Proyecto de Software 2024

Trabajo Integrador (TI) - Etapa 2

Aplicación Privada

1 Módulo de Administración de Contenido

Este módulo permitirá administrar el contenido que se visualizará en la aplicación pública, estas publicaciones se ofrecerían por medio de una API a la aplicación pública/portal. Estas publicaciones pueden servir como: artículos informativos, publicaciones, notificación de eventos, etc. Como mínimo deberá contemplar la siguiente configuración:

- Fecha publicación: fecha en la que se publica el contenido
- Fecha creación: fecha en la que se crea el contenido
- Fecha actualización: esta fecha se modifica con cada actualización del registro
- Título: título de la publicación, no más de XXX caracteres
- Copete: breve resumen y anticipación de la noticia que se encuentra debajo del título
- Contenido: texto enriquecido de la publicación
- Autor/a: usuario que cargó la publicación
- Estado: Borrador/Publicado/Archivado

Se deben poder realizar todas las operaciones **CRUD** de cada contenido a publicar en la aplicación pública. Los registros deberán verse paginados.

A continuación se definen los roles y las acciones que pueden realizar sobre este módulo:

- **Administradora, Editora:** index, show, update, create.
- **Administradora:** index, show, update, create, destroy.

2 Módulo de Reportes

Esta sección va a permitir visualizar información estadística que pueda ser de utilidad tanto para personal administrativo como profesionales de la Institución. Solo podrán visualizarse desde la aplicación privada.

En esta vista se deberán diseñar y desarrollar 3 gráficos y 3 reportes distintos. Queda a criterio de las y los desarrolladores/as con una validación previa del ayudante asignado que tipo de gráficos/reportes generar considerando los datos utilizados.

Algunos ejemplos de reportes pueden ser:

- *Ranking de los propuestas de trabajos más solicitadas (Hipoterapia – Monta Terapéutica – Deporte Ecuestre Adaptado – Actividades Recreativas - Equitación)*
- *Reporte con las personas que adeudan pagos*
- *Reporte histórico de cobros en un rango de fechas asociado a una persona.*

Ejemplos de gráficos:

- *Gráfico de tortas/barras con la cantidad de las J&A becados.*
- *Gráfico de barra con la cantidad de ingresos por año/mes*
- *Cantidad de Personas con Certificado Único de Discapacidad (Este dato aparece en la Ficha 01 y después como Archivo a cargar)*
- *Gráfico con Discapacidades (Dato de la Ficha 01)*
- *Gráfico con Tipo de Discapacidad (Dato de la Ficha 01. Mental, Motora, Sensorial, Visceral)*

3 Módulo de contacto

Este módulo permitirá al usuario Administrativo visualizar un listado con las consultas recibidas desde el formulario de contacto del Portal público.

A su vez a cada registro de consulta, se le podrá asignar un Estado (pensar los estados que crean conveniente para el módulo) y un comentario (campo de texto opcional, sólo para referencia interna). Este comentario puede utilizarse por ejemplo para dejar información del estado final de la consulta.

Los resultados se deben poder ordenar por fecha, de manera ascendente y descendente. Y deben poder filtrarse por estado.

A continuación se definen los roles y las acciones que pueden realizar sobre este módulo:

- **Administración:** index, show, update, create, destroy.

4 API

Se deberán implementar en la aplicación privada una serie de endpoints que permitan realizar distintas operaciones u obtener contenido desde la aplicación pública.

Pueden acceder a la especificación de la API publicada en nuestra web en el siguiente enlace.

Nota: La especificación es una guía para el/la estudiante que podrá modificar en caso que necesite agregar nuevos endpoints.

Aplicación Pública (Portal)

Se deberá desarrollar **otra aplicación** a la cual accederán el público general para obtener información de interés sobre la Institución, actividades, noticias y contacto.

El código de esta nueva aplicación deberá compartir espacio dentro del mismo repositorio de código de la aplicación Flask. La aplicación pública la deberán realizar utilizando el framework VueJs 3. **Cómo requisito para el correcto funcionamiento en el servidor se deberá ubicar el código correspondiente a esta aplicación dentro del directorio web del proyecto**.

1 Home

La vista principal de la aplicación deberá contar con secciones que muestren la información básica de la Institución, tareas que desarrolla, contacto y noticias.

2 Contacto

En la sección de contacto, se deberá desarrollar un **formulario** que permita a los usuarios enviar mensajes. Este formulario deberá incluir al menos los siguientes campos:

- Nombre completo
- Dirección de correo electrónico
- Cuerpo del mensaje
- Captcha (para evitar que creen múltiples entradas con un script)

El campo de correo electrónico debe validarse para garantizar que tenga el formato correcto. Todos los campos deben ser obligatorios, y el usuario debe recibir un mensaje de error si intenta enviar el formulario sin completarlos.

