

PRIMERA APLICACIÓN

```
$ rails
```

Usage:

```
rails new APP_PATH [options]
```

Options:

- O, [--skip-active-record] # Skip Active Record files
- d, [--database=DATABASE] # Preconfigure database
Default: sqlite3
- j, [--javascript=JAVASCRIPT] # Preconfigure JavaScript library
Default: jquery

Runtime options:

- f, [--force] # Overwrite files that already exist
- p, [--pretend] # Run but do not make any changes

Example:

```
rails new ~/Code/Ruby/weblog
```

PRIMERA APLICACIÓN

```
$ rails new primera_app
```

\$ rails new primera_app

create

create README.rdoc

create Rakefile

create config.ru

create .gitignore

create Gemfile

create app

.

.

.

create vendor/assets/stylesheets

create vendor/assets/stylesheets/.keep

run bundle install

.

.

.

Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.

Instala las
dependencias



primera_app				
Nombre				
	▲	Fecha de modificación	Tamaño	Clase
▶ app		hoy 17:31	--	Carpeta
▶ bin		hoy 17:31	--	Carpeta
▶ config		hoy 17:31	--	Carpeta
config.ru		hoy 17:31	154 bytes	Documento
▶ db		hoy 17:31	--	Carpeta
Gemfile		hoy 17:31	1 KB	Documento
Gemfile.lock		hoy 17:31	3 KB	Documento
▶ lib		hoy 17:31	--	Carpeta
▶ log		hoy 17:31	--	Carpeta
▶ public		hoy 17:31	--	Carpeta
Rakefile		hoy 17:31	255 bytes	Documento
README.rdoc		hoy 17:31	478 bytes	Documento
▶ test		hoy 17:31	--	Carpeta
▶ tmp		hoy 17:31	--	Carpeta
▶ vendor		hoy 17:31	--	Carpeta

BUNDLER

Se utiliza para instalar e incluir las **gems** necesarias por la aplicación (dependencias)

Las gems se especifican en el archivo Gemfile

```
gem 'rails', '4.0.8'

# Use sqlite3 as the database for Active Record
gem 'sqlite3'

# Use SCSS for stylesheets
gem 'sass-rails', '~> 4.0.2'
.
.
.
# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder
gem 'jbuilder', '~> 1.2'

group :doc do
  # bundle exec rake doc:rails generates the API under doc/api.
  gem 'sdoc', require: false
end

# Use ActiveSupport has_secure_password
# gem 'bcrypt', '~> 3.1.7'

# Use unicorn as the app server
# gem 'unicorn'

# Use Capistrano for deployment
# gem 'capistrano', group: :development

# Use debugger
# gem 'debugger', group: [:development, :test]
```

```
$ cd primera_app
```

```
$ rails
```

Usage: rails **COMMAND** [ARGS]

The most common rails commands are:

