DESARROLLO DE APLICACIONES WEB EN ENTORNO SERVIDOR

Proyecto Unidad 4 y 5.
Desarrollo de Backend en Codeigniter

Autor: Miguel Mira Flor

ÍNDICE

1. Objetivo

Desarrollar una aplicación web haciendo uso de Codeigniter, nosotros nos centraremos en la realización del Backend. Para ello, desarrollaremos el API Rest para que pueda ser accedido desde el Frontend y además, nutriremos nuestra base de datos obteniendo información desde Apis públicas y canales RSS.

2. Descripción del proyecto

La idea es desarrollar una aplicación de turismo para una localidad específica, donde realizaremos tanto la parte cliente (Frontend) como la parte servidor (Backend). En esta asignatura desarrollaremos el backend, y será utilizado posteriormente en la asignatura de desarrollo web en entorno cliente.

El proyecto podrá ser realizado de forma **individual** o en **pareja**:

- Cada pareja o persona individual, deberá elegir una población sobre la cual hacer la práctica.
- Anotad vuestro nombre y ciudad en el foro que hay abierto en Aules para ello.
- Recuerda no elegir una población pequeña porque habrá poca información.

El proyecto será realizado en **4** partes:

- **Individual**: las tres primeras partes serán obligatorias y la última parte será opcional (para subir nota).
- **Pareja**: las cuatro partes son obligatorias.

El contenido del proyecto consta de:

- **Restaurantes:** serán creados desde el backend (CRUD) *(no se obtienen de ninguna api o feed rss).*
- **Reseñas:** se podrán hacer reseñas de los restaurantes, para ello, éstas serán creadas desde nuestra API Rest (CRUD).
- **Gasolineras:** se obtendrán de una api pública.
- **Tiempo (meteorología):** se obtendrá de una api pública.
- **Noticias:** se obtendrán de un canal RSS.
- **Galería de vídeos:** se obtendrán de un canal RSS de Youtube.

2.1. Tecnología y elementos a emplear

El proyecto estará desarrollado enteramente en **Codeigniter**, por lo tanto, se deberán emplear todos los elementos que hemos visto en clase. Si habéis realizado el proyecto inicial de Codeigniter, tenéis todas las herramientas necesarias para desarrollar este proyecto.

Como el proyecto se realizará en varias partes, dependiendo de la parte a realizar se utilizarán unos elementos u otros. Por un lado, se emplearán:

- **Controladores**
- **Rutas**
- **Migraciones / Seeders**
- **Modelos / Entidades**
- **Comandos / Curl / Feed RSS**
- **Api Rest**

Por otro lado, si se realiza la cuarta parte (que veremos a continuación), a parte de los elementos anteriores, se deberán emplear también:

- **Vistas**
- **Ajax**
- **Sesiones**
- **Filtros**
- **Datatable**

2.2. Desarrollo de la Primera Parte (base de datos)

En esta parte nos centraremos en la parte de la base de datos y en los elementos que harán uso de ella:

- **Migraciones / Seeders**
- **Modelos**
- **Entidades**

Nota: guíate con lo que ya has hecho en la práctica inicial para aplicarlo a este proyecto.

1º Define el diagrama Entidad-Relación que contenga las siguientes tablas e información *(Eres libre de añadir la información extra que quieras, pero debes añadir a cada tabla como mínimo la información que te indico a continuación):*

- **restaurants**
  - **id (pk)**
  - name
  - description
  - address
  - latitude
  - longitude
  - reviewAverage *(indicará la media de la puntuación de las reviews)*
  - numReviews *(indicará el número de reviews que se han hecho)*

- **reviews**
  - **id (pk)**
  - description
  - punctuation
  - email
  - **restaurant_id (fk)**

- **gas_stations**
  - **id (pk)**
  - label
  - address
  - latitude
  - longitude
  - ideess *(identificador extra, en la parte correspondiente explico su utilidad)*

- **weather:**
  - **id (pk)**
  - main
  - description
  - icon

- **news**
    - **id (pk)**
    - title
    - description
    - pubDate
    - url
    - guid *(identificador extra, en la parte correspondiente explico su utilidad)*
- **videos**
    - **id (pk)**
    - title
    - description
    - pubDate
    - url
    - guid *(identificador extra, en la parte correspondiente explico su utilidad)*
- **roles**
    - **id (pk)**
    - name
- **users**
    - id **(pk)**
    - username
    - email
    - password
    - name
    - surname
    - **role_id (fk)**

*(Consúltame si tienes cualquier duda respecto al **tipo de dato** de algún atributo).*

2º Crea un **Migration** por cada tabla, acuérdate de que no debes crear o modificar la estructura de la base de datos directamente con phpmyadmin o similar, si no que lo hacemos con los migrations.

