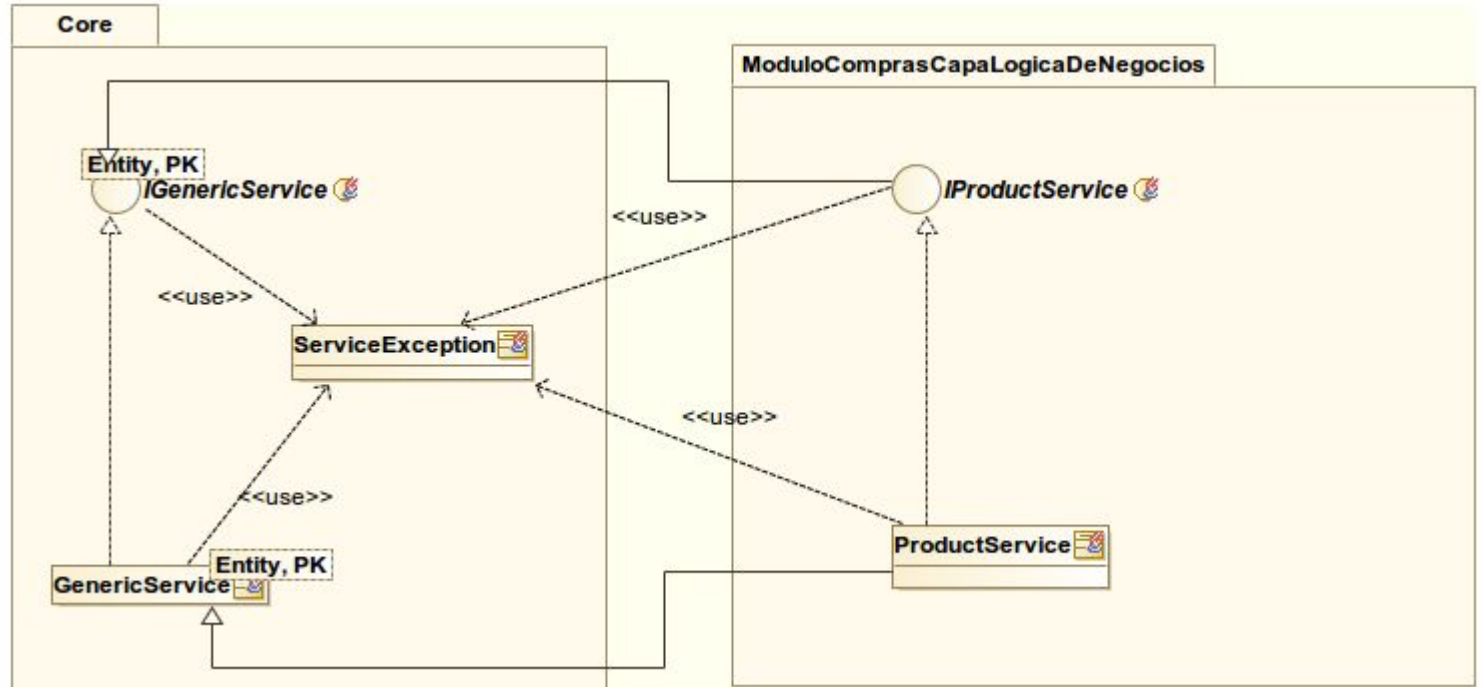
## Lógica de Negocios:

- Implementar los servicios genéricos de lógica de Negocios definidos en **Core**, interface *IGenericService<Entity, PK>* extendiendo a *GenericService<Entity, PK>*
- Extender los servicios definidos en **Core** de ser necesario.
- Encapsular las excepciones a *ServiceException* (definida en **Core**)
- Logging de ERROR y DEBUG por método/servicio implementado.
- **Los servicios deben ser totalmente agnósticos a la presentación final.**
- Tests Unitarios



# Responsabilidades de los módulos

Lógica de  
Negocios



# Servicios Core de Persistencia

```
public interface IGenericDAO<Entity, PK extends Serializable> {  
    public Entity save(Entity t) throws PersistenceException;  
    public Entity update(Entity t) throws PersistenceException;  
    public Entity saveOrUpdate(Entity t) throws PersistenceException;  
    public Entity load(PK id) throws PersistenceException;  
    public void delete(Entity t) throws PersistenceException;  
    public List<Entity> list() throws PersistenceException;  
}
```

# Servicios Core de Lógica de Negocios

```
public interface IGenericService<Entity, PK extends Serializable> {  
    public Entity save(Entity entity) throws ServiceException;  
    public Entity update(Entity entity) throws ServiceException;  
    public Entity saveOrUpdate(Entity entity) throws ServiceException;  
    public Entity load(PK id) throws ServiceException;  
    public void delete(Entity entity) throws ServiceException;  
    public List<Entity> list() throws ServiceException;  
}
```

# Web

REST: <http://www.restdoc.org/spec.html>

# Documentación REST

## Swagger

- <http://editor.swagger.io/>

# Estructura básica cliente Angular.JS

