**SOLUCIÓN A PRUEBA TÉCNICA OWNK**

**PRESENTADO POR**

**MIGUEL ÁNGEL BUITRAGO OVIEDO**

**Modelo Vista Controlador (Patrón MVC)**

Modelo–vista–controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

De manera genérica, los componentes de MVC se podrían definir como sigue:

* **El Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
* **El Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.
* **La Vista:** Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

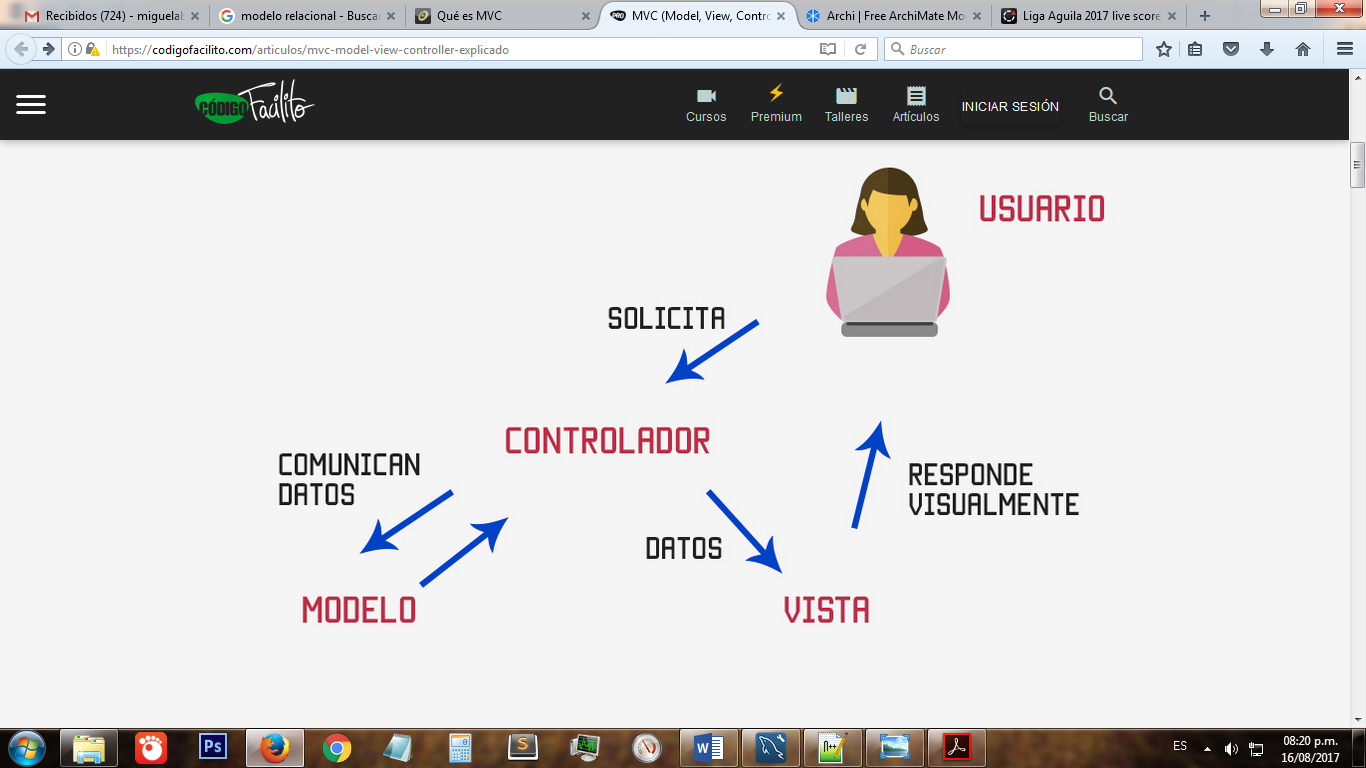


Imagen 1 Patrón MVC. Tomado de: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

**MyBatis**

MyBatis es un framework de persistencia que soporta SQL, procedimientos almacenados y mapeos avanzados. MyBatis elimina casi todo el código JDBC, el establecimiento manual de los parámetros y la obtención de resultados. MyBatis puede configurarse con XML o anotaciones y permite mapear mapas y POJOs (Plain Old Java Objects) con registros de base de datos. A diferencia de las herramientas ORM MyBatis no mapea objetos Java a tablas de base de datos sino métodos a sentencias SQL.

