**Apache**

; Abrimos otra consola para desplegar apache

$ sudo a2enmod proxy

$ sudo a2enmod proxy\_http

$ sudo vim /etc/apache2/sites-available/000-default.conf

; En el archivo comentamos la línea

# DocumentRoot /var/www/html

; Y agregamos

ProxyPass / <http://localhost:3000/>

ProxyPassReverse / <http://localhost:3000/>

$ sudo service apache2 restart

**Rails**

$ rails new railsgirls -d postgresql

$ cd railsgirls

$ rake db:create

$ rails s

! Mi URL: <https://leogavidia-1.paiza-user.cloud/>

! SI el servidor postgresql no está activo, lo activamos con:

$ sudo service postgresql start

! SI la base lanza un error indicando que el usuario *ubuntu* no existe entonces hacemos lo siguiente:

$ sudo su - postgres

$ psql

postgres=# CREATE USER ubuntu CREATEDB;

Ctrl+D ; Salimos de psql

Ctrl+D ; Salimos del usuario postgresql

**Requerimiento 1:**

Desarrollar una aplicación que permita registrar estudiantes y asignarle deudas, es necesario poder trabajar con la información vía un API JSON

; Tipos soportados por *scaffold*: <http://overooped.com/post/100354794/ruby-scriptgenerate-scaffold-types>

$ rails g scaffold student document\_number:string name:string email:string phone:string address:text

$ rake db:migrate

; Ya podemos ingresar a <https://leogavidia-1.paiza-user.cloud/students>

! Si el Modelo se llama STUDENT la tabla se llama STUDENT**S** (Convención de pluralización)

! SIngular: Modelos

! Plural: Tablas, Controladores (y Vistas)

**Requerimiento 2:**

Crear un modelo DEUDA (debt) con los campos ESTUDIANTE (student) FECHA (date), FECHA\_MAXIMA (date\_due), MONTO (amount), DESCRIPCION (description)

$ rails g scaffold debt student:references date:date date\_due:date amount:decimal description:text

$ rake db:migrate

**Requerimiento 3:**

Crear un modelo METODO\_DE\_PAGO (payment\_method) con el campo NOMBRE (name)

$ rails g scaffold payment\_method name:string

$ rake db:migrate

**Requerimiento 4:**

Crear un modelo PAGO (payment) con los campos METODO\_DE\_PAGO(payment\_method), ESTUDIANTE (student) FECHA (date), MONTO (amount), descripcion (description)

$ rails g scaffold payment payment\_method:references student:references date:date amount:decimal description:text

$ rake db:migrate

**Requerimiento 5:**

Crear una tabla que concilie pagos con deudas (payment\_debt) con los campos PAGO (payment\_id), DEUDA (debt) y MONTO (amount)

$ rails g scaffold payment\_debt payment:references debt:references amount:decimal

$ rake db:migrate

**Requerimiento 6:**

Agregar el campo SALDO (residual) en las tablas de pagos y deudas

$ rails g migration add\_residual\_to\_debts residual:decimal

$ rails g migration add\_residual\_to\_payments residual:decimal

$ rake db:migrate

## Modelos

***Validaciones***

; Tutorial <https://guides.rubyonrails.org/active_record_validations.html>

***Migraciones***

; Tutorial: <https://edgeguides.rubyonrails.org/active_record_migrations.html>

***Callbacks***

; Tutorial: https://guides.rubyonrails.org/active\_record\_callbacks.html

; Ingresamos al modelo ~/railsgirls/app/models/student.rb

class Student < ApplicationRecord

# Creamos la relacion con el modelo Debt

# Esto nos permite usar el método *student* para acceder al objeto

# relacionado al campo *student\_id*

has\_many :debts

# Agregamos validadores

validates :document\_number, length: {is: 10}

validates :document\_number, numericality: {only\_integer: true}

validates :name, length: {minimum: 5, maximum: 100}

validates :phone, length: {minimum: 9, maximum: 10}

# SI instalamos phonelib podremos usar phone: true

# validates :phone, phone: true

validates :address, presence: true

validates :email, format: { with: URI::MailTo::EMAIL\_REGEXP }

# Es importante tener campos que nos ayuden a identificar facilmente

# los registros

def display\_name

"#{document\_number} - #{name}"

end

end

; Ingresamos al modelo ~/railsgirls/app/models/debt.rb

class Debt < ApplicationRecord

belongs\_to :student

validates :student, :date, :date\_due, :description, presence: true

validates :amount, numericality: {greater\_than: 0}

validate :date\_due\_cannot\_be\_in\_the\_past

def date\_due\_cannot\_be\_in\_the\_past

if date.present? and date\_due.present? and date\_due < date

errors.add(:date\_due, "bust be greater or equal than debt date")

end

end

end

## Relaciones entre modelos en las vistas

Cuando creamos con scaffold relaciones entre modelos (mediante el tipo *references*) en las vistas se crea un text input donde nos solicita ingresar un ID, y se usará el ID crudo en TODO, incluso en los campos que muestran la información

***; Por ejemplo en la vista de lista:***

; app/views/debts/index.erb

<td><%= debt.student %></td>

; El campo student es de tipo <Student>, por lo que disponemos de todas sus características

<td><%= debt.student.display\_name %></td>

***; Por ejemplo en la vista show:***

<%= @debt.student.display\_name %>

; Podemos reemplazar el campo de tipo text por un select, para fines practicos podemos usar collection\_select

; Reemplazamos:

<%= form.text\_field :student\_id %>

; por:

<%= form.collection\_select :student\_id, Student.all, :id, :display\_name %>

## Instalando gemas

; Editamos el archivo Gemfile

; Vamos a instalar PhoneLib la cual nos permitirá tener un validador extra (phone: true)

gem 'phonelib'

$ bundle install

EJERCICIO COMPLETO: <https://drive.google.com/file/d/13nKxeCGyL6zN2LaV_R4REVFvxfQyHA_d/view?usp=sharing>