Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 - Ingeniería de Software 2023-1

# Tarea Introduccion a *Rails*

En clase se desarrolló una API que involucraba un controlador Students y un modelo Student. También se configuró el archivo de rutas de tal forma que se podían realizar operaciones (mostrar, crear, editar, borrar) sobre la información de los estudiantes. En esta tarea se pide implementar una API que permita realizar operaciones sobre la información de personas y vacunas.

- Aprender a utilizar el *framework Ruby on Rails*.
- Entender e implementar el concepto de *API*.

## Vaccine API

En esta tarea usted tendrá que crear una API para registrar, crear, actualizar, y borrar personas y sus correspondientes vacunas. Para lo cual deberá implementar las siguientes rutas:

```
# permite recuperar todas las personas
get "/persons", to: "persons#index"
# permite registrar una persona en la base de datos
post "/persons", to: "persons#create"
# permite recuperar los datos de una persona en base al id
get "/persons/:id", to: "persons#show"
# permite borrar todas las personas de la base de datos
delete "/persons", to: "persons#destroy_all"

# permite recuperar todas las vacunas de una persona
get "/persons/:person_id/vaccines", to: "vaccines#index"
# permite registrar una vacuna
post "/persons/:person_id/vaccines", to: "vaccines#create"
# permite actualizar la informaci n de una vacuna
patch "/persons/:person_id/vaccines/:id", to: "vaccines#update"
#permite recuperar la informacion de una vacuna especifica
delete "/persons/:person_id/vaccines/:id", to: "vaccines#destroy"
```

A continuación se detalla las clases de modelo que deben crear con sus respectivos atributos. Además, se detalla los controladores a crear y sus métodos.

**Model Person.** El modelo tiene como atributo `name` de tipo `text`. Además, tenga en cuenta que el modelo involucra:

- *Asociaciones* – Una persona puede tener muchas vacunas (`has_many :vaccines`). Note que debido a esta asociación, debe tener cuidado al eliminar personas de la base de datos (hint: `dependent: :destroy`).
- *Validaciones* – Una persona tiene un atributo que no puede ser vacío ni nulo (hint: `validates :name, presence: true`).

**Model Vaccine.** El modelo tiene como atributos `vaccine_type` de tipo `text`, `vaccine_date` de tipo `date` y `person` de tipo `references`. Además, tenga en cuenta que el modelo involucra:

- *Asociaciones* – Una vacuna se aplica a una persona (`belongs_to :person`).
- *Validaciones* – Una vacuna tiene atributos que no pueden ser vacíos ni nulos (hint: `validates :vaccine_type, presence: true`).

**Controller Persons.** Una clase que tiene definidos los métodos:

- *index* – Método que muestra en formato JSON todas las personas de la base de datos.
- *create* – Método que instancia una nueva persona con la información del nombre y lo guarda en la base de datos. Note que la información necesaria para crear una nueva persona se pasa por medio de los parámetros (`params`). En caso de que la instancia se pueda guardar en la base de datos, este método muestra en formato JSON esa persona, caso contrario retorna un error (`render json: @person.errors, status: :unprocessable_entity`).
- *show* – Método que, dado el `id` de una persona, muestra en formato JSON la información de esa persona.
- *destroy_all* – Método que elimina a todas las personas de la base de datos y muestra en formato JSON la información de `Person.all`.

**Controller Vaccines.** Una clase que tiene definidos los métodos:

- *index* – Método que dado el parámetro del `person_id`, muestra en formato JSON todas las vacunas de la persona con ID igual a `params[:person_id]`.
- *create* – Método que dado el parámetro del `person_id`, instancia una nueva vacuna con la información del tipo de vacuna (vaccine_type) y la fecha (vaccine_date) y guarda esa instancia en las vacunas de la persona con ID igual a `params[:person_id]` en la base de datos. Note que información necesaria para crear una nueva vacuna se pasa por medio de los parámetros (`params`). Este método muestra en formato JSON las vacunas de la persona.
- *update* – Método que, dado el `id` de una persona, muestra en formato JSON las vacunas de la persona con ID igual a `params[:person_id]`.
- *destroy* – Método que elimina la vacuna cuyo ID sea igual al valor del parámetro `id` y person_id sea igual al valor del parámetro `person_id` de la base de datos. Este método muestra en formato JSON las vacunas de la persona cuyo ID es igual al valor del parametro `person_id` después de borrar la vacuna indicada.