Permite utilizar todas las funcionalidades de la base de datos como procedimientos almacenados, vistas, consultas de cualquier complejidad o funcionalidades específicas del proveedor. Es una herramienta indicada para bases de datos legadas, desnormalizadas o cuando es preciso tener el control total del SQL ejecutado.

Simplifica la programación frente al uso directo de JDBC. Las líneas de código necesarias para ejecutar una sentencia se reducen casi siempre a una. Esta simplificación ahorra tiempo y evita errores habituales como olvidar cerrar una conexión a base de datos, realizar incorrectamente un mapeo de datos, exceder el tamaño de un result set u obtener varios resultados cuando se esperaba solo uno.

**Spring**

Un elemento clave de Spring es el soporte de infraestructura a nivel de aplicación: Spring se concentra en la “plomería” de las aplicaciones empresariales por lo que los equipos de trabajo se pueden concentrar en el nivel de lógica de negocio, sin innecesarias ataduras para especificar los ambientes de despliegue.

Spring ayuda a solventar este problema (gestionar los objetos generados por diferentes frameworks) ya que cambia las responsabilidades y en vez de que el propio desarrollador sea el encargado de generar los objetos de cada uno de los frameworks es Spring basandose en ficheros xml o anotaciones el encargado de construir todos los objetos que la aplicación va a utilizar.