- **Importante**: cada tabla debe tener su **created_at, updated_at y deleted_at.**

3º Añade un mínimo de información a las tablas haciendo uso de los **Seeders**:

- **restaurants**: busca restaurantes de tu localidad asignada y añade alguno de ellos.
- **reviews**: puedes dejarla vacía, o añadir algún review *(no pierdas mucho tiempo con esta información)*
- **gas_stations**: se rellenará posteriormente con la llamada api, pero puedes crear alguna de prueba con los seeders *(no pierdas mucho tiempo con esta información).*
- **weather**: se rellenará posteriormente con la llamada api, pero puedes crear alguna de prueba con los seeders *(no pierdas mucho tiempo con esta información).*
- **news**: se rellenará posteriormente con el feed RSS, pero puedes crear alguna de prueba con los seeders *(no pierdas mucho tiempo con esta información).*
- **videos**: se rellenará posteriormente con el feed RSS, pero puedes crear alguno de prueba con los seeders *(no pierdas mucho tiempo con esta información).*
- **roles***:* crea un solo rol administrador.
- **users**: crea un solo usuario con rol administrador.

Nota: la idea es que tengáis algo de información para poder hacer pruebas en la segunda parte.

4º Crea un **Modelo** y una **Entidad** por cada tabla (acuérdate de activar el tema de las fechas, created_at, updated_at, deleted_at. Si te guías con la práctica inicial, no tendrás problemas).

2.3. Desarrollo de la Segunda Parte (Api Rest)

Esta es la **parte principal y más grande** de nuestra aplicación, ya que crearemos nuestra API Rest y por lo tanto, los servicios que utilizaréis desde el cliente (Frontend). Los elementos de Codeigniter que emplearemos en esta parte serán:

- **Controladores**
- **Rutas**
- **Modelos / Entidades**
- **Api Rest**

Nota: para poder probar los servicios utiliza **Postman** o un programa similar.

A continuación, se detallan todos los servicios que debéis crear:

(No te asustes si ves muchos servicios, el 80% son iguales únicamente cambia la tabla).

Obtener restaurantes (opcional)

Tipo de petición: **GET**
Parámetros: **id** (id del restaurante) (opcional)
Formato de respuesta: **JSON**

Funcionamiento:

- Si la ruta no recibe ningún parámetro, el servicio debe devolver un listado de todos los restaurantes (sin eliminar) de la base de datos.
- Si la ruta recibe el id del restaurante, devolverá únicamente el restaurante con ese identificador

Respuestas:

- 200: listado de restaurantes (si no se le ha pasado el ID) (si no hubieran restaurantes, se obtendrá un listado vacío).
- 200: restaurante (si se le ha pasado el ID)
- 404: si el id enviado no corresponde con algún restaurante de la base de datos.
- 500: para cualquier otro error.

Obtener gasolineras (opcional)

Tipo de petición: **GET**
Parámetros: **id** (id de la gasolinera) (opcional)
Formato de respuesta: **JSON**

Funcionamiento:

- Si la ruta no recibe ningún parámetro, el servicio debe devolver un listado de todas las gasolineras (sin eliminar) de la base de datos.
- Si la ruta recibe el id de la gasolinera, devolverá únicamente la gasolinera con ese identificador.

Respuestas:

- 200: listado de gasolineras (si no se le ha pasado el ID) (si no hubieran gasolineras, se obtendrá un listado vacío).
- 200: gasolinera (si se le ha pasado el ID)
- 404: si el id enviado no corresponde con alguna gasolinera de la base de datos.
- 500: para cualquier otro error.

Obtener tiempo actual (opcional)

Tipo de petición: **GET**

Parámetros: **ninguno**

Formato de respuesta: **JSON**

Funcionamiento:

- El servicio devolverá el último registro insertado en la base de datos respecto al tiempo (weather).

Respuestas:

- 200: detalle del tiempo.
- 404: si no hay ningún registro de tiempo en la base de datos.
- 500: para cualquier otro error.

Obtener noticias (opcional)

Tipo de petición: **GET**

Parámetros: **id** (id de la noticia) (opcional)

Formato de respuesta: **JSON**

Funcionamiento:

- Si la ruta no recibe ningún parámetro, el servicio debe devolver un listado de todas las noticias (sin eliminar) de la base de datos.
- Si la ruta recibe el id de la noticia, devolverá únicamente la noticia con ese identificador.