generate **Generate new code (short-cut alias: "g")**

console **Start the Rails console (short-cut alias: "c")**

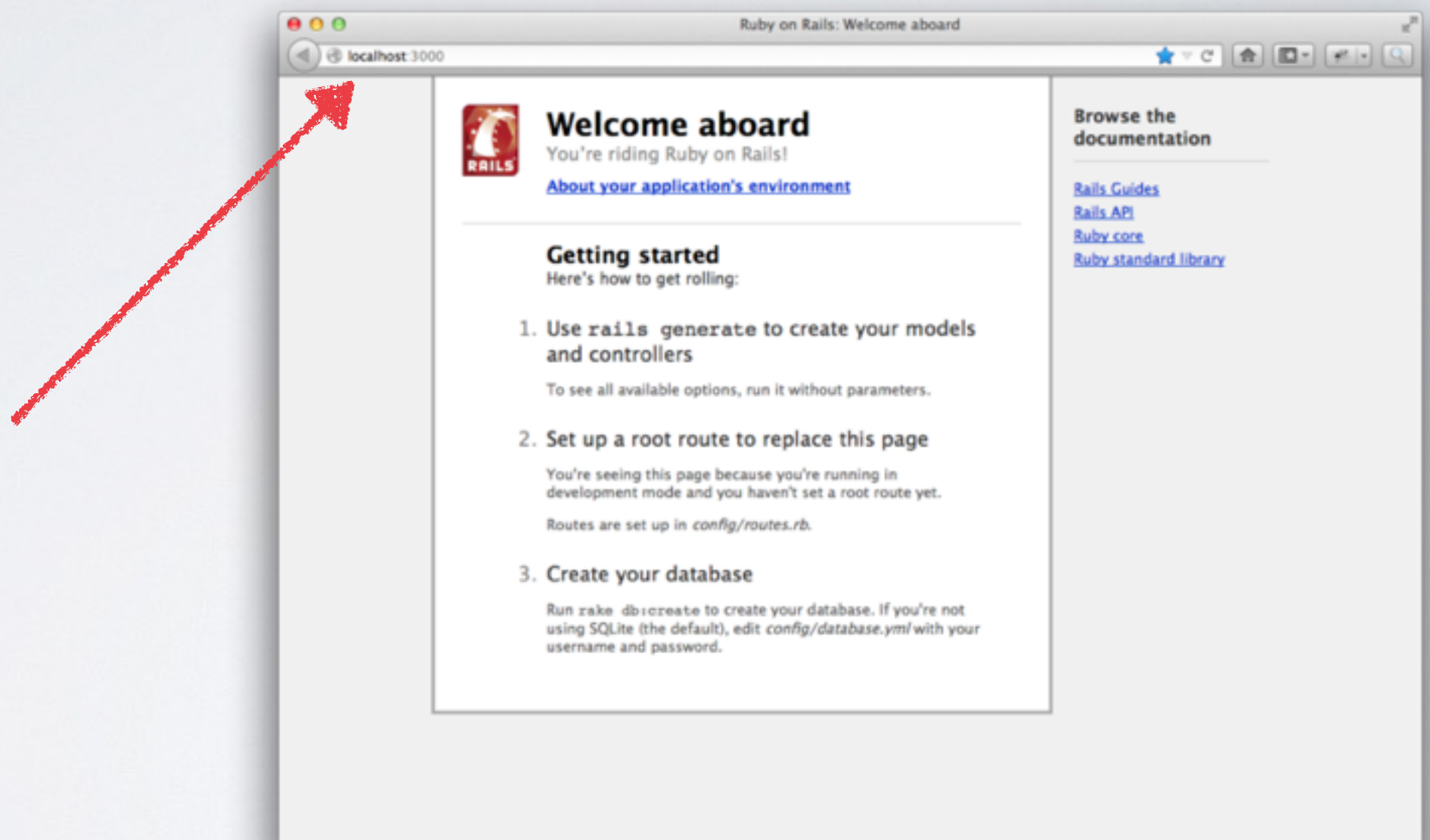
server **Start the Rails server (short-cut alias: "s")**

dbconsole **Start a console for the db in config/database.yml
(short-cut alias: "db")**

CORRIENDO LA APLICACIÓN

\$ rails server

localhost:3000



Atajo: *rails s*

\$ rails generate

Usage: rails generate GENERATOR [args]

Please choose a generator below.

Rails:

helper

mailer

migration

model

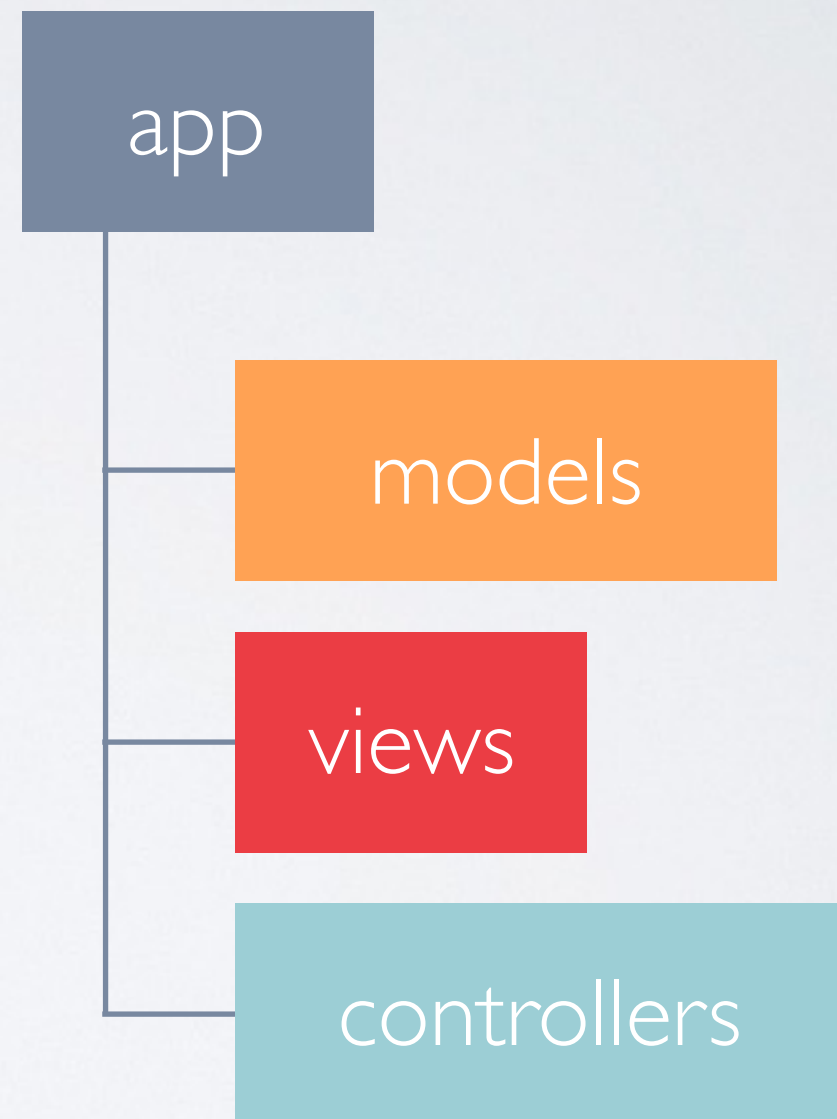
\$ rails console

- Habilita una consola para depurar nuestra aplicación.
- Podemos correr comandos en la BD