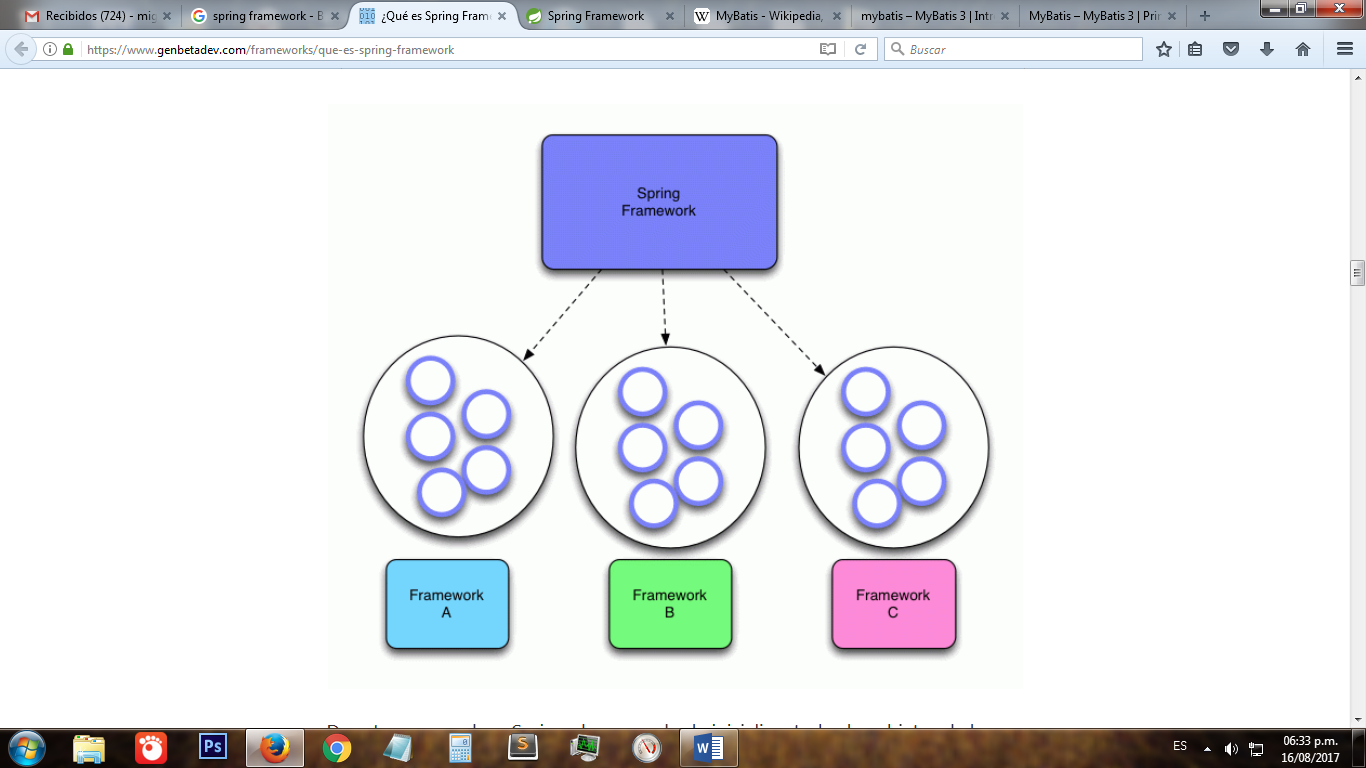


Imagen 2 Framework Spring. Tomado de: https://www.genbetadev.com/frameworks/que-es-spring-framework

De esta manera al ser Spring el encargado de inicializar todos los objetos de los distintos frameworks, es también el responsable de asegurarnos que se integran de la forma correcta.