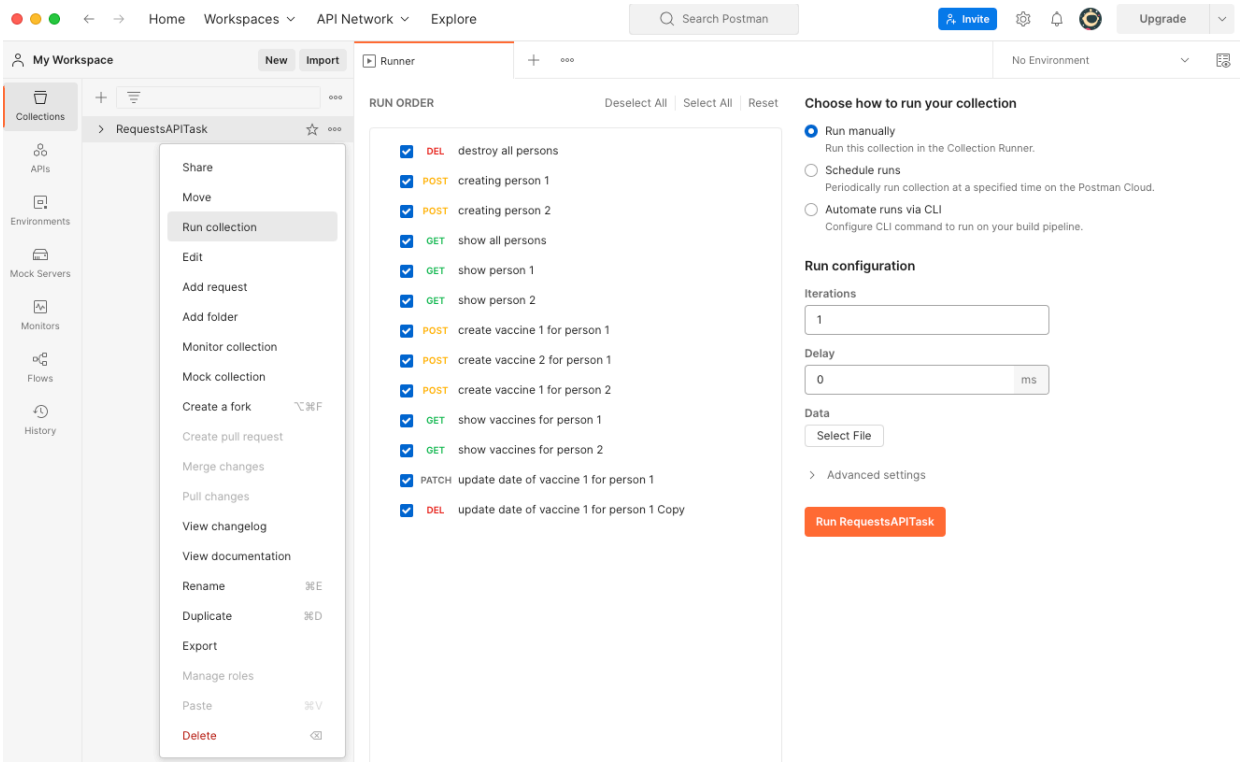
# Correctitud

El equipo docente para ayudarlos a probar si su implementación es correcta, se creó un conjunto de pruebas. Las pruebas fueron implementadas utilizando Postman. Para ejecutar las pruebas deben realizar los siguientes pasos:

