

**Design and Development of a REST API in Python. Part 2**

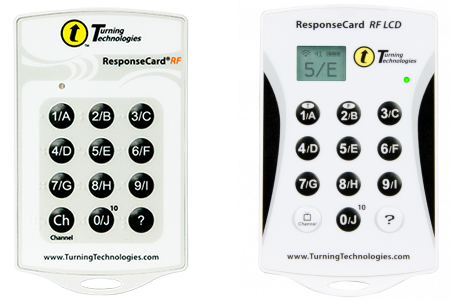
[Iván González](https://mamiot.com/author/ivan/)

Continuamos con el "Diseño y Desarrollo de una API REST en Python". En la [Parte 1](https://mamiot.com/diseno-y-desarrollo-de-una-api-rest-en-python/) desarrollamos, como ejemplo instructivo, una Web API muy sencilla para la **gestión de usuarios**, algo bastante necesario en todo tipo de aplicaciones. En esta segunda parte, vamos a trabajar en el diseño de una API REST más compleja, con más recursos que se relacionan entre sí y están caracterizados por un mayor número de atributos.

# Contexto: Sistemas de Respuesta de Audiencia

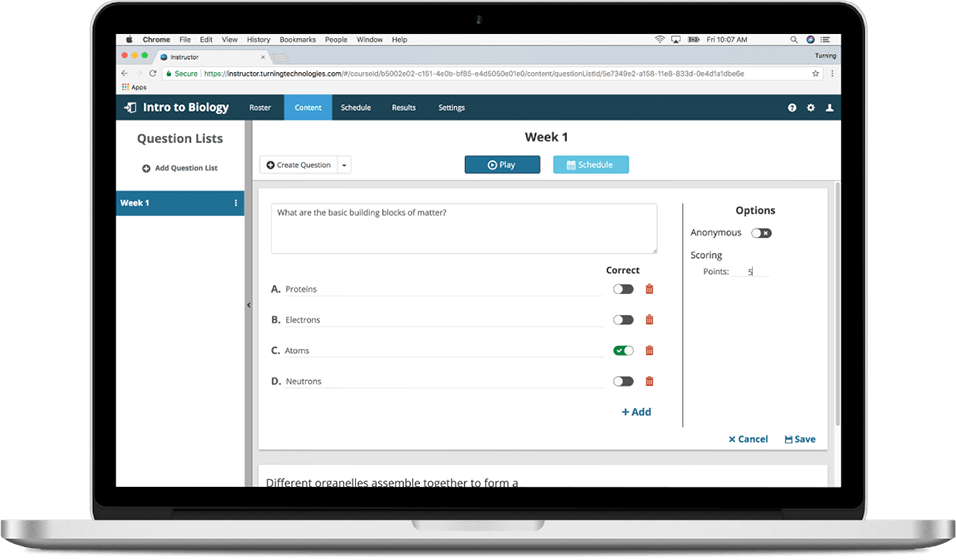
Vamos a diseñar la nueva Web API focalizando en un contexto particular relacionado con uno de los [proyectos](https://mamiot.com/projects/) del laboratorio MAmIoT. Especificamente, vamos a crear recursos y peticiones para un tipo de sistemas denominados: **Sistemas de Respuesta de Audiencia (ARS)**. Los ARS, comúnmente denominados **clickers**, son unos pequeños dispositivos "de mano" que permiten realizar encuestas interactivas en el aula. A cada alumno, se le proporciona uno de estos dispositivos que lo identifica temporalmente durante el desarrollo de la encuesta. A grandes rasgos, un clicker consta de varios botones para emitir las respuestas a cada una de las preguntas que se realicen en el transcurso de una encuesta (y en algunos casos, también incorpora una pequeña pantalla en la que ver reflejada la respuesta emitida).

Cada clicker integra un módulo emisor (habitualmente de radiofrecuencia) que sirve para comunicar la respuesta junto con el identificador unívoco del dispositivo clicker <id\_clicker,Respuesta> a un único dispositivo receptor, en posesión del profesor. Éste último está conectado a un computador con una aplicación específica que almacena la respuesta emitida por cada uno de los clickers. En la figura siguiente pueden verse dos clickers de la empresa ©Turning Technologies.