Para realizar esta operación, deberá utilizar el endpoint correspondiente en la **API**.

3 Actividades y Noticias

Esta sección permite visualizar el listado de noticias (artículos informativos, publicaciones, eventos, etc) cargados desde la aplicación privada.

Las notas deben mostrarse ordenadas por fecha de creación (las más recientes primero) e incluirá solo un resumen de las mismas, pudiendo ver a la nota completa al acceder a un enlace sobre la misma. Los campos resumen a mostrar en el listado serán:

- Fecha de publicación
- Título
- Copete

Para realizar esta funcionalidad deberá utilizar la **API**.

Importante

Consideraciones técnicas

- El prototipo debe ser desarrollado utilizando Python, JavaScript, HTML5, CSS3 y PostgreSQL, y **respetando el modelo en capas MVC**.
- El código deberá escribirse siguiendo las guías de estilo de Python.
- El código Python deberá ser documentado utilizando docstrings.
- **El uso de Jinja como motor de plantillas es obligatorio para la aplicación privada.**
- Se debe utilizar Flask como framework de desarrollo web para la aplicación privada.
- Se deberán realizar **validaciones de los datos de entrada** tanto del lado del cliente como del lado del servidor. *Para las validaciones del lado del servidor se deben realizar en un módulo aparte* que reciba los datos de entrada y devuelva el resultado de las validaciones. En caso de fallar el controlador debe retornar la respuesta indicando el error de validación. Para los campos de texto se deben hacer las validaciones básicas como por ejemplo validar sólo números es un DNI, sólo caracteres válidos para un nombre, etc. Para los campos de varias opciones se debe chequear que la opción ingresada sea una de las opciones válidas.
- Podrán utilizar librerías que faciliten algunas de las tareas que deben realizar en el trabajo como pueden ser: conexión a servicios externos, librerías de parseo, librerías con patrones para buenas prácticas, validaciones de datos, etc. **Pero todos los miembros del equipo deben demostrar en la defensa pleno conocimiento del funcionamiento de estas librerías y una idea de cómo solucionan el problema**.
- Para la implementación del Login **no se podrá utilizar librerías como Flask-Login** dado que consideramos que ocultan implementación que queremos asegurarnos que entiendan en el transcurso de esta materia.
- Para la interacción con la base de datos se deberá utilizar el ORM SQLAlchemy que nos permita además tener una capa de abstracción con la BD.
- Debe tener en cuenta los conceptos de *Semántica Web* proporcionada por HTML5 siempre y cuando sea posible con una correcta utilización de las etiquetas del lenguaje.
- Cada entrega debe ser versionada por medio de git utilizando el sistema de Versionado Semántico para nombrar las distintas etapas de las entregas. Ejemplo: para la etapa 1 utilizar la versión v1.x.x.
- Puede utilizar frameworks de CSS como Bootstrap, Foundation, Bulma o Tailwind CSS, entre otros.
- Todas las vistas deben ser web responsive y visualizarse de forma correcta en distintos dispositivos. Al menos deben contemplar 3 resoluciones distintas:
  - res < 767px (móviles)
  - 768px < res < 1024px (tablets)
  - 1025px < res (portátiles y escritorio)
- Se debe utilizar VueJs (versión 3.5.X) como framework de desarrollo web para la aplicación pública/portal.
- El código deberá estar bien documentado y prolijo.

Consideraciones generales

- La entrega es obligatoria. Todos y todas los/as integrantes deben presentarse a la defensa.
- El/la ayudante a cargo **evaluará el progreso y la participación** de cada integrante mediante las consultas online y el seguimiento mediante GitLab.
- El proyecto podrá ser realizado en grupos de tres o cuatro integrantes (será responsabilidad de los y las estudiantes la conformación de los equipos de trabajo). Todos y todas los/as estudiantes cumplirán con la totalidad de la consigna, sin excepciones.
- Deberán visualizarse los aportes de cada uno/a de los/as integrantes del grupo de trabajo tanto en Git como en la participación de la defensa.
- La defensa será de forma virtual a convenir con el ayudante asignado. En caso de no tener los medios necesarios para realizar la defensa de forma correcta (micrófono y cámara), se deberá realizar la defensa de manera presencial.
- El trabajo será evaluado desde el servidor de la cátedra que cada grupo deberá gestionar mediante Git. **NO se aceptarán entregas que no estén realizadas en tiempo y forma en el servidor provisto por la cátedra**.
- El código en el servidor debe ser funcional y cumplir con las condiciones indicadas.
- Toda funcionalidad que no se haya terminado en la etapa 1 puede completarse en esta etapa.