Respuestas:

- 200: listado de noticias (si no se le ha pasado el ID) (si no hubieran noticias, se obtendrá un listado vacío).
- 200: noticia (si se le ha pasado el ID)
- 404: si el id enviado no corresponde con alguna noticia de la base de datos.
- 500: para cualquier otro error.

Obtener vídeos (opcional)

Tipo de petición: **GET**

Parámetros: **id** (id del vídeo) (opcional)

Formato de respuesta: **JSON**

Funcionamiento:

- Si la ruta no recibe ningún parámetro, el servicio debe devolver un listado de todos los vídeos (sin eliminar) de la base de datos.
- Si la ruta recibe el id del vídeo, devolverá únicamente el vídeo con ese identificador.

Respuestas:

- 200: listado de vídeos (si no se le ha pasado el ID) (si no hubieran vídeos, se obtendrá un listado vacío).
- 200: video(si se le ha pasado el ID)
- 404: si el id enviado no corresponde con algún vídeo de la base de datos.
- 500: para cualquier otro error.

A continuación, se detallan los servicios respecto al CRUD de **reviews**:

Obtener reseñas

Tipo de petición: **GET**

Parámetros: **restaurant_id (obligatorio)**

Formato de respuesta: **JSON**

Funcionamiento:

- El servicio devolverá todas las reseñas del restaurante especificado.

Respuestas:

- 200: listado de reseñas (si no hubieran reseñas, se obtendrá un listado vacío).
- 400: si no se ha enviado el id del restaurante
- 404: si el id enviado no corresponde con algún restaurante de la base de datos.
- 500: para cualquier otro error.

Obtener reseñas por email y restaurante

Tipo de petición: **GET**

Parámetros:

- **restaurant_id (obligatorio)**
- **email (obligatorio)**

Formato de respuesta: **JSON**

Funcionamiento:

- El servicio devolverá el listado de reseñas de un usuario (email) sobre un restaurante.

Respuestas:

- 200: listado de reseñas (si no hubieran reseñas, se obtendrá un listado vacío).
- 400: si no se ha enviado el id del restaurante o el email.
- 404: si el id enviado no corresponde con algún restaurante de la base de datos.
- 500: para cualquier otro error.

Obtener reseña

Tipo de petición: **GET**

Parámetros: **review_id (obligatorio)**

Formato de respuesta: **JSON**

Funcionamiento:

- El servicio devuelve la reseña que se le haya pasado por parámetro.

Respuestas:

- 200: reseña.
- 400: si no se ha enviado el id de la reseña.
- 404: si no existe la reseña enviada (review_id).
- 500: para cualquier otro error.

Crear/Actualizar reseña

Tipo de petición: **POST**

Parámetros:

- **review_id (opcional)**
- **restaurant_id (obligatorio)**
- **email (obligatorio)**
- **description (obligatorio)**
- **punctuation (obligatorio)**

Formato de respuesta: **JSON**

Funcionamiento:

- Si el servicio recibe el review_id, significa que queremos ACTUALIZAR, por lo tanto, actualizamos con los datos enviados.
- Si el servicio NO recibe el review_id, significa que queremos CREAR, por lo tanto, creamos la nueva reseña con los datos enviados.
- Acuérdate actualizar: **reviewAverage** y **numReviews** del restaurante involucrado.

Respuestas:

- 200: mensaje editado correctamente.
- 400: si no se ha enviado algún dato obligatorio.
- 404: si no existe la reseña enviada (review_id) o el restaurante enviado.
- 500: para cualquier otro error.

Eliminar reseña

Tipo de petición: **DELETE**

Parámetros: **review_id (obligatorio)**

Formato de respuesta: **JSON**

Funcionamiento:

- El servicio eliminará la reseña especificada.

Respuestas:

- 200: mensaje eliminado correctamente.
- 400: si no se ha enviado el id de la reseña.
- 404: si el id enviado no corresponde con alguna reseña de la base de datos.
- 500: para cualquier otro error.

2.4. Desarrollo de la Tercera Parte (Comandos, Apis y RSS)

En esta parte y última (si lo estás realizando de forma individual), rellenaremos las tablas:

- **gas_stations**
- **weather**
- **news**
- **videos**

Lo haremos de la misma forma que lo hemos hecho en la práctica inicial, a través de un comando y haciendo una petición bien a un api o a un feed RSS para obtener los datos a insertar.