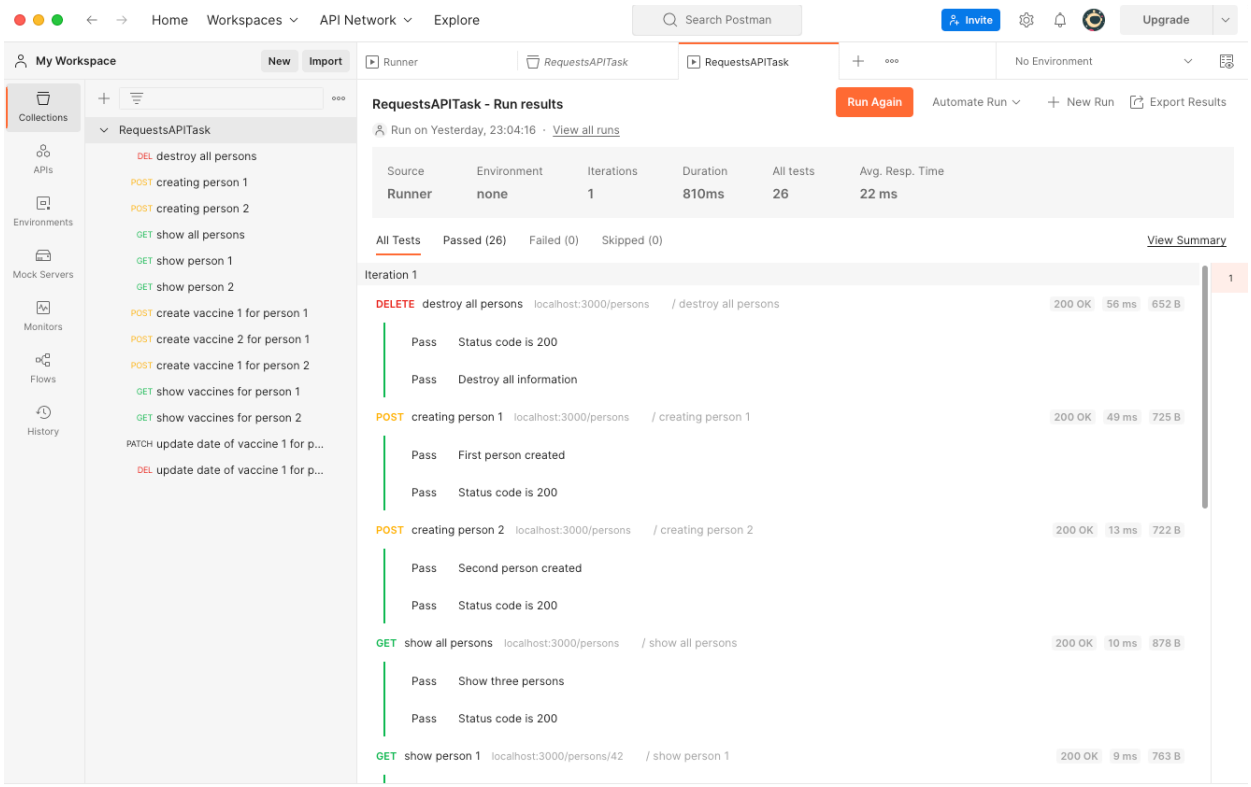
1. Iniciar el servidor en el directorio del proyecto.

2. Importar la colección de requests llamada `RequestsAPITask` en Postman. Usted puede encontrar la lista de request en Postman en la sección de Archivos en canvas.

3. En el menú de `RequestsAPITask` seleccione la opción 'Run collection' y después presione el botón naranja 'Run RequestsAPITask' como se observa en la figura 3.




4. El resultado sera un informe sobre las requests que atendió su API. En caso de que `task` API este bien implementada, saldrá un reporte similar al que se observa en la figura 4, en el cual los 26 tests pasaron. En caso de que algún test falle, puede acceder al request en particular y explorar la información.




# Entrega

La entrega de esta tarea deberá realizarse en un repositorio de GitHub que se les facilitara, es importante que su código se encuentre en la rama *main*, de lo contrario la tarea no se revisara. Es responsabilidad del alumno preocuparse de aceptar la invitación a su repositorio respectivo.

# Política de integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería en el SIDING. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1.1 en el curso y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente. Lo anterior se

entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.