Análisis y diseño del back-end

Descripción del sistema

De acuerdo a la Especificación de Requisitos del Sistema, el sistema permitirá al usuario martillero dar de alta usuarios consignatarios, quienes consignarán (entregarán) artículos propios al martillero para la venta. El sistema también permitirá al usuario martillero crear subastas para poner en venta dichos artículos como lotes. En dichas subastas los usuarios pujadores podrán pujar por dichos lotes, siendo el dueño de la última puja el ganador del lote.

Límites del sistema

1. El programa no permitirá a los usuarios consignatarios crear sus artículos directamente ya que el usuario martillero debe verificar las condiciones y tasarlos antes de ponerlos en subasta.
2. Tampoco permitirá a los consignatarios o pujadores comunicarse entre ellos, se podrán únicamente comunicar con el martillero a través del formulario de contacto.
3. El martillero decide el precio base al cual se venderá cada lote.
4. No soportará precio de reserva por debajo del cual el lote no se venda.
5. No permitirá baja de usuarios.

Definición del modelo del sistema

- Usuario: Existen tres tipos de usuarios en el sistema, a saber:
    - *Martillero* (único en el sistema y que tiene jerarquía de administrador, creado al momento de instalar el sistema)
    - *Consignatario* (cuyas cuentas son creadas por el usuario *Martillero*)
    - *Pujador* (cuyas cuentas son creadas por los pujadores mediante la página de registración).
- Artículo: Un artículo representa un objeto presentado por el consignatario al martillero cuyo estado ha sido verificado, tasado e ingresado al sistema pero que todavía no ha sido puesto en subasta.
- Lote: Es un artículo puesto a la venta en la siguiente subasta.
- Subasta: Es el proceso por el cual una lista de lotes es presentada a los pujadores en un orden predeterminado donde cada uno puede ser vendido a la puja más alta o puede quedar vacante.
- Puja: Es la acción por parte del pujador de aumentar el precio del lote actual a la venta.
- Venta: Es la puja más alta para un determinado lote que ha sido aceptada por el martillero.

Diagrama de colaboración entre clases

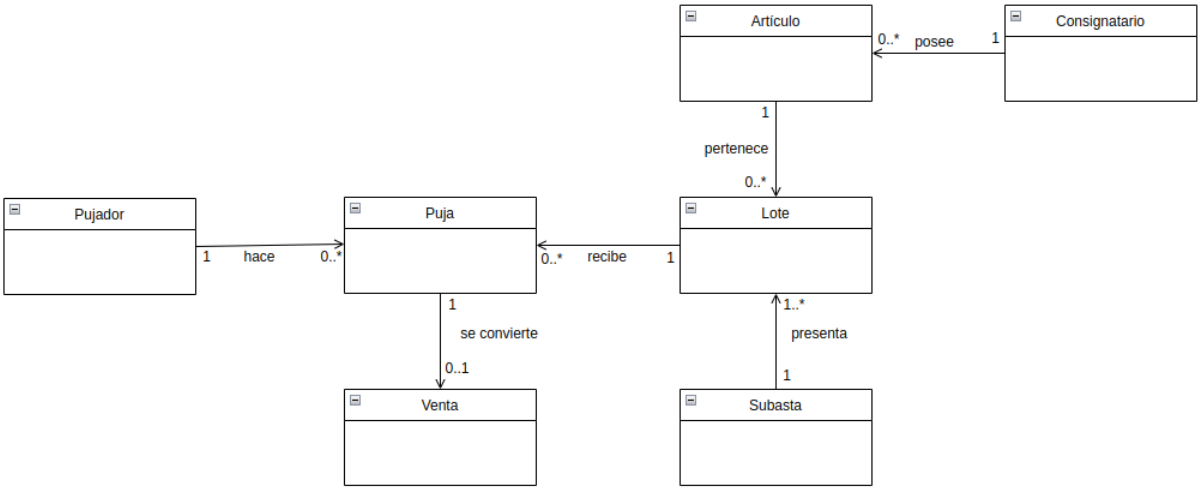


Fig. 1: Diagrama de colaboración entre clases

Empezamos realizando un diagrama de colaboración entre clases para obtener una vista general del sistema. Identificamos en primera instancia a un pujador, la persona interesada en comprar un lote a través de una puja. Y también al consignatario, la persona interesada en vender un artículo en la subasta.