Por lo tanto, de los **cuatro** comandos a realizar, deberás elegir mínimo 2, **los otros 2 serán opcionales:**

- **Deberás elegir 2 comandos de los siguientes cuatro:**

Comando gasolineras

1º Deberás realizar una petición de tipo CURL (como has hecho con los pokemon) a la siguiente URL:

https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/

2º Obtendrás un listado de TODAS las gasolineras de España (JSON).

3º Recorre ese listado y guarda en la base de datos únicamente las gasolineras que correspondan a tu localidad:

- Si ya existe la gasolinera en la base de datos, actualízala.
- Si no existe la gasolinera, créala.

Nota: fíjate que cada gasolinera lleva un atributo llamado IDEESS, ese atributo también lo estamos guardando, es el que nos servirá para saber si la gasolinera existe o no en nuestra base de datos, y por lo tanto, saber si tengo que crearla o actualizarla.

Comando tiempo

Regístrate en esta página y crea un api key:
https://openweathermap.org/

1º Deberás realizar una petición de tipo CURL (como has hecho con los pokemon) a la siguiente URL:

http://api.openweathermap.org/data/2.5/weather?zip=**{codigo_postal}**,es&appid=**{API key}**

2º Obtendrás un JSON con la información sobre el tiempo actual.

3º Guarda un nuevo registro con esa información.

Comando noticias

1º Deberás realizar una petición de tipo feed RSS a la URL que se te asigne según tu localidad*

2º Obtendrás un listado de las últimas noticias (XML).

3º Recorre ese listado y guarda en la base de datos las noticias obtenidas:

- Si ya existe la noticia en la base de datos, actualízala.
- Si no existe la noticia, créala.

Nota: fíjate que cada noticia lleva un atributo llamado guid, ese atributo también lo estamos guardando, es el que nos servirá para saber si la noticia existe o no en nuestra base de datos, y por lo tanto, saber si tengo que crearla o actualizarla.

***Nota**: como habréis enviado al foro el nombre de vuestra localidad, yo buscaré una URL relacionada con noticias de vuestra localidad y os la pasaré.

Comando vídeos

1º Deberás realizar una petición de tipo feed RSS a la URL que se te asigne según tu localidad*.

2º Obtendrás un listado de los últimos vídeos (XML).

3º Recorre ese listado y guarda en la base de datos los vídeos obtenidos.

- Si ya existe el vídeo en la base de datos, actualízalo.
- Si no existe el vídeo, créalo.

Nota: fíjate que cada vídeo lleva un atributo llamado guid, ese atributo también lo estamos guardando, es el que nos servirá para saber si el vídeo existe o no en nuestra base de datos, y por lo tanto, saber si tengo que crearlo o actualizarlo.

***Nota**: como habréis enviado al foro el nombre de vuestra localidad, yo buscaré una URL con vídeos relacionados con vuestra localidad y os la pasaré.

2.5. Desarrollo de la Cuarta Parte (Gestor de contenidos)

Hasta ahora si te das cuenta, no hemos creado ninguna vista, es decir, hemos hecho todo el desarrollo y no tenemos parte visual.

Esta parte consistirá en realizar el panel admin (gestor de contenidos) que utilizaría nuestro cliente para gestionar todos los datos del backend (al igual que hemos hecho en la práctica inicial).

IMPORTANTE: si vas a hacer el proyecto en pareja, esta parte es **obligatoria**, si no, servirá para subir nota.

Los nuevos elementos que utilizaremos en esta parte, además de los ya utilizados son:

- **Vistas**
- **Ajax**
- **Sesiones**
- **Filtros (opcional)**
- **Datatable**

Básicamente tenéis que **hacer lo mismo que hemos hecho en la práctica inicial** con los festivales, a continuación detallo las funcionalidades:

- El contenido será **TODO privado**, es decir, ya no tenemos parte pública, todo será parte de administración. *(en la práctica inicial, recordad que teníamos una home public con el listado de festivales, aquí no queremos esa parte)*.
- Al ser el contenido privado, se deberá realizar un **login**.
- Al iniciar sesión, accederemos a la pantalla de inicio de nuestro panel admin y además:
    - El login no será accesible si ya hemos iniciado sesión.
    - Las pantallas privadas no deberán ser accesibles si no hemos iniciado sesión y si el rol no es administrador.
- Tendremos un menú de administración con tres puntos:
    - Inicio
    - Restaurantes
    - Cerrar sesión

(Es exactamente lo mismo que ya habéis hecho. Lo único nuevo es el "cerrar sesión", pero es muy sencillo).