*Clickers de la empresa ©Turning Technologies sin/con pantalla.*

A continuación, se muestra también una captura del software (plataforma Web) de ©Turning Technologies para la creación, gestión y planificación de encuestas interactivas que pueden iniciarse/lanzarse en cualquier momento y responderse empleando los clickers.



*Aplicación Web de la empresa ©Turning Technologies para la gestión de encuestas.*

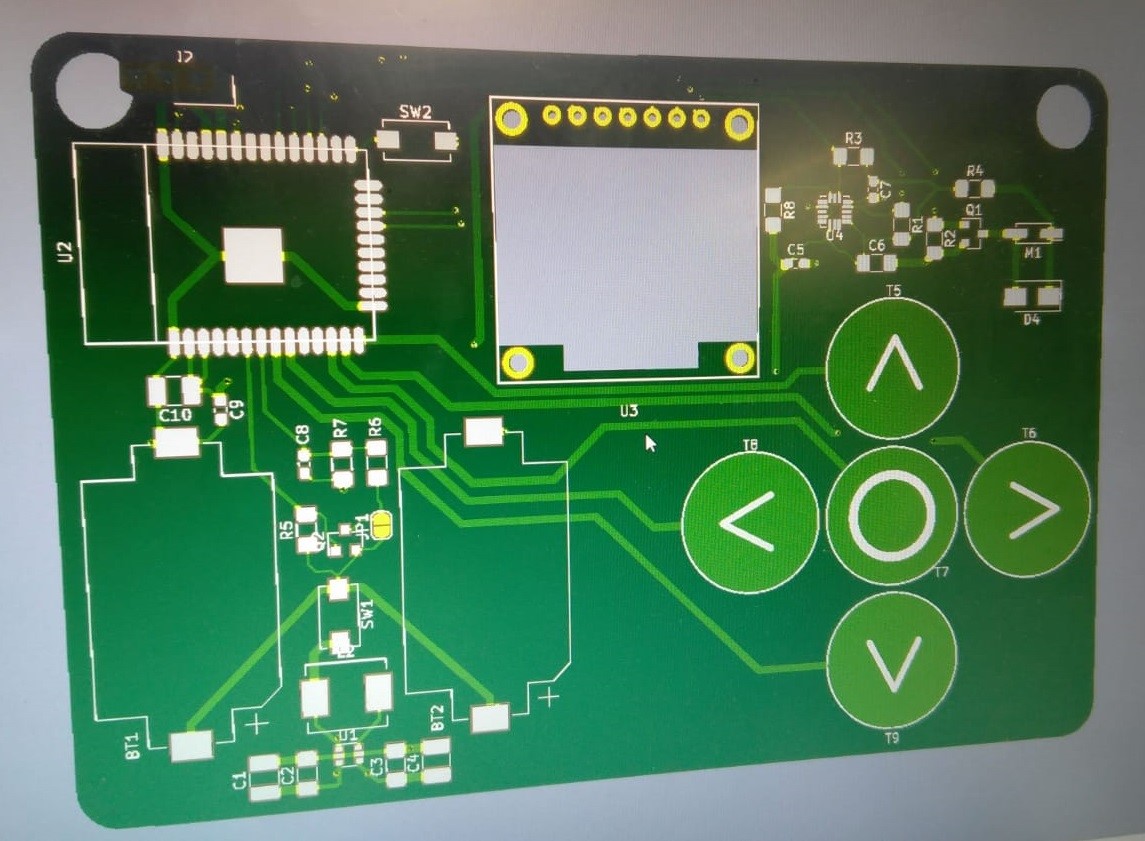
# Prototipo de clicker con funcionalidades añadidas

Partiendo del contexto de los sistemas ARS, explicado anteriormente, en [MAmIoT](http://www.mamiot.com/) estamos desarrollando nuestro propio prototipo de clicker con algunas funcionalidades más que no poseen el resto de dispositivos de este tipo. Por ejemplo, con la adición de los artefactos software necesarios, nuestro prototipo de clicker proporcionará **identidad unívoca y permanente tanto al propio estudiante, como al profesor**. La idea es que todos ellos porten su dispositivo en cada lección, con independencia del aula donde se lleve a cabo, dentro del campus universitario. En cualquiera de estas lecciones podrán participar en una encuesta (poll) interactiva con su clicker personal, cada vez que el profesor crea conveniente lanzar una de ellas como parte de sus dinámicas de enseñanza.