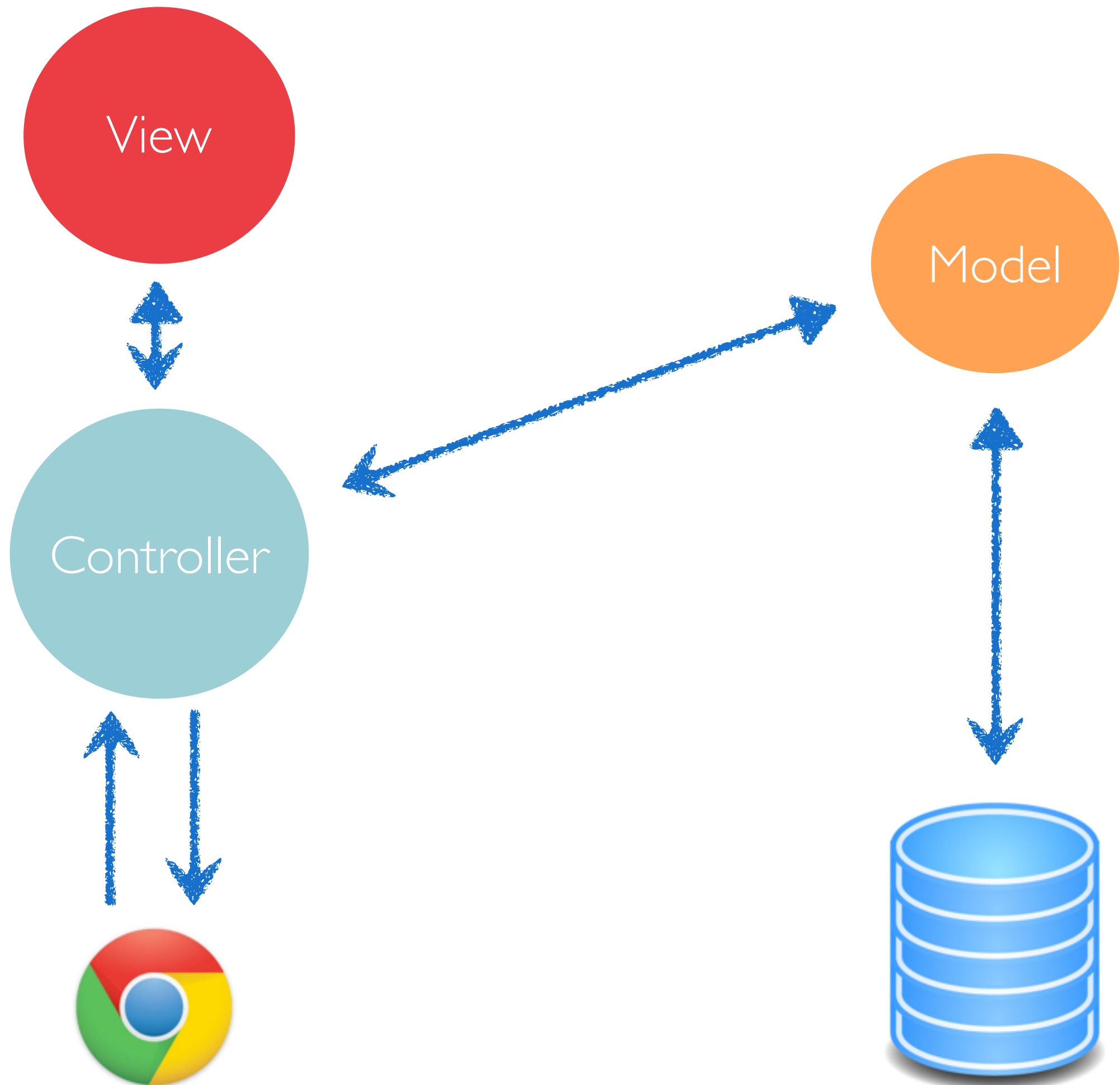
MVC

Separa la **lógica del dominio** de la **entrada y presentación de la información**

En el caso de los sistemas web la lógica del dominio está representada por modelos de datos (usuario, artículos, etc)



