

Ingeniería de Software

RoR - First API

Juan Pablo Sandoval

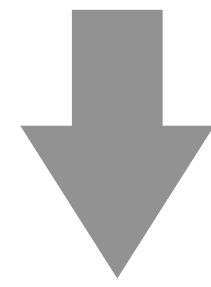
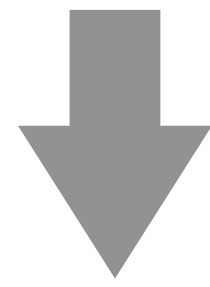
RoadMap

- *Web y HTTP*
- *Postman*
- *Hello World - API*
- *API - Estudiantes (crear, mostrar, listar)*
- *Ejercicio para la casa*

00110010 00110000 00110010 0011

50

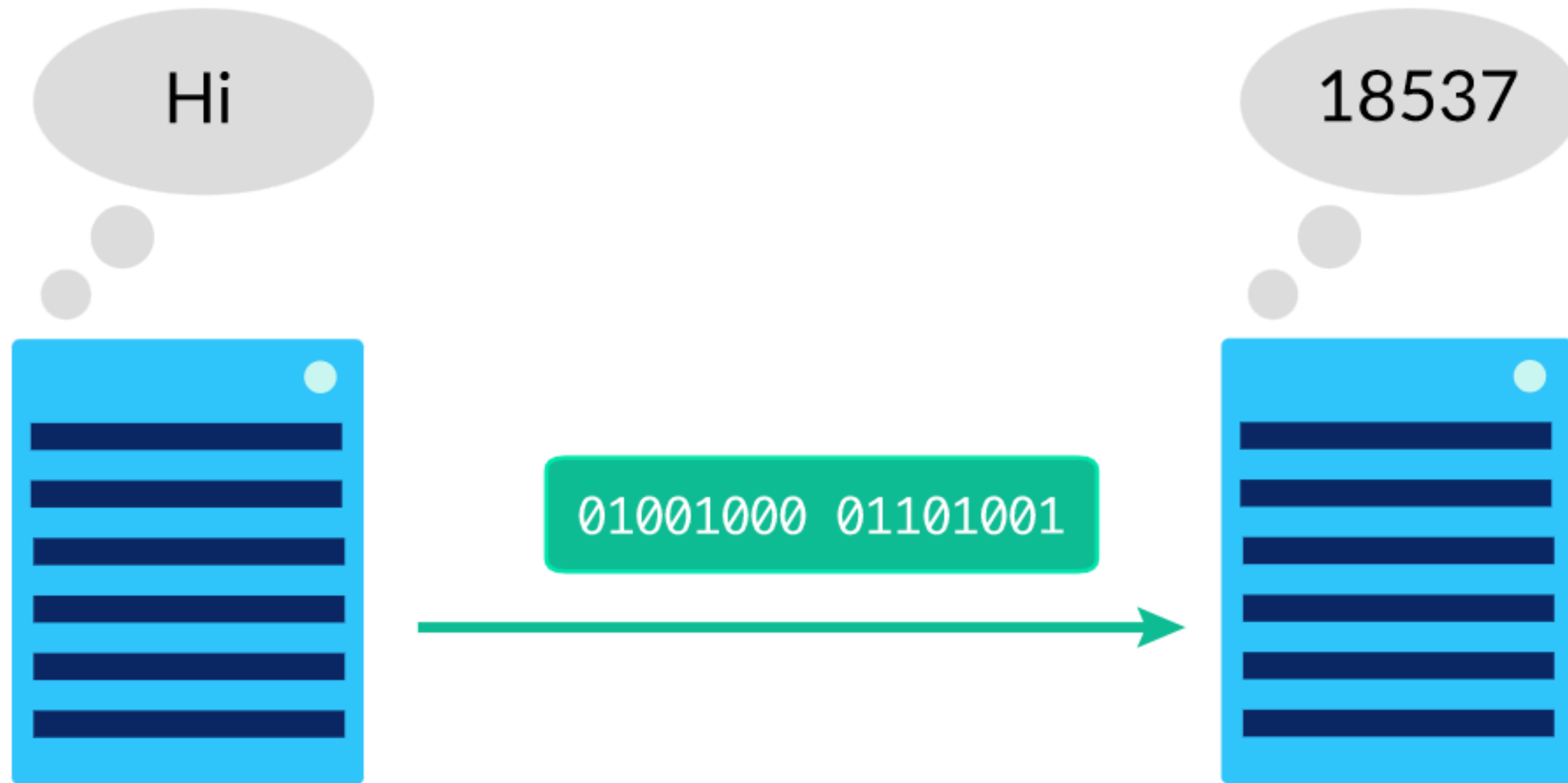
48



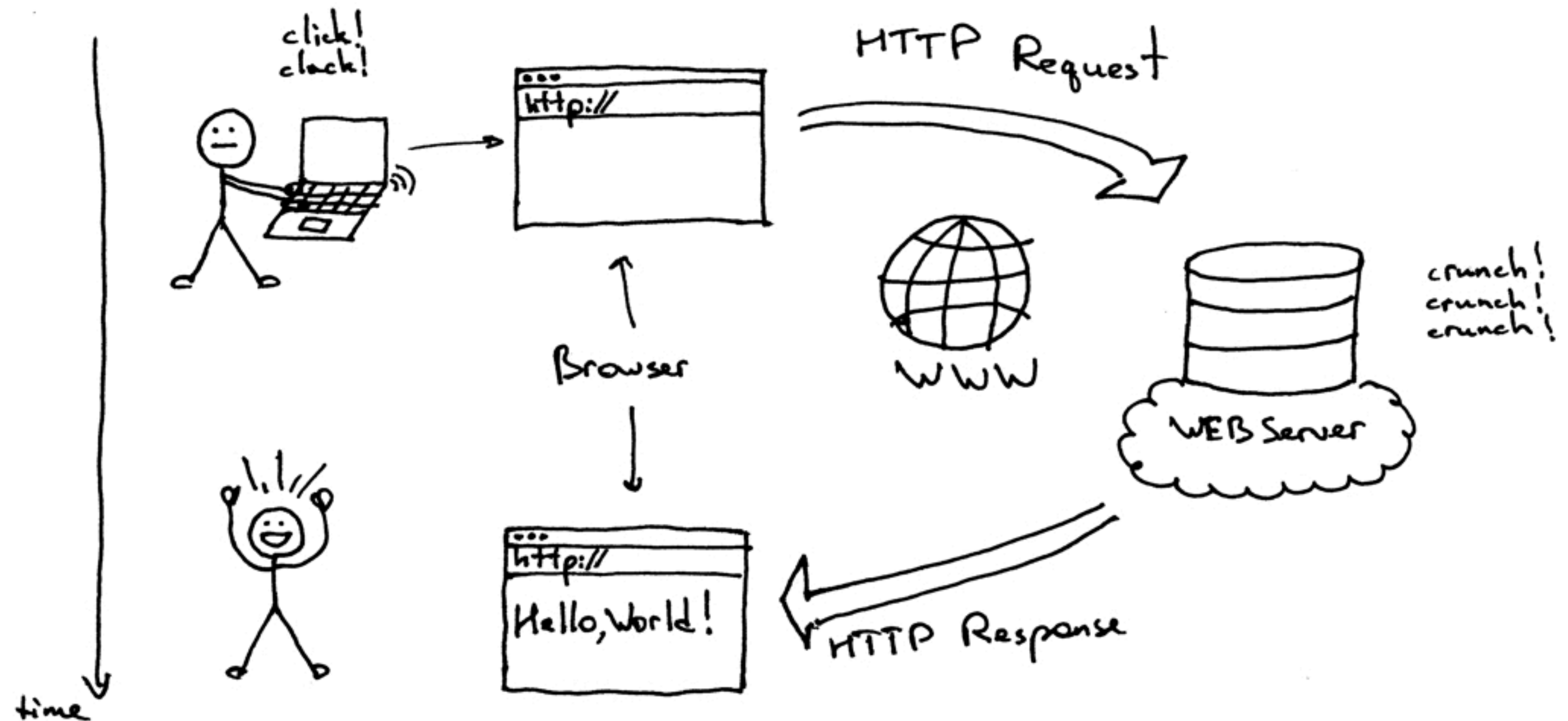
00110010 00110000 00110010 0011

“2022”

Desafortunadamente, la computadora B piensa que está recibiendo un número e interpreta el mensaje como el número decimal 18537.