Cada clicker tiene embebido un SoC (System on Chip) Espressif [ESP32](https://www.espressif.com/en/products/hardware/esp32/overview) que integra un transceptor inalámbrico WiFi 802.11 b/g/n, así como una implementación completa de la pila TCP/IP, como parte de su firmware. Gracias al ESP32, podemos ofrecer diferentes modos de conexión inalámbrica: i) el modo "estación" (STA, Station), es decir, funcionar como un dispositivo que se conecta a un punto de acceso específico; ii) o el modo de "punto de acceso" (AP, Access Point) al que otras estaciones/dispositivos pueden conectarse. La alternancia periódica y sincronizada de estos dos modos de conexión y las opciones de escaneo de redes inalámbricas en la cercanía del SoC embebido en el clicker, proporcionan datos que, con el desarrollo del software y algoritmos específicos, **ofrecen información contextual interesante sobre lo que ocurre dentro del aula. Desde "pasar lista" de manera implícita y saber qué alumnos asistieron a cada lección (a partir de las MACs de sus clickers), hasta conocer si se sientan cerca del puesto del profesor; o si se pueden distinguir diferentes grupos de alumnos en un curso o asignatura específica.**

Además de una pequeña pantalla, que en el prototipo actual es una pantalla OLED monocromo de 128x64 píxeles (0.96 pulgadas) y un arreglo de botones capacitativos, nuestro prototipo integra un acelerómetro triaxial y un micromotor DC que sirve para emitir pequeñas vibraciones. **La combinación de estos sensores y actuadores abre la puerta a nuevas formas de interacción dentro del aula, incluyendo gestos naturales y diferentes notificaciones en las que intervenga la pantalla y el micromotor**. Además, en versiones venideras del prototipo también se incorporará un **lector/escritor RFID que permitirá interacciones por contacto entre clickers; y de clickers con elementos del aula y del campus** (pupitres, pizarras, tablones de anuncios, etc). También, pretendemos sustituir la pequeña pantalla OLED por una e-INK de 2.19 pulgadas, que puede mostrar permanentemente imágenes estáticas con un consumo marginal (al contrario que las OLED que deben apagarse para reducir su consumo).

La figura siguiente muestra una captura del diseño de la PCB de nuestro prototipo de clicker. Se ha optado por un diseño en forma de tarjeta de acreditación que puede colgarse con una cinta a modo de badge.



*Diseño de la PCB del prototipo de clicker.*

# Diseño de la Web API para un ARS

Dicho todo esto, toca pasar a la acción. Naturalmente en esta receta no vamos a poder abarcar todos los recursos necesarios que requiere un ARS para que sea completamente funcional, pero sí que vamos a cubrir los recursos más importantes y cómo se relacionan entre sí, esto último no se llegó a ver en la [Parte 1](https://mamiot.com/diseno-y-desarrollo-de-una-api-rest-en-python/) de la receta sobre Diseño y Desarrollo de una API REST en Python, donde tan sólo teníamos un recurso (usuario) aislado. Por tanto, lo que trataremos a continuación nos ayudará a comprender mejor como asumir la tarea de diseñar y desarrollar una Web API más compleja.

Para usar los clickers necesitamos estudiantes (students), por tanto, ese será nuestro primer recurso. Las URIs para el recurso student son mostradas en la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| /v1/students/  <student\_id>/ | GET | Devuelve los datos de un estudiante |
| /v1/students/ | GET | Devuelve el listado de todos los estudiantes (probablemente en una implementación real sólo un administrador tenga acceso a todos los estudiantes) |
| /v1/students/ | POST | Añade un nuevo estudiante al sistema (probablemente sea una tarea a realizar por un administrador) |
| /v1/students/ | PUT | Edita datos de un estudiante en particular |
| <student\_id>/ |  |  |
| /v1/students/  <student\_id>/ | DELETE | Elimina a un estudiante (probablemente sea una tarea a realizar por un administrador) |
| |  |  | | --- | --- | | /v1/students/ |  | | <student\_id>/subjects/ | | | GET | Devuelve todas las asignaturas en las que está matriculado un estudiante. Por lo tanto, podemos prever otro recurso para modelar asignaturas |
| |  |  | | --- | --- | | /v1/students/ |  | | <student\_id>/polls/ | | | GET | Devuelve todas las encuestas en las que el estudiante ha sido incluido como participante. Por lo tanto, podemo prever otro recurso para modelar encuestas/polls |
| /v1/students/  <student\_id>/subjects/  <subject\_id>/ | POST | Añade una asignatura a un estudiante (matricular - sólo un administrador) |
| /v1/students/  <student\_id>/subjects/  <subject\_id>/ | DELETE | Elimina una asignatura de un estudiante (quitar matrícula - sólo un administrador) |

