**Modelo de base de datos en ruby**

**Relaciones entre tablas**

Rails implementa las relaciones entre tablas mediante etiquetas :has\_one, :has\_many, :belongs\_to y :has\_and\_belongs\_to\_many Este último para relaciones muchos-a-muchos.

Al definir las relaciones entre tablas se crean una serie de metodos de instancia muy útiles en su posterior manejo.

Una ayuda muy útil es la que nos proporciona para mantener un contador automático y evitarnos constantes consultas del tipo "select count(\*)" mediante el uso de un campo \_count.

Otra de las ventajas de Rails es la herencia. Al trabajar con objetos podemos aprovechar la herencia para crear modelos de datos que comparten una única tabla de la base de datos. Por ejemplo: podemos tener una tabla llamada vehículos y crear nuestros modelos de datos de la siguiente forma:

*class vehiculo < ActiveRecord::Base Esto generaría el modelo de datos maestro. Gracias a la herencia ahora podemos hacer:*

*Turismo < Vehiculo*

*Furgoneta < Vehiculo*

Rails se encarga de diferenciar automáticamente los diferentes tipos de registros (Turismos, Furgonetas) gracias al empleo de un campo type (que habremos de definir en la tabla maestra).

**Validación**

Una de las opciones más útiles y potentes son los validadores. Mediante esta opción podremos verificar la coherencia e integridad de los datos.

Rails nos proporciona 3 eventos de validación: Validate, Validate\_on\_create y Validate\_on\_update donde podremos situar todo el código que deseemos para asegurarnos de la coherencia de los datos recibidos con los esperados (números, fechas, nombres, etc....). Además nos proporciona unos cuantos validadores predefinidos (helpers) que simplificaran las tareas de validación como ejemplo ( validates\_length\_of ó validates\_numericality\_of.

**Callbacks**

Los callbacks son a Rails como los Trigges a las base de datos. Existen una serie de métodos del tipo:

*-before\_validation*

*-after\_validaton*

*-before\_save*

*-after\_update*

*-before\_create*

*-after\_save*

*-etc.....*

en los que podremos insertar nuestro código. Rails nos permite crear callbacks comunes a varios modelos de datos. Recordemos que Rails es un framework DRY.

**Observers**

Los observers están pensados para añadir funcionalidad a los modelos de datos sin tener que modificar su código. Los métodos creados como observers se añaden al modelo de datos aunque sin llegar a pertenecer al mismo de manera que el modelo de datos no se ve afectado. También es posible crear observers comunes a varios modelos de datos.