Práctica de Python, Django y REST

Master Programación Apps

[www.u-tad.com](http://www.u-tad.com)

Autor: Jorge Marcos Fernández

He creado un proyecto con dos Aplicaciones:

* Pictures: almacenamiento de datos de imágenes.
* Images: End point para subida de imágenes.

**1. Api REST: Aplicación *Pictures*:**

Pictures/models.py: Tabla “Picture” que almacena todos los campos solicitados

Tabla puente “UserSignUp” para el registro de usuarios

Pictures/api.py: Vistas Api REST

Pictures/urls.py : Api REST urls

Pictures/serializers.py: serializers

Pictures/permissions.py: permisos

**Tabla de End Points**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **End Point** | **URL** | **VIEW** | **SERIALIZER** | **PERMISSIONS** |
| Listado de imágenes Públicas | r'^api/pictures/$' | PictureList | PicturePublicSerializer | CanListOrCreate |
| Listado de imágenes de usuario | r'^api/pictures/$' | PictureList | PictureUserSerializer | CanListOrCreate (1) |
| Creación de una imágen | r'^api/pictures/$' | PictureList | PictureDetailsSerializer | CanListOrCreate (1) |
| Detalle de una imágen | r'^api/pictures/(?P<pk>[0-9]+)/$' | PictureDetail | PictureDetailsSerializer | CanRetrieveUpdateOrDelete (2) |
| Actualización de una imágen | r'^api/pictures/(?P<pk>[0-9]+)/$' | PictureDetail | PictureDetailsSerializer | CanRetrieveUpdateOrDelete (1) |
| Borrado de una imágen | r'^api/pictures/(?P<pk>[0-9]+)/$' | PictureDetail | PictureDetailsSerializer | CanRetrieveUpdateOrDelete (1) |
|  |  |  |  |  |
| Listado de usuarios | r'^api/users/$' | UserList | UserSerializer |  |
| Detalle de un usuario | r'^api/users/(?P<pk>[0-9]+)/$' | UserDetail | UserSerializer |  |
|  |  |  |  |  |
| Login | r'^api/login/' | DRF Login |  |  |
| Signup | r'^api/signup/' | SignUp | UserSignUpSerializer |  |

1. – Necesita autenticación
2. – Necesita autenticación si la imágen no es pública.

**Tabla de Requisitos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Punto** | **Requisito** |  | **Observaciones** |
| Listado de imágenes Públicas | Debe listar las imágenes clasificadas como públicas por los usuarios | **√** |  |
|  | Se mostrará el título y la URL de cada imagen | **√** | + propietario |
|  | No será necesaria autenticación | **√** |  |
| Listado de imágenes de usuario | Debe listar las imágenes del usuario que está autenticado | **√** |  |
|  | Se mostrará título, URL y si es pública o no | **√** |  |
|  | Será necesaria autenticación | **√** |  |
| Creación de una imágen | Permite crear una imagen enviando los datos: URL, titulo, descripción, datos de geolocalización y flag de pública/privado | **√** |  |
|  | Será necesaria autenticación | **√** |  |
| Detalle de una imágen | Muestra todos los campos de la imagen | **√** |  |
|  | La autenticación será opcional | **√** |  |
|  | Sólo el propietario de una imagen podrá ver el detalle de la imagen, salvo que ésta sea pública. | **√** |  |
| Actualización de una imágen | Permite actualizar una imagen enviando los datos: URL, titulo, descripción, datos de geolocalización y flag de pública/privado | **√** |  |
|  | Será necesaria autenticación | **√** |  |
|  | Sólo el propietario de la imagen o un usuario super administrador podrá actualizar los datos | **√** |  |
| Borrado de una imágen | Permite borrar una imagen | **√** |  |
|  | Será necesaria autenticación | **√** |  |
|  | Sólo el propietario de la imagen o un usuario super administrador podrá eliminar la imagen | **√** |  |