MVC



TWITTER

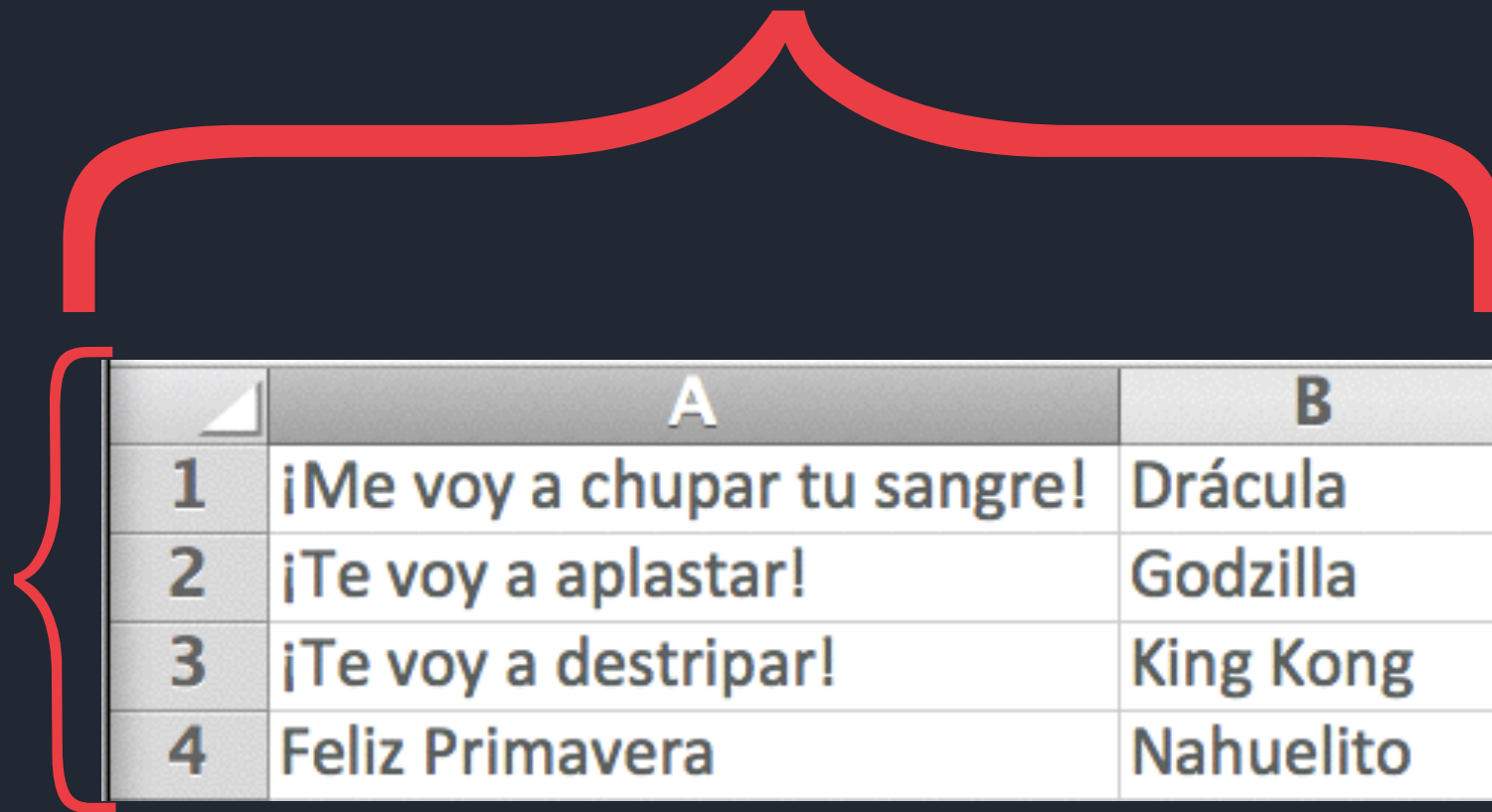
TWITTER  ~~C PARTIAL MONSTRUOS~~



COLUMNAS

(tenemos 3)

FILAS
(tenemos 4)



	A	B
1	¡Me voy a chupar tu sangre!	Drácula
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz Primavera	Nahuelito

tweets



id



estado



monstruo

id	estado	monstruo
1	Me voy a chupar tu sangre	Drácula
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz primavera	Nahuelito

**Queremos obtener
el tweet con id = 3**

HASH

serie de pares clave y valor

Una clave y valor

$a = \{ id: 3 \}$

Múltiples clave y valor

$b = \{ id: 3, \\ estado: "¡Te voy a destripar!", \\ monstruo: "King Kong" \}$

HASH

serie de pares clave y valor

CLAVES	VALORES
:id	3
:estado	¡Te voy a destripar!
:monstruo	King Kong



Símbolos

$b = \{$ *id: 3,*
estado: "¡Te voy a destripar!",
monstruo: "King Kong" }

```

b = { id: 3,
      estado: "¡Te voy a destripar!",
      monstruo: "King Kong" }
    
```

```

b[:estado]
    
```

```

=> "¡Te voy a destripar!"
    
```

```

b[:monstruo]
    
```

```

=> "King Kong"
    
```

```

b[:monstruo] + " dijo " + b[:estado]
    
```

```

=> "King Kong dijo ¡Te voy a destripar!"
    
```


id	estado	monstruo
1	Me voy a chupar tu sangre	Drácula
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz primavera	Nahuelito

