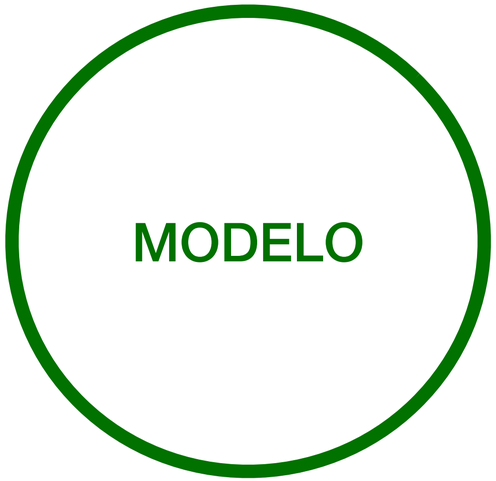
Ingeniería de Software 2021

Laboratorio 2 - Modelos

MVC: Model View Controller

Capas de la aplicación

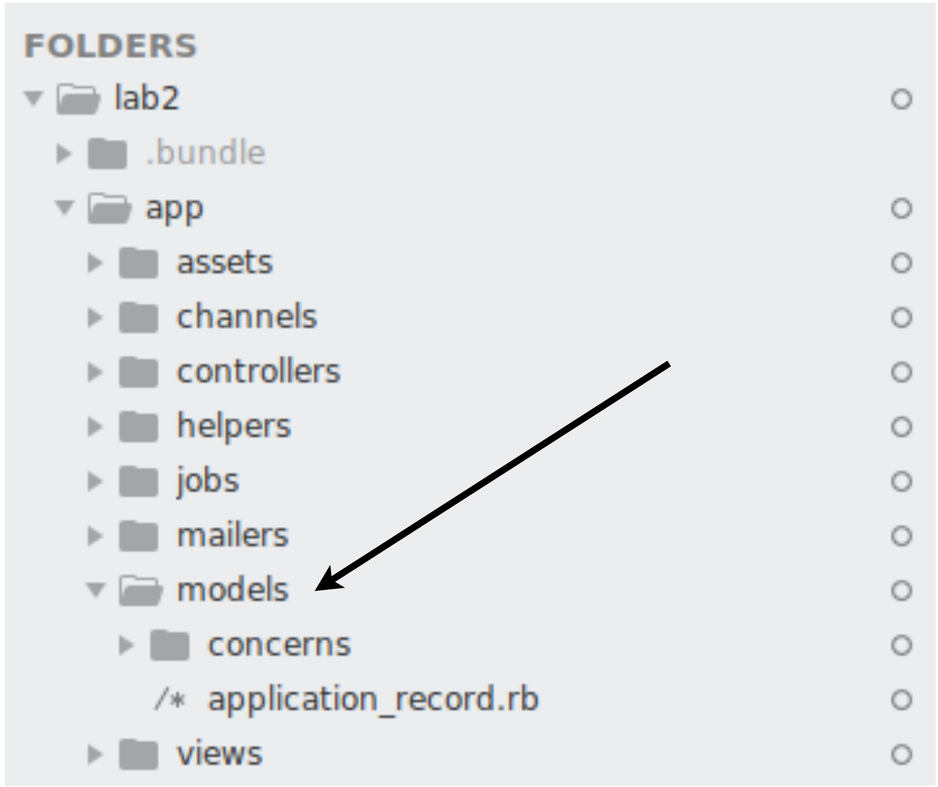


Accesos a la BD

Funciona de intermediario entre el modelo y las vistas

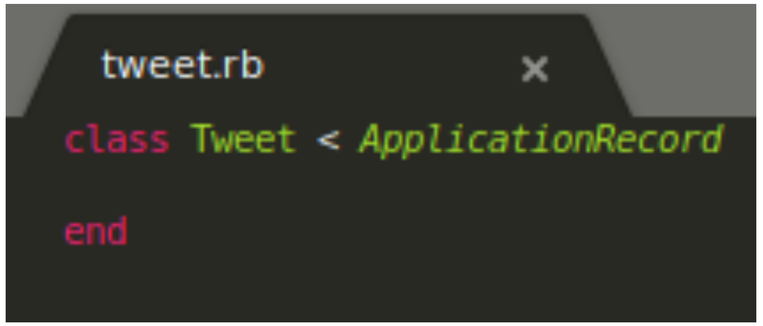
Representación visual de la información (usuario final)