Diagrama de clases

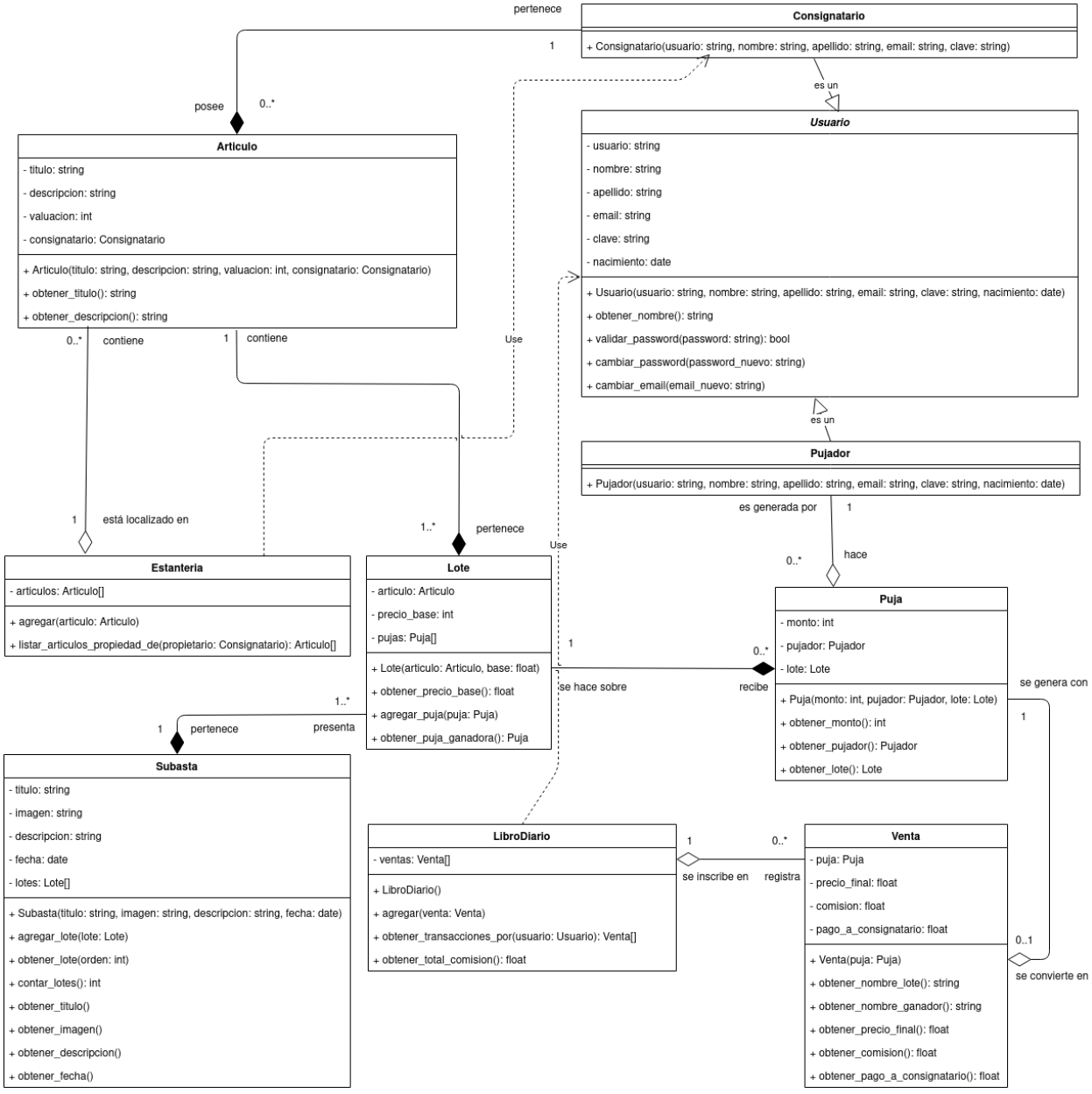


Fig. 2: Diagrama de clases

A partir de dicho diagrama se generó varias iteraciones del diagrama de clases. Todos los usuarios del sistema comparten una base abstracta *Usuario* que provee servicios para acceder y grabar información por lo que las clases *Pujador* y *Consignatario*, sin embargo al igual que en una subasta real poseen distintos derechos y obligaciones (el consignatario no puede pujar, el pujador no puede consignar artículos a la venta por conflicto de intereses).