**Queremos obtener
el tweet con id = 3**

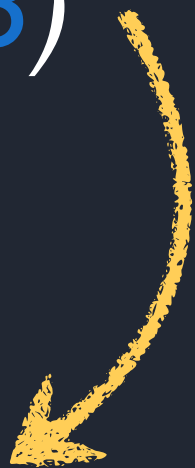
RESULTADO

$b = \{$ *id*: 3,
estado: "¡Te voy a destripar!",
monstruo: "King Kong" $\}$

Obtener el tweet con id = 3

RESPUESTA

```
t = Tweet.find(3)
```



RESULTADO

```
t = { id: 3,  
      estado: "¡Te voy a destripar!",  
      monstruo: "King Kong" }
```

tweets

Plural y en minúsculas

id	estado	monstruo
1	Me voy a chupar tu sangre	Drácula
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz primavera	Nahuelito

RESPUESTA

t = Tweet.find(3)

*El nombre de la tabla
singular y en mayúscula*

tweets

id	estado	monstruo
1	Me voy a chupar tu sangre	Drácula
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz primavera	Nahuelito

```
t = Tweet.find(3)
```

```
puts t[:id]
```

```
=> 3
```

```
puts t[:estado]
```

```
=> ¡Te voy a destripar!
```

```
puts t[:monstruo]
```

```
=> King Kong
```

puts *t[:id]*

=

puts *t.id*

puts *t[:estado]*

=

puts *t.estado*

puts *t[:monstruo]*

=

puts *t.monstruo*

NOTACIÓN HASH

NOTACIÓN PUNTO



¿Cual hay que utilizar? ¿La notación hash o punto?

Create

```
t = Tweet.new  
t.estado = "Tengo hambre"  
t.save
```

Read

```
t = Tweet.find(3)
```

Update


```
t = Tweet.find(3)  
t.monstruo = "Hombre lobo"  
t.save
```

Delete

```
t = Tweet.find(3)  
t.destroy
```


Create

```
t = Tweet.new  
t.estado = "Tengo hambre"  
t.estado = "Hombre lobo"  
t.save
```



El id se setea automáticamente

```
t = Tweet.new (  
  estado: "Tengo hambre",  
  monstruo: "Hombre lobo")  
t.save
```

```
t = Tweet.create (  
  estado: "Tengo hambre",  
  monstruo: "Hombre lobo")
```

Read

Tweet.find(3)

=> *Retorna el tweet con id 3*

Tweet.find(3, 4, 5)

=> *Retorna un array con los tweets con id 3, 4 y 5*

Tweet.first

=> *Retorna el primer tweet*

Tweet.last

=> *Retorna el último tweet*

Tweet.all

=> *Retorna un array con todos los tweets*

Read

READ

Tweet.count

=> *Retorna la cantidad de tweets*

Tweet.order(:monstruo)

=> *Retorna todos los tweets ordenados por monstruo*

Tweet.limit(10)

=> *Retorna los primeros 10 tweets*

Tweet.where(monstruo: "Drácula")

=> *Retorna los tweets de Drácula*

Se pueden combinar los métodos

Read

READ

Tweet.where(monstruo:“Drácula”).order(:estado).limit(10)

=> *Retorna los primeros 10 tweets de Drácula ordenados por estado*

Tweet.where(monstruo:“Drácula”).first

=> *Retorna el primer tweet de Drácula*

Update

UPDATE

```
t = Tweet.find(3)
t.monstruo = "Hombre lobo"
t.save
```

```
t = Tweet.find(3)
t.attributes = {
  estado: "¿Y Caperucita?",
  monstruo: "Hombre lobo"
}
t.save
```

```
t = Tweet.find(3)
t.update (
  estado: "Tengo hambre",
  monstruo: "Hombre lobo"
)
```

Delete

DELETE

```
t = Tweet.find(3)  
t.destroy
```

```
t = Tweet.find(3).destroy
```

```
t = Tweet.where("monstruo = 'Dracula']").destroy_all
```


LABORATORIO



<http://cor.to/ingcomplab1>