A continuación, tenéis una explicación de cada punto de menú:

Inicio

Crear una página inicial con algún mensaje que nos de la bienvenida. Esta será la página a la que entraremos después del login.

Restaurantes

En este apartado realizaremos el CRUD de restaurantes, al igual que hemos hecho con los festivales en la práctica inicial.

En esta página cargaremos un **datatable** con todos los restaurantes, y las acciones que podremos realizar serán:

- **Paginar en la tabla**, para ir viendo todos los restaurantes página por página.
- **Crear un nuevo restaurante**, navegará a un formulario para añadir la información del nuevo restaurante.

- **Editar un restaurante**, navegará a un formulario para editar la información del restaurante elegido.
- **Eliminar un restaurante**, se rellenará el campo **deleted_at** para hacer el eliminado lógico (como ya habéis hecho).

Nota: los atributos **reviewAverage** y **numReviews** son campos que se calculan automáticamente en el servicio rest *crear/actualizar review*, eso significa que no debemos poder editarlos desde aquí. Por lo tanto, estos campos deberían ser **no editables (readonly).**

Cerrar sesión

Al hacer click a este punto de menú, se deberá cerrar la sesión y volver a la página de login.

4. Entrega

La entrega se realizará por Aules en la **fecha marcada para ello**. Deberéis entregar un fichero .zip con todo el proyecto.

La corrección se realizará de la siguiente forma:

- **1º** Se realizará una revisión la **primera semana de Febrero** para ver cómo váis avanzando.
- **2º** El **8, 9, 10, 11 y 12 de Febrero** corregiré la práctica in situ, como ya he hecho otras veces.
    - Esta vez, me defenderéis la práctica, y seré más exigente con las explicaciones, os voy a pedir que me expliquéis cómo habéis hecho cada parte, código, etc.
    - Siento hacerlo de esta forma, pero tengo que asegurarme que habéis entendido todo lo que habéis hecho.

(Si hubiera alguna variación en las fechas os lo haré saber y volveré a subir una versión de este documento con las nuevas fechas).

6. Criterios de evaluación

- **Individual:**
  - Parte 1: **20% (2 puntos)**
  - Parte 2: **50% (5 puntos)**
  - Parte 3: **30% (3 puntos)**

- **Parejas:**
  - Parte 1: **15% (1,5 puntos)**
  - Parte 2: **45% (4,5 puntos)**
  - Parte 3: **20% (2 puntos)**
  - Parte 4: **20% (2 puntos)**

- **No os preocupéis si no conseguís hacerlo todo al 100%**, al final se os irá sumando una nota según los elementos y partes que tengáis, tened en cuenta que si lo hacéis todo implica tener un 10.
  - Pero, sí debéis tener un mínimo hecho de cada parte para que os pueda corregir el proyecto completo.
  - ***Por ejemplo:*** *no me vale que una pareja haga la parte 1, 2 y 3 completa y no hagan nada de la parte 4, deberían hacer un mínimo de la parte 4.*

- Haré una selección de las partes que quiero que defendáis y expliquéis, dependiendo de la defensa, la nota podrá bajar o anular completamente cada parte.

- Tenéis que sacar mínimo un 4 en este proyecto para que os pueda hacer media con la nota de la práctica inicial.
- **IMPORTANTE:**
  - Las partes marcadas en azul son opcionales, si se realizan será para subir nota.
  - En caso de obtener un 10, se tendrá en cuenta para el redondeo de la nota en la evaluación final.

5. Observaciones

A partir de ahora, vamos a trabajar en el proyecto todos los días de clase. Yo os recomiendo que os centréis en cumplir los objetivos del proyecto, y si váis bien de tiempo, poco a poco lo podéis ir mejorando.

No quiero que GIT os retrase, por ello no voy a tener en cuenta que no hagáis uso de él, pero mi consejo es que lo utilicéis.

Yo voy a estar día a día con vosotros, por lo tanto a modo de jefe de proyecto os iré guiando durante todo el proceso de desarrollo para asegurarme de vuestros avances y del trabajo individual o en parejas que vayáis realizando.

Mucho cuidado con las copias, es algo que no voy a tolerar y seré firme con ello.

6. Ampliaciones

De momento, como ampliación propongo hacer la parte visual (parte cuatro), si lo haces de forma individual.

Llegado el momento, estoy abierto a escuchar cualquier mejora que queráis hacer si tenéis tiempo.

Lógicamente, hacer una ampliación os permitirá subir nota, e incluso si superáis el 10, lo tendré en cuenta para la evaluación final.