Al igual que en un negocio de subastas la *Estantería* contiene un conjunto de artículos que todavía no han sido puestos en una subasta. Es posible entregarle a un subastador un mueble

o artículo en consignación y éste puede esperar la subasta adecuada para ponerlo a la venta, por ejemplo si en la subasta hay una mayoría de cuadros y obras de arte poner a la venta un artículo de época como un bastón o una galera hará que obtenga pujas mucho menores que las que recibiría si se pusiesen en una subasta con mayoría de artículos de vestimenta. Al momento de recibir el artículo el subastador lo verificará y lo tasará con lo cual asignará un valor probable de venta.

La clase *Lote* representa al artículo puesto a la venta en la subasta con un número de orden y un valor base de venta. A diferencia de la tasación el valor base es el valor por el cual comenzará la subasta de estilo inglés a partir del cual cada pujador aumentará el precio. La clase Subasta representa a la información de la subasta incluyendo la lista de lotes que se van a vender.

La puja de cada pujador es representada por la clase *Puja* y no es más que la relación entre un lote y un pujador o postor. Si el lote no posee pujas se lo considera desierto y no se vende, si por el contrario posee pujas se venderá a la puja más alta. Esa puja más alta se convierte en una venta que queda registrada en el *LibroDiario* que no es más que una lista de ventas de la misma manera que la *Estantería* es una lista de artículos.

Algo para aclarar es que el lote es único para la subasta: si no se vende el lote en sí queda registrado en esa subasta, sin embargo para la siguiente vez que se pone en venta se asocia el mismo artículo con un nuevo lote ya que es posible que el martillero decida bajar el precio base si no tuvo pujadores.