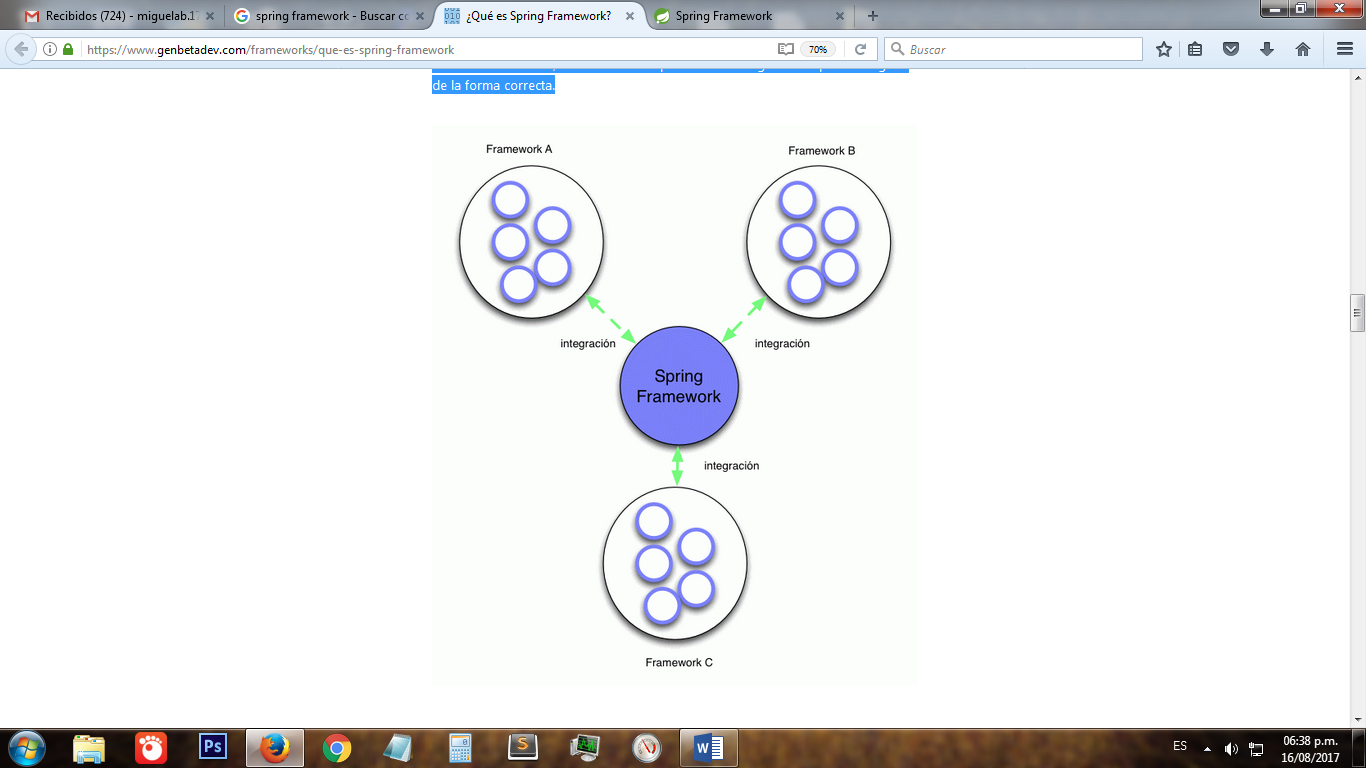


Imagen 3 Framework Spring. Tomado de: https://www.genbetadev.com/frameworks/que-es-spring-framework

**AngularJS**

AngularJS es un famework de JavaScript. Puede ser añadido a una página HTML con una etiqueta <script>. AngularJS extiende atributos HTML con directivas, y enlazamiento de información a HTML con expresiones. AngularJS es un framework MVC de JavaScript para el Desarrollo Web Front End que permite crear aplicaciones SPA Single-Page Applications.

**PROBLEMA**

Se debe crear una aplicación web con las siguientes funcionalidades

✓ Registrar/Actualizar/Eliminar Vendedor.

✓ Registrar/Actualizar/Eliminar Producto.

✓ Asociar un uno o varios productos al vendedor.

✓ ver productos por vendedor.

El vendedor tiene los siguientes atributos principales

• Nombres

• Apellidos

• Identificación

E producto contiene los siguientes atributos

• Código

• Descripción

• Imagen

**SOLUCIÓN**

Posterior al análisis del problema, se plantea una arquitectura MVC: debido a la modularidad y la separación de la vista (lo que se presenta al usuario, la información), de los datos y de su procesamiento. Además de ser una buena práctica de programación, el mantenimiento y las modificaciones, como cambios de la vista, uso de otro framework o cambios en la persistencia son relativamente sencillos de hacer debido a la flexibilidad de la estructura. Por todo lo anterior se eligió como la mejor opción.

