04.

Caso de Negocio II

Job Readiness

Biblioteca App

Caso de negocio N° 2

Situación inicial 📋

Como parte de un equipo de desarrolladores, recibimos el pedido del departamento de ***Producto*** para desarrollar ***una aplicación para dar funcionalidades a una biblioteca,*** la cual permitirá a sus usuarios ***consultar el catálogo de libros, y un listado de los préstamos de libros realizados a sus socios.*** Nuestro líder técnico ya cuenta con los requerimientos desagregados en un backlog de tareas listo para que comencemos la etapa de desarrollo.

Nuestro objetivo 🎯

El objetivo consiste en **desarrollar una aplicación web** que permita registrar los libros que la biblioteca ofrece, registrar empleados y socios, realizar el préstamo de algún libro a un socio y visualizar diferentes tipos de listados. Por último, se pide la posibilidad de acceder a la información de los servicios a través de un *endpoint* en donde se podrán consultar todos los libros disponibles y poder filtrarlos por el *id* en donde se visualizará el detalle completo del mismo.

Metodología de trabajo🔧

A lo largo de las próximas **5 semanas** se deberá trabajar en la implementación de ***una aplicación base para una biblioteca,*** aplicando un marco de trabajo ágil y teniendo espacios de reunión con el líder técnico, para realizar consultas, hacer seguimiento de las tareas realizadas y asegurar la calidad del código producido.

Para ello, vamos a utilizar **Python/Django** como entorno de desarrollo, una base de datos **SQL**, y las librerías **Jinja** y **Bootstrap.**

Requerimientos 🤝

La ***aplicación de la biblioteca*** deberá cumplir con una serie de características y requerimientos técnicos para garantizar la calidad y funcionalidad de la misma.

- **Sugerencia:** se sugiere crear un proyecto en Django llamado *WebApp*, y luego agregar una app llamada *biblioteca*.

Requerimiento general

El requerimiento general que será transversal a todo el desarrollo del será la implementación de todas las funcionalidades básicas para poder ***registrar los libros, socios y empleados, poder listarlos y almacenar toda esa información en la base de datos***, las cuales son:

- ✓ CRUD (crear, leer, actualizar y eliminar) de autores, más el listado correspondiente. En el listado se deben visualizar solo los autores activos, y se debe ofrecer la posibilidad de restaurar registros no activos.
- ✓ CRUD (crear, leer, actualizar y eliminar) de libros, más el listado correspondiente. En el listado se deben visualizar solo los libros activos, y se debe ofrecer la posibilidad de restaurar registros no activos. Al crear/actualizar un registro, solo se debe permitir seleccionar autores activos (relación libro-autor).
- ✓ CRUD (crear, leer, actualizar y eliminar) de socios, más el listado correspondiente. En el listado se deben visualizar solo los socios activos, y se debe ofrecer la posibilidad de restaurar registros no activos.
- ✓ CRUD (crear, leer, actualizar y eliminar) de empleados, más el listado correspondiente. En el listado se deben visualizar solo los empleados activos, y se debe ofrecer la posibilidad de restaurar registros no activos.
- ✓ CRUD (crear, leer, actualizar y eliminar) de préstamos de libros, más el listado correspondiente. Al crear/actualizar un registro, solo se debe permitir seleccionar libros/socios/empleados activos.
- ✓ API para visualizar todos los libros que la empresa ofrece: https://localhost:8000/api/libros
- ✓ API para visualizar el detalle de un libro: https://localhost:8000/api/libros/<libro_id>
- ✓ Se deben poder visualizar todos los modelos desde el Admin de Django.
- ✓ En el Admin permitir realizar búsquedas por nombre de Libro
- ✓ En el Admin permitir realizar búsquedas por nombre y apellido para Socio, Autor y Empleado.
- ✓ Documentar el proyecto (archivo README) indicando todo lo que hay que hacer para ponerlo en marcha, además de todas las funcionalidades disponibles.