MODELOS EN



Podemos crear el archivo manualmente o usando el comando rails g

app/models/tweet.rb



app/models/tweet.rb

tweet.rb

```ruby
class Tweet < ApplicationRecord

end
```

Mapea la clase con la tabla (Tweet - tweets)

Tabla: 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

app/models/tweet.rb

tweet.rb

```ruby
class Tweet < ApplicationRecord

end
```

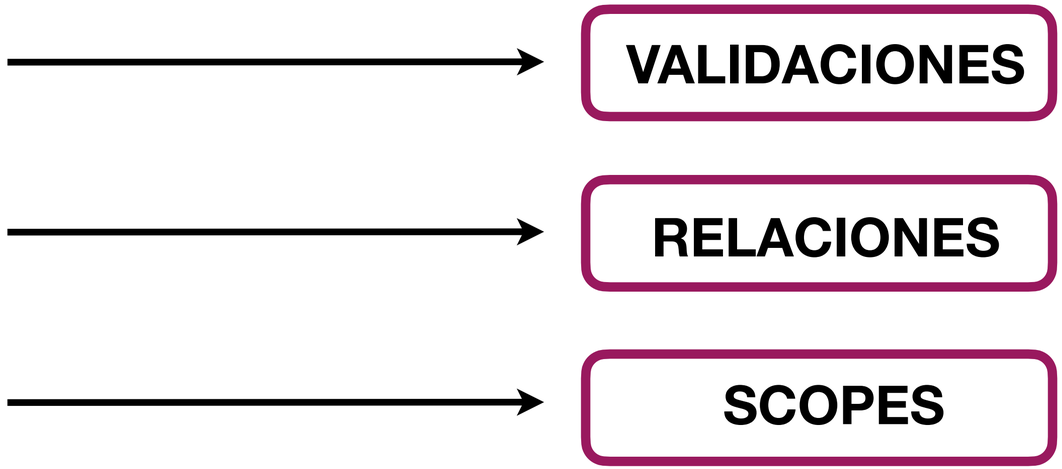


Tabla: tweets

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

VALIDACIONES

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
5		

```
t = Tweet.new
t.save
```

tweet.rb

```ruby
class Tweet < ApplicationRecord
  validates :estado, presence: true
end
```

rails c

```
t = Tweet.new
 => #<Tweet id: nil, estado: nil, monstruo: nil>
t.save
 => false
t.errors.messages
 => {estado: ["no puede estar en blanco"]}
```

Otras validaciones

```
validates :estado,
          presence: true,
          uniqueness: true,
          numericality: true,
          length: { minimum: 0, maximum: 2000 }
          format: { with: /.*/ }
```

Podemos especificar validaciones para cualquier atributo de la tabla asociada al modelo

```
validates :estado, presence: true

validates :brazos, numericality: true

validates :huella, uniqueness: true

validates :clave, length: { minimum: 3, maximum: 20 }

validates :age, inclusion: { in: 21..99 }

validates :age, exclusion: { 0...21 }, message: "Debes ser mayor a 21"
```

RELACIONES

tweets

id	estado	monstruo
1	Barilo Barilo, nos vamo a Barilo	Nahuelito
2	¡Te voy a aplastar!	Godzilla
3	¡Te voy a destripar!	King Kong
4	Feliz Primavera	Nahuelito