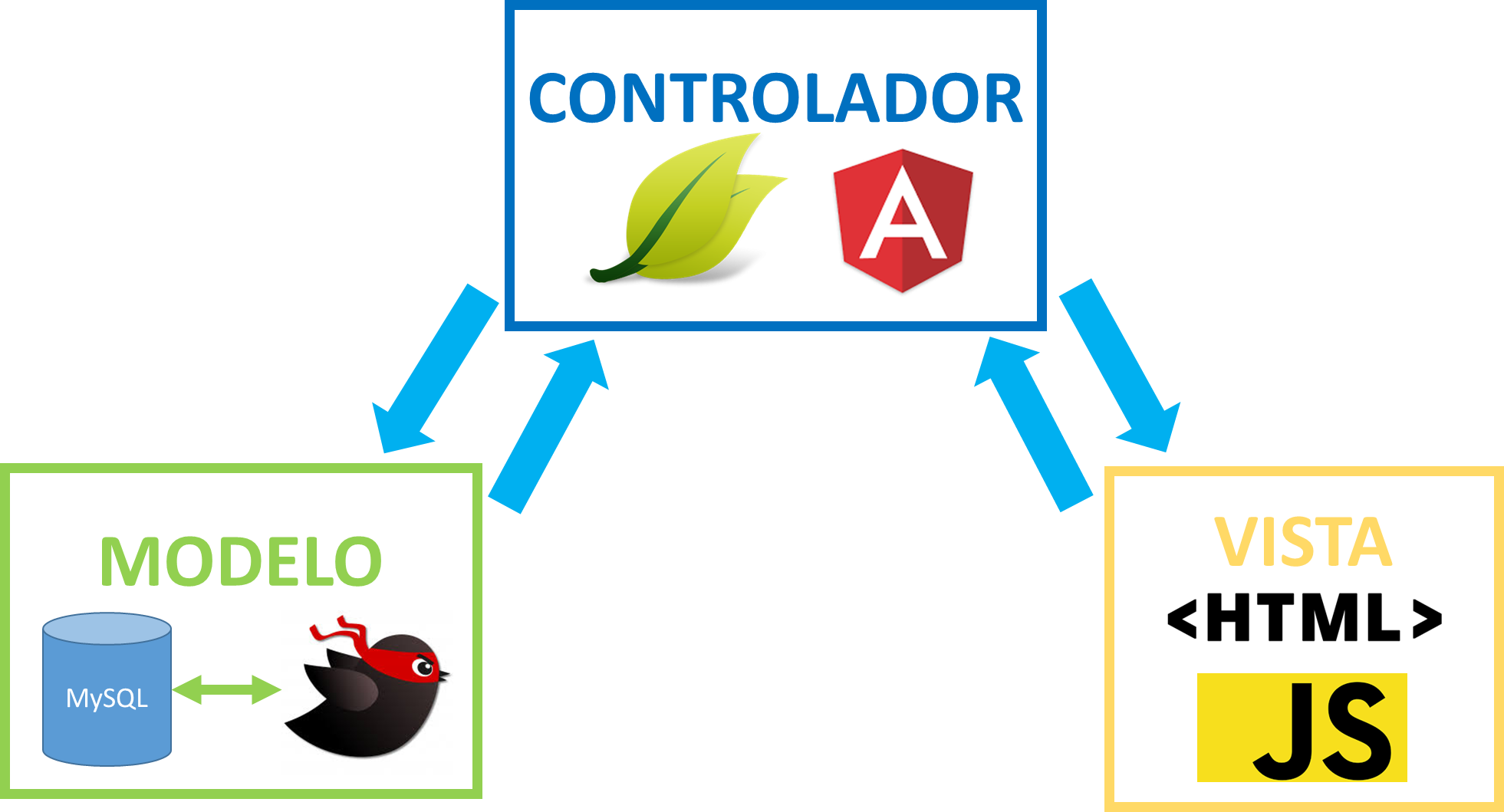


Imagen 4 Modelo Vista Controlado del proyecto

**Entorno utilizado**

* Eclipse IDE for Java Developers Oxygen Release (4.7.0)
* MySQL Workbench 6.3
* Xampp V3.2.2
* Spring IDE V3.9 for Eclipse
* Notepad V7.3.3
* Spring framework V4.1.6. RELEASE
* MyBatis V3.4.1.
* Java Version 1.8

**Persistencia**

Se utilizó Xampp V3.2.2 para el servicio de MySQL. En un primer analisis del problema, se lograron identificar dos tablas: vendedor y producto, además en las especificaciones es evidente la relación entre ambas tablas. A pesar de lo anterior un modelo de base de datos con dos tablas es una solución insuficiente por dos razones: la primera, es semanticamente incorrecto, debido a que un producto solo puede existir siempre y cuando haya un vendedor registrado, lo cual no tiene mucho sentido y la segundo, es muy estático, ya que no permite, por ejemplo, que un empleado se relacione con muchos productos y viceverza, lo cual, aunque no esté en las especificaciones, es una posibilidad muy real para el futuro.

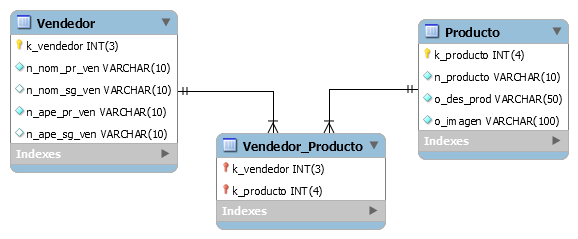
****

Imagen 5 Modelo Relacional. Hecho en MySQL Workbench 6.3

Se puede argumentar en contra del modelo debido a la tabla extra, un cargo adicional para el procesamiento; sin embargo se debe evaluar de manera más detenida qué implica realmente la tabla adicional. Lo que aporta a la solución, es flexibilidad, lo que permite escalar la solución a posibles cambios en la lógica de negocio, aspecto siempre deseado cuando se diseña un sistema de información. Ahora bien, ¿cuál es el precio que se paga? Es casi nulo, porque las consultas con la tabla extra no presentan un gran cargo computacional si se compara con el modelo de dos tablas, la razón: las tablas no tienen muchos campos, lo que supondría una gran carga computacional en las proyecciones (Select) si así fuese, ¿por qué evaluar sobre las proyecciones? Porque la consulta típica cuando se trabaja con las llamadas “tablas de rompimiento” implica join o unión entre tablas, este tipo de consultas crece su complejidad con respecto a la cantidad de campos y no a la cantidad de registros, así que en este caso una tabla extra no tiene costo, mientras que representa una gran ganancia.