- **Sugerencia:** para la API es recomendable crear una app llamada api, y allí colocar las vistas y configurar las urls.

El sistema ofrece la posibilidad de realizar préstamos de libros a los socios registrados. A continuación algunos ejemplos de libros:

- El principito - Antoine Saint-Exupery
- El señor de los anillos - JRR Tolkien
- Don Quijote de la Manch - Miguel de Cervantes Saavedra
- Cien años de soledad - Gabriel García Marquez

Cuando se realiza un préstamo de libro, se deben registrar todos los datos necesarios, como cliente a quien se le presta, el empleado que realiza el préstamos, la fecha, etc.

⚠️**Importante**: la función de eliminar para *Socio, Autor, Libro* y *Empleado* es solo una baja lógica, es decir que que al hacer *click* en en el link de *"eliminar"*, lo que se hará es un update en el campo activo dejándolo en ***False***. Para el modelo de Préstamos de Libros si se puede eliminar de la base de datos.

Requerimientos específicos

El **modelo de Autor** debe contar con los siguientes campos:

- nombre (texto)
- apellido (texto)
- nacionalidad (texto)
- activo (boolean, por default True)

El **modelo de Libro** debe contar con los siguientes campos:

- título (texto)
- descripción (texto largo)
- ISBN (integer) [Standard número de 13 cifras que identifica a cada libro en el mundo]
- autor (relación con el modelo de Autor)
- activo (boolean, por default True)

El **modelo de Socio** debe contar con los siguientes campos:

- nombre (texto)
- apellido (texto)
- fecha nacimiento (date)
- activo (boolean, por default True)

El **modelo de Empleado** debe contar con los siguientes campos:

- nombre (texto)

- apellido (texto)
- numero legajo (integer)
- activo (boolean, por default True)

Cuando se realiza el **préstamo de algún libro**, se debe registrar:

- la fecha en que se realiza el préstamo
- la fecha en que debe ser devuelto (48 hs luego del préstamo)
- el socio a quien se le presta
- el libro solicitado
- y la persona que otorga el préstamo (que es un empleado)

Como vemos, para esto necesitamos un **modelo** que debe estar relacionado al **Libro**, al **Socio** y al **Empleado**.

➕ Extras:

- ✓ Agregar la posibilidad de que se puedan consumir los datos de los socios a través de una API. Por ejemplo https://localhost:8000/api/socios
- ✓ Agregar la posibilidad de que se puedan consumir los datos de los empleados a través de una API. Por ejemplo https://localhost:8000/api/empleados
- ✓ Agregar la posibilidad de que se puedan consumir los datos de los autores a través de una API. Por ejemplo https://localhost:8000/api/autores
- ✓ Agregar una página (vista) inicial con links a todas las funcionalidades del proyecto. Esto vendría a ser el Home Page, y desde allí se podría acceder a todos los módulos del sistema.

Requerimientos técnicos:

- Framework Django para el desarrollo de la aplicación
- Bases de datos SQLite para almacenar datos
- Utilización del ORM de Django
- Utilizar Jinja2 como sistema de templates
- Uso de Bootstrap para aplicar estilo a los templates.
- Utilizar GitHub como repositorio del código del proyecto
- Documentación del proyecto

¿Qué vamos a validar?🔍

Para asegurar el correcto cumplimiento de las tareas, el líder se encargará de revisar diferentes aristas del proyecto. Estos son algunos puntos que vamos a validar:

- Calidad y funcionalidad del proyecto
- Correcta configuración de Django
- Implementación correcta de las vistas y plantillas
- Implementación correcta de los endpoints para obtener los datos en formato JSON
- Legibilidad del código y documentación.
- Experiencia del usuario.
- Cumplimiento de los requisitos

Referencias🦺

- Backlog de tareas

Entregables ✅

- Repositorio en GitHub
    - Código
    - Documentación pertinente