

Instituto de formación técnica superior N° 18

Backend - Proyecto integrador

Curso: 2° Año

Profesor: Bonini Juan Ignacio

Carga horaria: 9 hs cátedra/semana

Régimen: Cuatrimestral

Para obtener la aprobación del proyecto, es fundamental entregar el código dentro del plazo establecido.


Adicionalmente, será necesario llevar a cabo una presentación individual del proyecto, donde se exhiba tanto su funcionamiento como el código realizado por el alumno.

Si el trabajo se llevó a cabo en equipo, el estudiante deberá describir la funcionalidad que lo distingue de los demás de manera individual.

Cabe destacar que la presentación deberá ajustarse a un límite máximo de 20 minutos de duración.

El proyecto integrador a entregar será una aplicación web con temática a elección, orientada a la gestión y que cumpla con los siguientes requisitos:

1. El trabajo se debe realizar en equipos de hasta máximo 4 participantes.
2. Se entrega de forma individual y cada miembro del equipo debe realizar una funcionalidad adicional.
3. El proyecto debe subirse a un repositorio git en la nube y debe estar accesible para los demás integrantes del equipo y para el docente.
4. Deben realizar tantas aplicaciones de Django como sean necesarias. Como mínimo cinco aplicaciones, lo ideal son siete o más.
5. Deben existir al menos 10 recursos distintos, algunos de ellos con parámetros de consulta y otros con parámetros de ruta y deben estar bien aplicados. * Ver punto 13 para el CRUD
6. Se deberá aplicar Programación Orientada a Objetos lo más que se pueda.
7. Las vistas deberán ser basadas en clases
8. Se deben utilizar templates que cumplan con las siguientes características:
 - 8.1. Deben estar asociados a las vistas. * TemplateView
 - 8.2. Debe existir la relación de herencia entre los templates (utilizando el base.html).
 - 8.3. Deben aplicarse filtros en los templates.
 - 8.4. Los templates deberán utilizar archivos estáticos (js, css, etc). * Archivos físicos
9. Se deben utilizar Django Forms que cumplan con las siguientes características
 - 9.1. Tener validaciones tanto en el front-end como en el back-end (excluyente).
 - 9.2. Deberán estar asociados a los templates.
 - 9.3. Al menos un formulario basado en clases.
 - 9.4. Debe haber formularios asociado a un modelo y no asociado a modelos.
10. Todas las aplicaciones deben poseer sus modelos o incluso más de un modelo por aplicación cuando corresponda. Deben poseer una relación de uno a muchos, muchos a muchos, uno a uno, dependiendo del uso que se le quiera dar.
11. El proyecto debe funcionar utilizando un servidor de base de datos local dentro de los soportados. **Se puede utilizar SQLite**, pero será un plus instalar y configurar PostgreSQL. Además, debe poseer las migraciones necesarias para su correcto funcionamiento. (El makemigrations lo tienen que hacer ustedes)
12. Se debe poder acceder al Admin de Django y registrar todos los modelos y estos deberán cumplir con las siguientes características
 - 12.1. Se deberá poder buscar por más de un campo a elección.
 - 12.2. Se deberán poder filtrar
 - 12.3. Se deberán poder ordenar
13. El proyecto debe poseer páginas a las que solo se pueda acceder mediante autenticación y la misma debe ser validada tanto en el front-end como en el back-end.
14. Se deberá crear altas, bajas y modificaciones en al menos 3 aplicaciones.

- 
15. El proyecto debe incluir una API RESTful que exponga programáticamente todas las funcionalidades principales del sitio web desarrollado con Django.
 16. La API debe estar implementada utilizando Django Ninja.
 17. Se deben implementar al menos 10 endpoints distintos como mínimo, distribuidos entre distintas aplicaciones del proyecto. Estos endpoints deben representar recursos relevantes del sistema y permitir operaciones CRUD completas (crear, leer, actualizar, eliminar).
 18. La API debe incluir: Endpoints con parámetros de ruta (/recurso/<id>/) y parámetros de consulta (/recurso/?filtro=valor).
Protección mediante autenticación, utilizando mecanismos como Session, Token o JWT.
Todas las rutas sensibles deben estar correctamente protegidas.
 19. Schemas basados en Pydantic para definir la estructura de entrada y salida de los datos.
 20. Se debe documentar la API utilizando alguna de las siguientes herramientas:
 21. Documentación automática provista por Swagger UI (incluida en Django Ninja por defecto), o un archivo README.md que describa los endpoints, sus métodos, parámetros esperados, posibles respuestas y ejemplos de uso.
 22. Se valorará el uso de: Paginación, ordenamiento y filtros personalizados en los endpoints.
Relaciones entre modelos representadas adecuadamente en los schemas (anidamiento de esquemas cuando sea necesario).
 23. La API debe reflejar todas las funcionalidades principales del sitio web, de modo que el frontend pueda consumir la API para replicar completamente el comportamiento de las vistas tradicionales de Django.