Queremos guardar los monstruos en su propia tabla

tweets

id	estado	monstruo_id
1	Barilo Barilo, nos vamo a Barilo	1
2	¡Te voy a aplastar!	2
3	¡Te voy a destripar!	3
4	Feliz Primavera	1

monstruos

id	nombre	descripcion
1	Nahuelito	Vive en el Nahuel Huapi
2	Drácula	Chupa sangre
3	King Kong	Es un gorila gigante
4	Godzilla	¿Es japonés?

tweet.rb

```ruby
class Tweet < ApplicationRecord
  belongs_to :monstruo
end
```

Singular
Un tweet va a tener un sólo monstruo asociado (belongs_to)

monstruo.rb

```ruby
class Monstruo < ApplicationRecord
  has_many :tweets
end
```

Plural
Un monstruo puede tener 0 o más tweets asociados (has_many)

rails c

```
t = Tweet.create(estado: "Yendo a la Abuela Goye”, monstruo_id: 1)
```

Usando las relaciones definidas

rails c

```
nahuelito = Monstruo.find(1)

t = Tweet.create(estado: "Yendo a la Abuela Goye", monstruo: nahuelito)
```

```
nahuelito.tweets.count
  => 3
nahuelito.tweets
  => [#<Tweet id: 1, estado: "Barilo Barilo, nos vamo a Barilo", monstruo_id: 1>,
      #<Tweet id: 4, estado: "Feliz Primavera", monstruo_id: 1>,
      #<Tweet id: 5, estado: "Yendo a la Abuela Goye", monstruo_id: 1>]
```

rails c

Otros usos de las relaciones definidas

rails c

```
t = Tweet.find(5)
    => #<Tweet id:5, estado: "Yendo a la Abuela Goye", monstruo_id: 1>
t.monstruo
    => #<Monstruo id: 1, nombre: "Nahuelito", descripcion: "Vive en el
Nahuel Huapi">
t.monstruo.nombre
    => "Nahuelito"
```

Borrando asociaciones

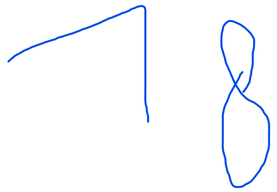
rails c

```
Tweet.find(1,4,5)
    => [#<Tweet id: 1, estado: "Barilo Barilo, nos vamo a Barilo", monstruo_id: 1>,

        #<Tweet id: 4, estado: "Feliz Primavera", monstruo_id: 1>,

        #<Tweet id: 5, estado: "Yendo a la Abuela Goye", monstruo_id: 1>]
```

Borrando asociaciones

monstruo.rb

```ruby
class Monstruo < ApplicationRecord
  has_many :tweets, dependent: :destroy
end
```



rails c

```
m = Monstruo.find(1)
    => #<Monstruo id: 1, nombre: "Nahuelito", descripcion: "Vive en el Nahuel Huapi">

m.destroy
    => true
Tweet.find(1,4,5)
    => []
```

SCOPES

Permiten generar consultas de datos de nuestro modelo de forma simple y legible

```ruby
class Tweet < ApplicationRecord
  default_scope -> { order(created_at: :desc) }
  scope :recientes, -> { order("created_at desc").limit(3) }
  scope :eliminados, -> { where(eliminado: true) }
end
```

Tweet.all	*Retorna los tweets ordenados por fecha descendente*
Tweet.recientes	*Últimos 3 tweets*
Tweet.eliminados	*Tweets eliminados*

MODELO FINAL

tweet.rb

```ruby
class Tweet < ApplicationRecord

  # Scopes
  default_scope -> { order(created_at: :desc) }
  scope :recientes, -> { order("created_at desc").limit(3) }
  scope :eliminados, -> { where(eliminado: true) }

  # Validaciones
  validates :estado, presence: true, uniqueness: true

  # Relaciones
  belongs_to :monstruo

  # Métodos
  ...

end
```