Protocollo HTTP



HTTP Request

Method



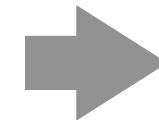
Target



Version



Headers



```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

Body



HTTP Response

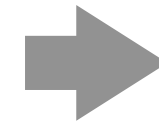
Version



Status



Headers

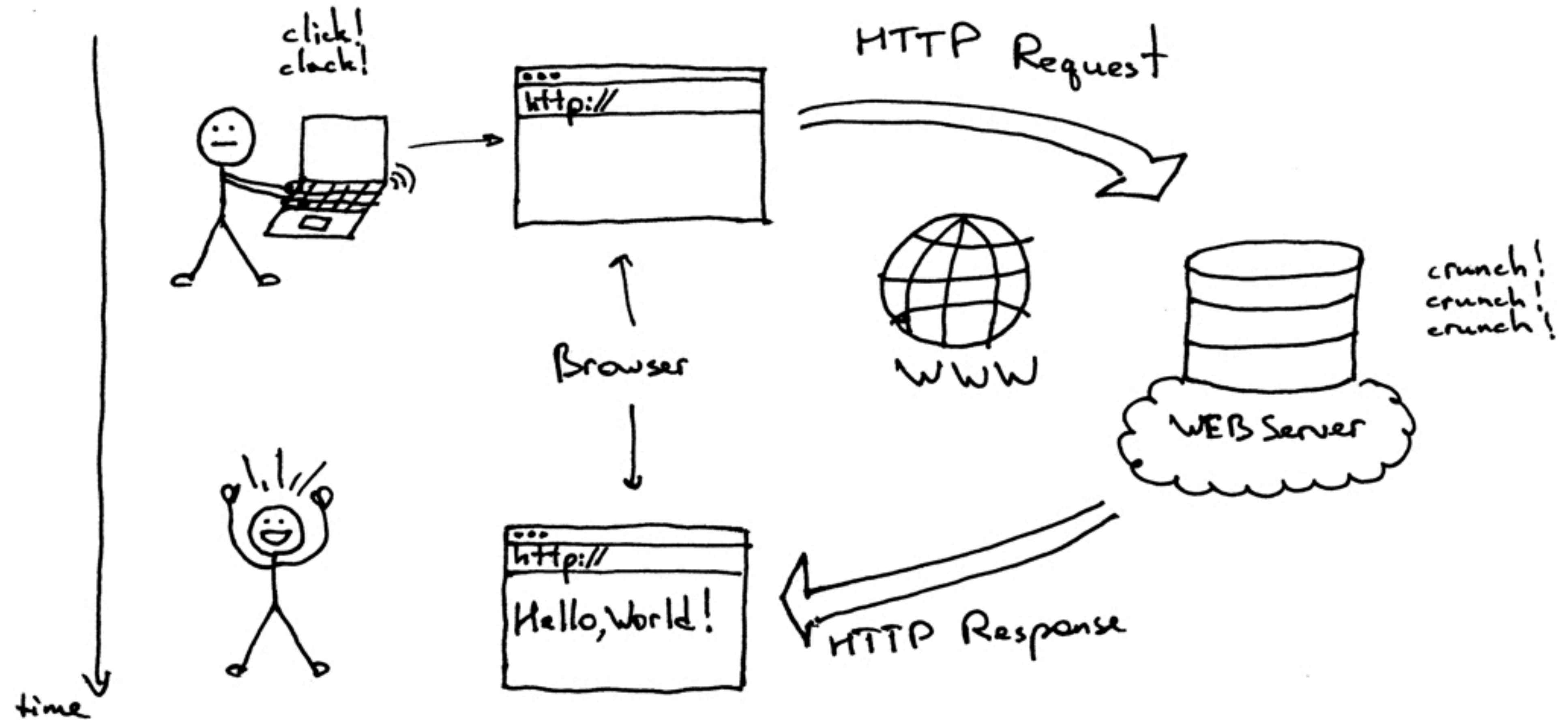