**2. Sitio web: Aplicación *Pictures*:**

Pictures/models.py: Tabla “Picture” que almacena todos los campos solicitados

templates/: Páginas HTML para el Api Web (basadas en estilo de Django Admin)

Pictures/admin.py: Django Admin

JMFDjangoUTad/urls.py: Api Web urls

Pictures/Views.py: Vistas Api Web

**Tabla de End Points**

|  |  |  |
| --- | --- | --- |
| **URL** | **View** | **Acción** |
| r'^pictures/' | PictureList (1) | Muestra la lista de imágenes públicas |
|  |  | Muestra botón de login arriba a la derecha |
| r'^pictures/' | PictureList (2) | Muestra la lista de imágenes públicas excepto las del usuario autenticado |
|  |  | Muestra la lista de imágenes propietarias del usuario autenticado |
|  |  | Arriba a la derecha muestra las opciones de cambiar contraseña y de logout |
| r'^pictures/(?P<urlUserName>.+) | UserPublicPictures | Muestra la lista de imágenes públicas del usuario del parámetro |
| r'^login/$' | DRF Login | Muestra la pantalla de login |
| r'^logout/$' | Logout | Realiza logout y muestra la pantalla de logout |
| r'^admin/' | DRF admin | Administrador integrado de Django para al Aplicación *Pictures* |

1. – Usuario autenticado
2. – Usuario no autenticado

**3. Api REST: Aplicación *Images* (Endpoint para la subida de imágenes)**

Images/Models.py: Tabla “Images” que almacena todos los campos solicitados

Images/admin.py: Django Admin

Images/api.py: Vistas Api REST

Images/Views: Vistas Api Web

Images/urls.py : Api REST urls

JMFDjangoUTad/urls.py: Url para la visualización de la imagen por HTTP (1)

Images/serializers.py: serializers

**Tabla de End Points**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **End Point** | **URL** | **VIEW** | **SERIALIZER** | **Observaciones** |
| Subida de imágenes | r'^api/images/$' | ImageCreate | ImageSerializer | FileUploadParser |
| Visualización de imágen | r'^images/(?P<imageName>.+) | ShowImage |  |  |

**Tabla de Requisitos**

|  |  |  |
| --- | --- | --- |
| **Requisito** |  | **Observaciones** |
| Los archivos de imagen deberán almacenarse en una carpeta del servidor | **√** | Carpeta /media/ |
| Será necesaria autenticación para poder utilizar el endpoint | **√** |  |
| Cada imagen, deberá quedar relacionada con el usuario que la crea | **√** | Campo user de la tabla “Images” |
| Se deberá registrar la hora de creación de cada imagen | **√** | Campo creationDate de la table “Images” |
| Cuando se cree la imagen, se deberá devolver la URL como respuesta a la petición de creación | **√** | Campo imageUrl de la tabla “Images” |
| Las imágenes subidas, deberán estar accesibles a través de HTTP para poder verlas con el navegador web | **√** | Endpoint r'^images/(?P<imageName>.+) con la Vista ShowImage. Ver (1) |
| Para poder implementar este servicio, será necesario instalar el paquete PIL o Pillow en el entorno virtual | **√** | Instalada la librería de imágenes de Python Pillow en el entorno virtual |

**4. Endpoint para registro de usuarios**

La aplicación Pictures contiene lo siguiente para realizar el registro de usuarios:

* La tabla puente UserSignUp en models.py con los campos requeridos para mapear a la table User de DRF
* El endpoint api/SignUp enlazado con
* La vista SignUp

Cumple con todas las restricciones del requisito:

* Todos los campos son obligatorios, ninguno puede ser vacío.
* Comprueba que no existe ningún usuario con el mismo nombre de usuario o e-mail.
* Se comprueba que el formato del email es correcto. (tipo de campo EmailField).
* Se comprueba que la contraseña y la confirmación son iguales.