Diagrama relacional

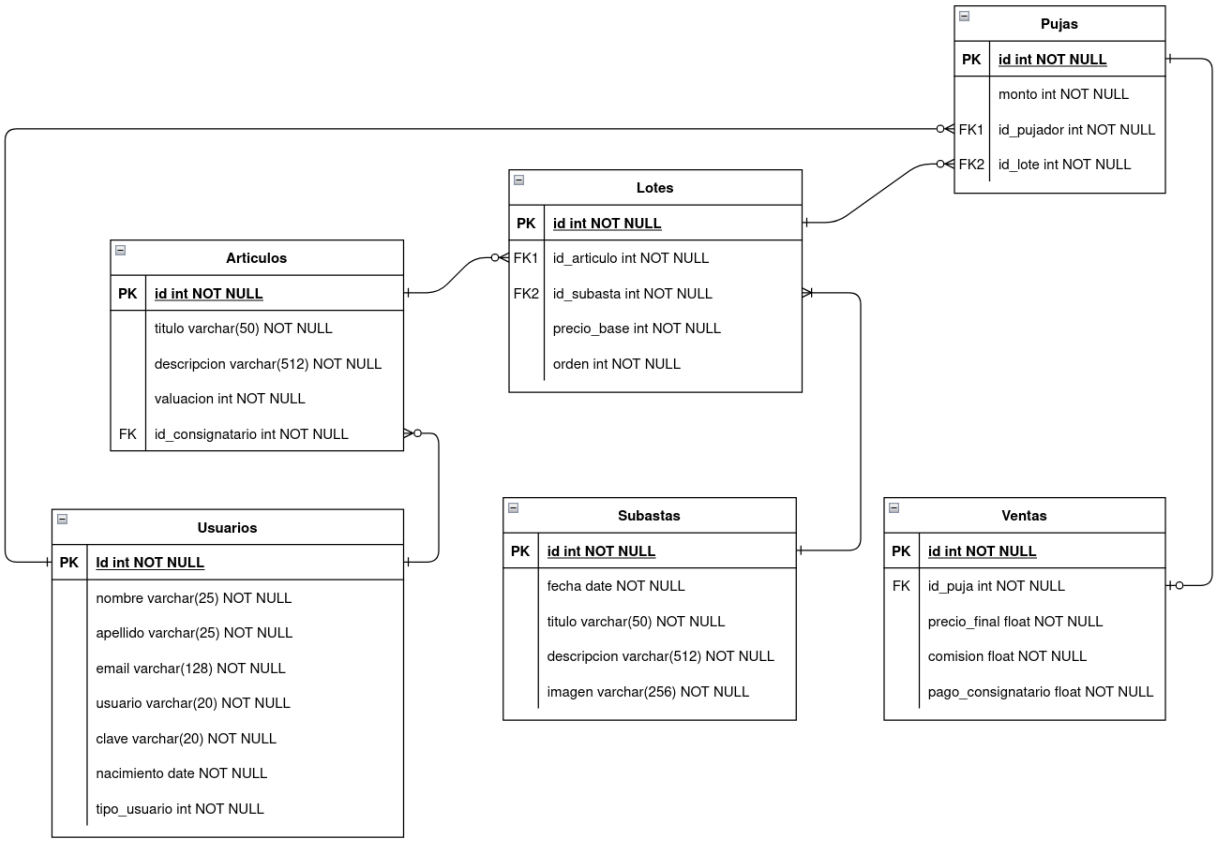


Fig. 3: Diagrama relacional

El modelo relacional es muy similar al de colaboraciones. Tanto la clase *Consignatario* como *Pujador* se guardan en la misma tabla *Usuario* con un tipo para diferenciarlos. Aunque el modelo de base de datos permitiría a un consignatario realizar pujas o a un pujador consignar artículos la lógica de negocios se encargará de evitarlo. Una subasta vacía no debería ser registrada por lo que también la lógica de negocios se encargaría de evitarlo. No todas las pujas son ventas, únicamente la más alta para un artículo determinado lo es. Los lotes siempre tienen un artículo para vender pero los artículos no necesariamente tienen lotes (como se dijo, si no están a la venta en la subasta no figurarán en ninguno).