```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

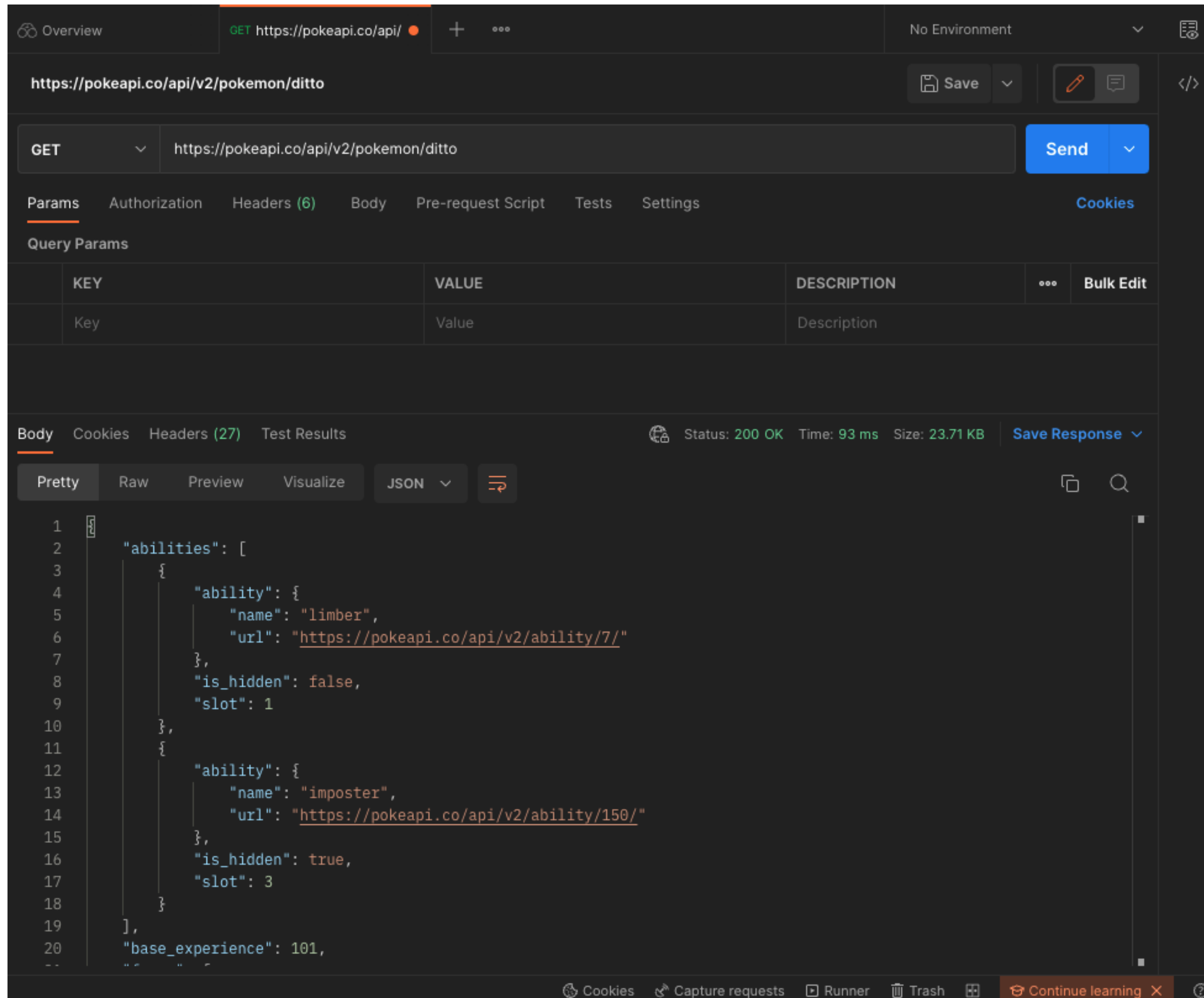


Body

Protocollo HTTP



Consumiendo el “Poke API” con Postman



The screenshot shows the Postman interface with a GET request to `https://pokeapi.co/api/v2/pokemon/ditto`. The request is saved and ready to be sent. The response is displayed in the 'Body' tab, showing a JSON object with abilities and base experience.