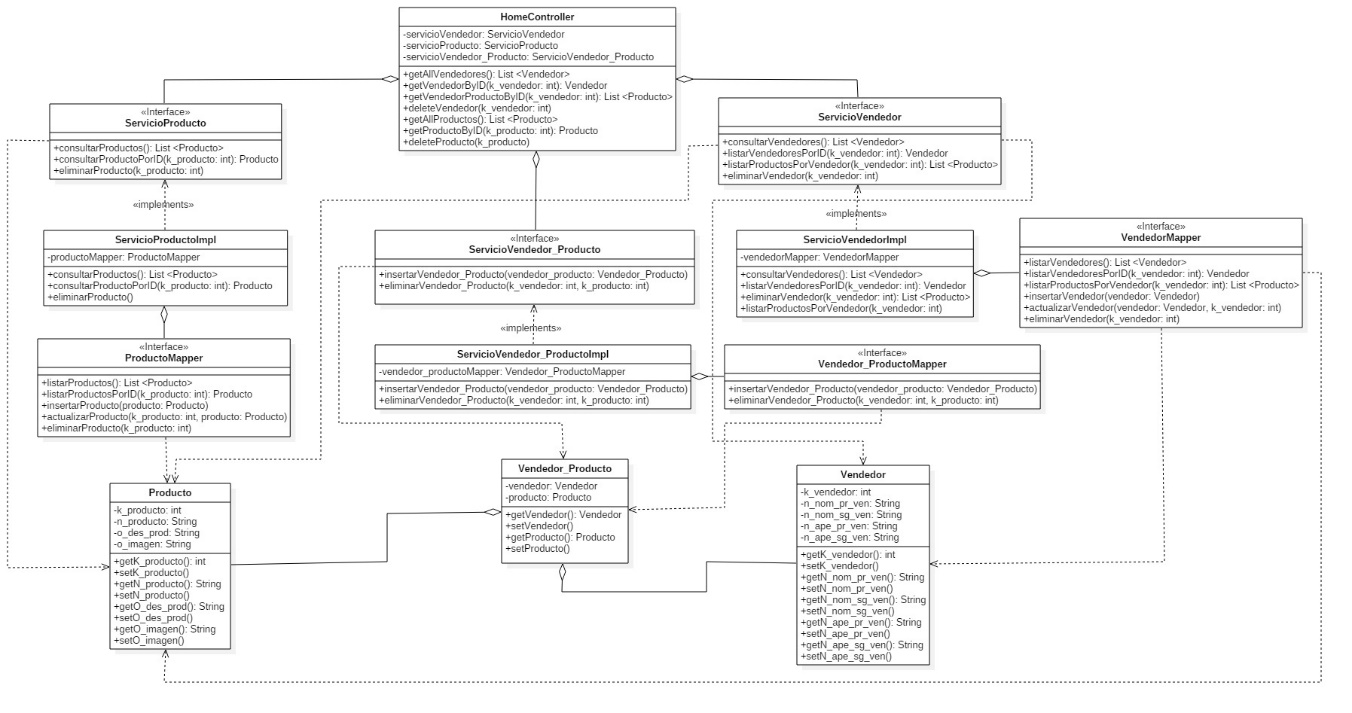


Imagen 6 Diagrama de Clases. Hecho en StarUML 2.8

**Bibliografía**

https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

http://www.mybatis.org/mybatis-3/es/

https://es.wikipedia.org/wiki/MyBatis

https://projects.spring.io/spring-framework/

https://www.genbetadev.com/frameworks/que-es-spring-framework

https://www.w3schools.com/angular/angular\_intro.asp

https://carlosazaustre.es/empezando-con-angular-js/