Diagrama entidad-relación

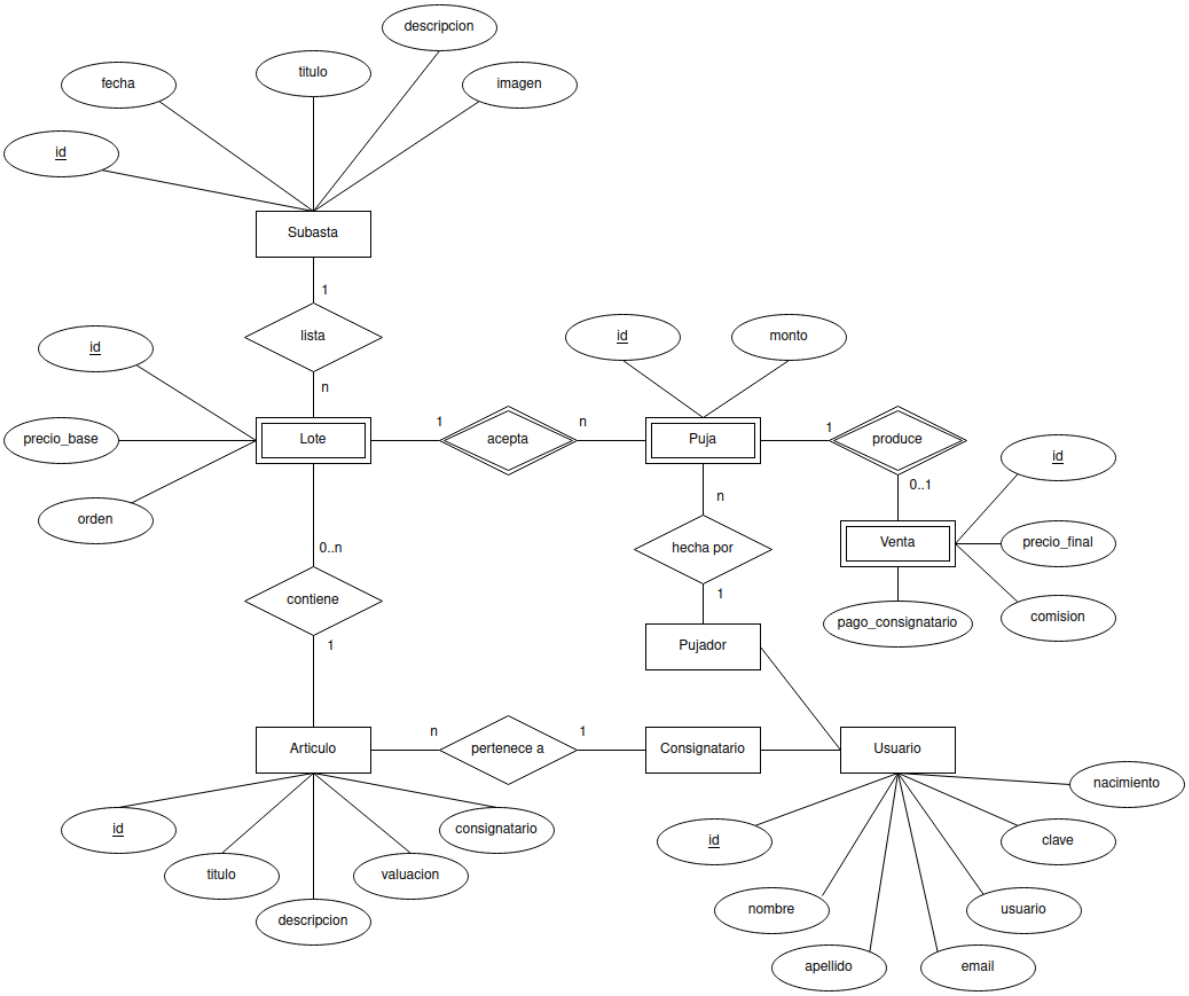


Fig. 4: Diagrama entidad-relación

Model-View Controller

En base al mapa de sitio que diseñamos terminamos creamos 17 páginas web. Sin embargo algunas son parecidas por lo que recibiendo un modelo vista distinto podrían ser reusadas sin problemas (por ejemplo, la lista de compras, la de consignaciones y la del catálogo, tal vez la de subasta por parte del martillero y por parte del pujador, las páginas de contacto internas y externas, etc).

También armamos una lista inicial de 11 controladores que usaríamos. No todas las páginas utilizarán un controlador (por ejemplo, la página sobre la compañía y la página de funcionamiento no necesitan ningún controlador ni modelo, mientras que las distintas páginas de contacto impactarán en el mismo controlador). También podríamos juntar algunos controladores o separarlos, eso se verá durante el segundo Sprint.

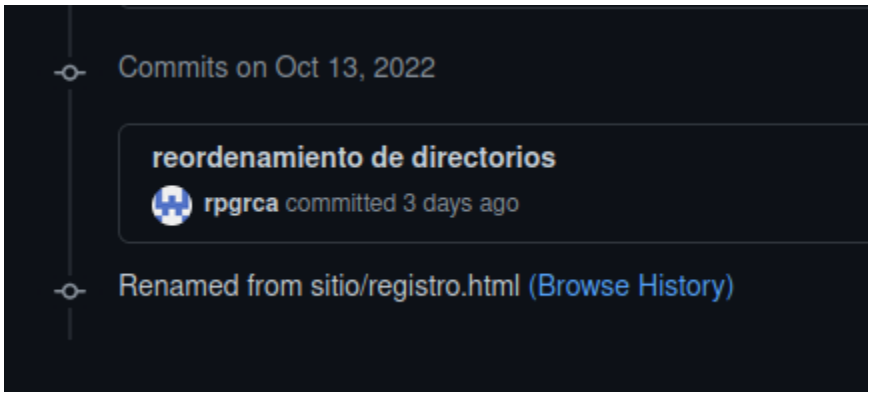
Finalmente tenemos 10 modelos para manejar la interacción con la base de datos incluyendo una abstracta.