Request:

- Method: GET
- URL: `https://pokeapi.co/api/v2/pokemon/ditto`

Response:

```
1  {
2    "abilities": [
3      {
4        "ability": {
5          "name": "limber",
6          "url": "https://pokeapi.co/api/v2/ability/7/"
7        },
8        "is_hidden": false,
9        "slot": 1
10     },
11     {
12       "ability": {
13         "name": "imposter",
14         "url": "https://pokeapi.co/api/v2/ability/150/"
15       },
16       "is_hidden": true,
17       "slot": 3
18     }
19   ],
20   "base_experience": 101,
21   "..."
22 }
```

HTTP Request

HTTP Response



POSTMAN

RoadMap

- *Web y HTTP*
- *Postman*
- ***Hello World - API***
- *API - Estudiantes (crear, mostrar, listar)*
- *Ejercicio para la casa*

Hello World API

Verificando qué tenemos ruby y rails instalado.

```
>ruby --version
```

```
ruby 3.1.2p20
```

```
>rails --version
```

```
Rails 7.0.4.2
```

Hello World API

Creando un proyecto en Rails modo API

```
> rails new hello --api
```

Ingresamos al folder recién creado

```
> cd hello
```

Hello World API

Creando un controlador

```
> rails generate controller Hello
```

Código generado en el archivo “app/controllers/hello_controller.rb”

```
class HelloController < ApplicationController  
  
end
```

Hello World API

Agregando la operación “sayHi”

```
class HelloController < ApplicationController
  def sayHi
    render json: 'Hello World!!'
  end
end
```

Hello World API

Enlazando la URL “<http://localhost:3000/hi>” con la operación **sayHi**

```
Rails.application.routes.draw do
  get 'hi', to: 'hello#sayHi'
end
```

Archivo “config/routes.rb”

Hello World API

Ejecutando el servidor y probando con Postman

```
>rails server
```

The screenshot shows the Postman application interface. At the top, the URL bar displays 'localhost:3000/hi'. Below this, the request method is set to 'GET'. The 'Send' button is visible. The 'Params' tab is selected, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The 'Body' tab is also visible, showing the response body 'Hello World!!'. The status bar at the bottom indicates a successful response with status '200 OK', time '83 ms', and size '595 B'.

KEY	VALUE	DESCRIPTION

Body: Hello World!!

Status: 200 OK Time: 83 ms Size: 595 B



RoadMap

- *Web y HTTP*
- *Postman*
- *Hello World - API*
- ***API - Estudiantes (crear, mostrar, listar)***
- *Ejercicio para la casa*

Creando el Modelo

Creando un modelo Student en Rails:

```
> rails generate model Student name score
```

Código generado por el comando anterior “app/models/student.rb”:

```
class Student < ApplicationRecord  
  
end
```

Creando el modelo

```
> rails generate model Student name score
```

El comando anterior también genera un script que permite crear una tabla en la base de datos que permitirá almacenar la información de los estudiantes.

```
class CreateStudents <
  ActiveRecord::Migration[7.0]
    def change
      create_table :students do |t|
        t.string :name
        t.string :score
        t.timestamps
      end
    end
  end
```

La tabla estudiantes tendrá las siguiente columnas:

- *id, agregada por defecto, se autorellena (empieza en 1)*
- *name, score (atributos del modelo)*
- *created_at, updated_at (debido al timestamps)*

Creando la base de datos (BD)

Creando la base de datos para guardar estudiantes

```
> rails db:migrate
```

Output generado por el comando anterior

```
== 20230312193956 CreateStudents: migrating
=====
-- create_table(:students)
   -> 0.0007s
== 20230312193956 CreateStudents: migrated (0.0007s)
=====
```

Creando y recuperando estudiantes de la BD

Creando un objeto estudiante y guardándolo en la base de datos.

```
student = Student.new(name:"Juan Pablo", score:100)
result = student.save
```

Recuperando al estudiante que tiene el id = 1.

```
student = Student.find(1)
```

Recuperando todos los estudiantes guardados en la base de datos

```
all_students = Student.all
```

Recuperando estudiantes de la BD

Recuperando todos los estudiantes guardados en la base de datos

```
all_students = Student.all
```