Además de estudiantes, requerimos profesores (teachers). Por tanto, ese es nuestro siguiente recurso. Las URIs para el recurso teacher son mostradas en la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| /v1/teachers/  <teacher\_id>/ | GET | Devuelve los datos de un profesor específico |
| /v1/teachers/ | GET | Devuelve el listado de todos los profesores (probablemente en una implementación real sólo un usuario administrador tenga acceso a todos los profesores) |
| /v1/teachers/ | POST | Añade un nuevo profesor al sistema (probablemente sea una tarea a realizar por un administrador) |
| /v1/teachers/  <teacher\_id>/ | PUT | Edita datos de un profesor en particular |
| /v1/teachers/  <teacher\_id>/ | DELETE | Elimina a un profesor (probablemente sea una tarea a realizar por un administrador) |
| |  |  | | --- | --- | | /v1/teachers/ |  | | <teacher\_id>/subjects/ | | | GET | Devuelve todas las asignaturas en las que imparte clase un profesor |
| |  |  | | --- | --- | | /v1/teachers/ |  | | <teacher\_id>/polls/ | | | GET | Devuelve todas las encuestas creadas por un profesor |
| /v1/teachers/  <teacher\_id>/subjects/  <subject\_id>/ | POST | Añade una asignatura para ser impartida por un profesor (sólo un administrador) |
| /v1/teachers/ | DELETE | Elimina una asignatura de las que imparte un profesor (sólo un |
| |  |  | | --- | --- | | <teacher\_id>/subjects/ | | | <subject\_id>/ |  | |  | administrador) |

En este diseño en particular, tanto estudiantes como profesores pueden ser "vinculados/desvinculados" a una asignatura creada previamente. No se está siguiendo esta misma política en las encuestas/polls. Las URIs de estudiante y profesor pueden obtener (GET) directamente las encuestas que han creado (profesores) y en las que han sido incluidos como participantes (estudiantes), pero no hay endpoints para añadir/quitar (POST/DELETE) una encuesta.

El siguiente recurso que requerimos y que ya aparecido en alguno de los endpoints anteriores es el de subject. Las URIs de este recurso se muestra en la tabla siguiente.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| /v1/subjects/  <subject\_id>/ | GET | Devuelve los datos de una asignatura/subject específica |
| /v1/subjects/ | GET | Devuelve el listado de todas las asignaturas |
| /v1/subjects/ | POST | Crea un nueva asignatura (probablemente sea una tarea a realizar por un administrador) |
| /v1/subjects/  <subject\_id>/ | PUT | Edita una asignatura concreta |
| /v1/subjects/  <subject\_id>/ | DELETE | Elimina una asignatura (probablemente sea una tarea a realizar por un administrador) |
| |  |  | | --- | --- | | /v1/subjects/ |  | | <subject\_id>/students/ | | | GET | Devuelve los estudiantes matriculados en una asignatura |
| |  |  | | --- | --- | | /v1/subjects/ |  | | <subject\_id>/teachers/ | | | GET | Devuelve los profesores que imparten en una asignatura |