Referencias

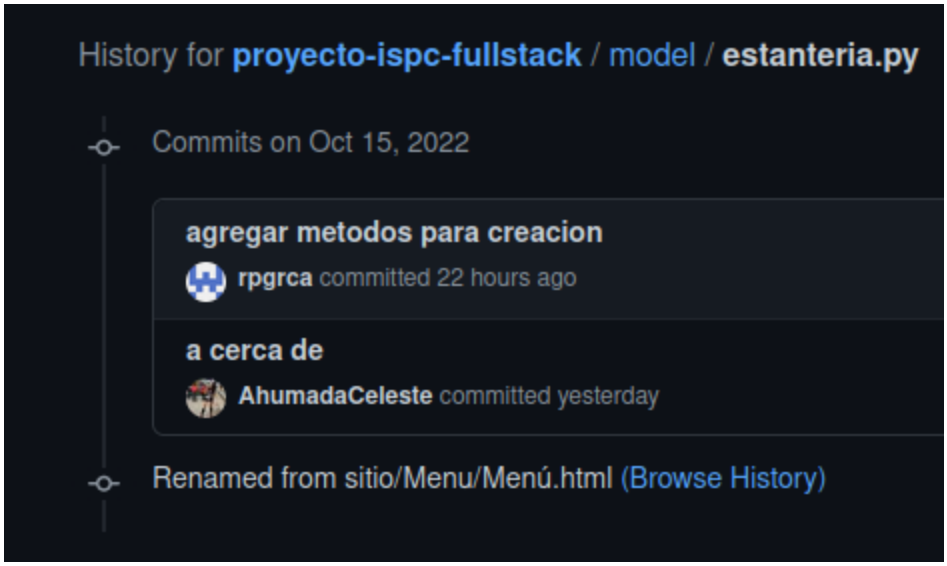
- Wiki del proyecto: https://github.com/rpgrca/proyecto-ispc-fullstack/wiki/An%C3%A1lisis-y-dise%C3%B1o-del-sistema
- Código fuente del proyecto: https://github.com/rpgrca/proyecto-ispc-fullstack

Fe de erratas

- Nosotros utilizamos sitio como base para el html, cuando se nos informó que debíamos tener un directorio llamado view en su lugar lo renombramos lo que ocasionó que el historial del directorio se volviese inaccesible. Sin embargo es posible revisar el historial de cada uno de los archivos html para ver los autores de las versiones más antiguas (equivalente a utilizar *git log –follow*).



- Tuvimos un problema de merging de un commit y sobreescribió algunos historiales con un commit erróneo. Afectó principalmente a los archivos Python, si se revisa el historial el commit de la creación fue reemplazado por un commit de un renombre como si todos los archivos hubiesen sido creados a partir del mismo archivo. No sabemos cómo pasó. El contenido que muestra es distinto al que muestra *git log –follow* por alguna razón así que suponemos que es un error de Github.




```
roberto@desktop:~/src/proyecto-ispc-fullstack/model$ git log --follow esta
nteria.py
commit f6b5202c352920d6325924f717cc7b87a1a4b127
Author: Roberto <rpgrca@gmail.com>
Date:   Sat Oct 15 20:12:03 2022 -0300

    agregar metodos para creacion

commit c2104a500548ad96dd790174b690bb791674bdaa
Author: AhumadaCeleste <ahumadamariaceleste@gmail.com>
Date:   Sat Oct 15 09:41:15 2022 -0300

    a cerca de

commit d82e4c530e28bb7810c2dd83ac92eae429a47fef
Author: Roberto <rpgrca@gmail.com>
Date:   Thu Oct 13 22:50:03 2022 -0300

    agregar modelos y controladores
```

- No tenemos una distribución uno a uno entre páginas web en la vista y modelos o controladores. Existen páginas que reusarán controladores, habrán páginas que no tendrán controladores ni modelos asociados (como las de Quiénes somos o la de Cómo funciona), existen controladores que utilizarán distintos modelos y otros que utilizarán los mismos.