Resultado:

```
[#<Student:0x000000010a1c7478  
  id: 1,  
  name: "Juan Pablo",  
  score: "100",  
  created_at: Sun, 12 Mar 2023 19:53:07.784504000 UTC +00:00,  
  updated_at: Sun, 12 Mar 2023 19:53:07.784504000 UTC +00:00>]
```

Creando un Controlador

Creando un controlador Student en Rails:

```
> rails generate controller Student
```

Código generado por el comando anterior “app/models/student.rb”:

```
class StudentController < ApplicationController  
  
end
```


API: Listando todos los estudiantes



POSTMAN

```
class StudentController < ApplicationController
  def index
    @all_students = Student.all
    render json:@all_students
  end
end
```

```
Rails.application.routes.draw do
  get 'index', to:'student#index'
end
```

API: Mostrando un estudiante

```
class StudentController < ApplicationController
  def show
    @student = Student.find(params[:id])
    render json: @student
  end
end
```

```
Rails.application.routes.draw do
  get 'index', to: 'student#index'
  get '/student/:id', to: 'student#show'
end
```

API: Creando un estudiante

```
class StudentController < ApplicationController
  def create
    @student = Student.new(student_params)
    if @student.save
      render json: @student
    else
      render json: @student.errors, status: :unprocessable_entity
    end
  end

  def student_params
    params.require(:student).permit(:name, :score)
  end
end
```

```
Rails.application.routes.draw do
  get 'index', to: 'student#index'
  get '/student/:id', to: 'student#show'
  post '/student', to: 'student#create'
end
```

Ejercicio para la casa (opcional)

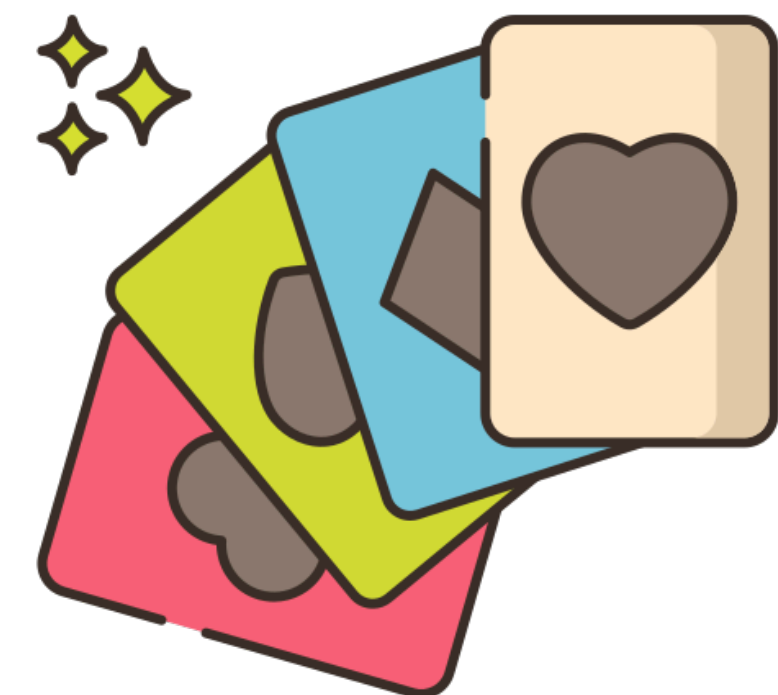
Ejercicio:

- ➡ Agregue la operación **filter** al controlador Student, el método filter debe recibir un numero como argumento, y retornar en formato JSON los datos de los estudiantes que tienen una nota igual al argumento del request (**params[:score]**).
- ➡ Crea la request en Postman para demostrar que el método filter funciona correctamente.

Entregable

- ➡ Debe subir a canvas en un solo archivo pdf:
- ➡ El código del controlador implementado (student_controller.rb)
- ➡ El código del archivo de rutas (routes.rb)
- ➡ Un screenshot de Postman donde se muestre que el API funciona!

Los que entreguen este ejercicio tendrán una décima extra para la I1!



Material Adicional

- Documentación de Ruby on Rails: https://guides.rubyonrails.org/getting_started.html
- Documentación Active Record: https://guides.rubyonrails.org/active_record_basics.html
- Documentación migraciones: https://guides.rubyonrails.org/active_record_migrations.html