Otro recurso necesario es el relacionado con las encuestas (poll), cuyas URIs se exponen en la tabla siguiente.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| /v1/polls/<poll\_id>/ | GET | Devuelve los datos de una encuesta/poll específica |
| /v1/polls/ | GET | Devuelve el listado de todas las encuestas (sólo administrador) |
| /v1/polls/ | POST | Crea una encuesta (sólo administrador) |
| /v1/polls/<poll\_id>/ | DELETE | Elimina una encuesta (sólo administrador) |
| /v1/polls/<poll\_id>/ | PUT | Edita una encuesta concreta (sólo administrador) |
| /v1/polls/teachers/  <teacher\_id>/subjects/  <subject\_id>/ | POST | Crea un nueva encuesta (por un profesor y para una asignatura en particular) |
| /v1/polls/teachers/  <teacher\_id>/subjects/  <subject\_id>/ | PUT | Edita los datos de una nueva encuesta (por un profesor y para una asignatura en particular) |
| |  |  | | --- | --- | | /v1/polls/<poll\_id>/ | | | question/ |  | | POST | Añade una nueva pregunta a una encuesta (poll\_id) |
| |  |  | | --- | --- | | /v1/polls/<poll\_id>/ | | | questions/ |  | | GET | Devuelve las preguntas que componen una encuesta |
| |  |  |  | | --- | --- | --- | | /v1/polls/<poll\_id>/ | | | | questions/ |  | | | <question\_id>/ | |  | | GET | Retorna la información de una pregunta concreta con identificador question\_id de la encuesta poll\_id |
| |  |  |  | | --- | --- | --- | | /v1/polls/<poll\_id>/ | |  | | questions/ |  | | <question\_id>/publish/ | | | | GET | Este endpoint puede usarse para lanzar la pregunta identificada como question\_id, dentro de la encuesta poll\_id, a los clickers de los estudiantes participantes |

Cada encuesta esta constituida por una serie de preguntas (question), por lo tanto, parece una buena decisión añadir ese recurso a la API REST. La tabla siguiente muestra algunos de los endpoints que pueden definirse para éste.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| /v1/questions/  <question\_id>/ | GET | Devuelve la información de la pregunta question\_id |
| /v1/questions/  <question\_id>/ | DELETE | Elimina la pregunta question\_id |
| /v1/questions/  <question\_id>/ | PUT | Modifica la pregunta question\_id con nueva información |

Se requerirá también un recurso para manejar cada respuesta (answer), emitida por cada estudiante, para cada una de las preguntas de una encuesta. La tabla siguiente muestra algunos de los endpoints que pueden definirse.

|  |  |  |
| --- | --- | --- |
| **URI** | **Método**  **HTTP** | **Descripción/Acción** |
| |  |  | | --- | --- | | /v1/answers/polls/ | | | <poll\_id>/ |  | | GET | Devuelve todas las respuestas de los participantes en la encuesta/poll con identificador poll\_id |
| /v1/answers/polls/ <poll\_id>/ students/ <student\_id> | GET | Devuelve todas las respuestas del estudiante student\_id en la encuesta/poll con identificador poll\_id |
| /v1/answers/ | GET | Devuelve la respuesta answer\_id de la encuesta/poll con |
| <answer\_id>/polls/  <poll\_id>/ students/ <student\_id> |  | identificador poll\_id del estudiante student\_id |
| /v1/answers/ | POST | Esta URI puede ser utilizada por los clickers para emitir su respuesta durante una encuesta en curso, adjuntando a la petición un objeto JSON con la información del dispositivo (device) que comunica la respuesta, el identificador de la encuesta y el identificador de la pregunta que está siendo contestada |
| /v1/answers/  <answer\_id>/ | DELETE | Elimina la respuesta con identificador answer\_id. Sólo el usuario administrador |

Esta tablas exponen algunas de las URIs/endpoints que puede implementarse. **El diseño, además de estar simplificado, no es único** y puede variar significativamente e incluir otros endpoints o prescindir de algunos de los definidos en las tablas. Igualmente, otro diseño puede incluir un mayor número de recursos o prescindir de algunos de ellos y reflejarlos como atributos de recursos concretos.

# Ejercicio

Como podéis comprobar, hemos facilitado algunos recursos, junto con sus tablas de URIs como parte del diseño de esta API REST en el contexto particular de los ARS (Sistemas de Respuesta de Audiencia). Si bien, no hemos facilitado los atributos que modelan cada uno de los recursos. El ejercicio demandado consiste en **realizar una propuesta de atributos para cada recurso. También podéis indicarnos si añadiríais algún recurso adicional más o haríais algún cambio en la tabla de URIs/endpoints.**