Proyecto programado #1 – 20%

Bases de datos II – Grupo 2

Semestre II 2020

Proyecto: Formularios Mongo

DESCRIPCION

Se requiere una aplicación (preferiblemente Web) que permita manejar Formularios, diversos formularios, con formatos diferentes, los cuales además, sigan un proceso de aprobación.

El requerimiento técnico que no puede omitir es que debe desarrolla el Backend en Node.js+Express y con base de datos Mongo.

REQUERIMIENTOS

SCHEMAS DE FORMULARIOS

1. Debe tener la opción de crear los formularios, es decir, las plantillas o *schema* en que se basará un grupo de formularios. En esta opción, debe indicarse el nombre del campo, el tipo del campo, si es requerido, y la lista de posibles valores que admite (opcional, algunos no tendrán una lista desplegable). La lista de desplegables

2. Las plantillas o *schemas* en el punto 1, deben ir a una collection de Mongo. Además debe hacer una página o pantalla en la cual se puedan crear, editar o eliminar los *schemas* de formularios.

3. Los schemas sirven para generar los campos de un formulario y por tanto, el formulario completo.

4. Basado en plantillas se habilitan formularios para los usuarios, para diversas situaciones: solicitar vacaciones, solicitar permisos de ausentarse, solicitar préstamos, etc.

APROBACIONES

5. Debe tener un collection y la página o pantalla de mantenimiento donde se defina rutas de aprobación por documento (formulario lleno).

6. Las rutas de aprobación debe tener cuáles usuarios son autores, cuáles son aprobadores y cuál es el tipo de formulario que aplica a esa ruta. Por ejemplo: si el operario A y el operario B son subalternos del Gerente Y, se puede hacer una ruta de aprobación para ambos operarios como autores, y el aprobador es el Gerente Y, esta ruta que aplique al formulario de vacaciones. También podría aplicarse una igual a un formulario de Permisos laborales.
   Adicionalmente, puede aplicar más de una ruta o aprobador, por ejemplo, que apruebe el Gerente Y y el Gerente Z. O bien, que apruebe solo 1 de los dos.
   No olvide de manejar la opción de rechazar, puede aprobar o rechazar el documento.

7. Cuando un usuario crea un documento y le aplica una ruta de aprobación (puede que no le aplique) quedará en un estado de sin aprobar o borrador. Hasta que cumpla con todas las aprobaciones quedará como creado.

8. Tanto el creador como el aprobador deben tener una pantalla o página donde puedan ver sus documentos creados y sus documentos pendientes de aprobar. Desde acá mismo, los aprobadores podrán aprobar o rechazar el documento.

USUARIOS

9. Tendrá que crear colección de usuarios, para mantener los user y password.

10. Debe habilitar un login a la aplicación.

11. Requiere de usuarios para las rutas de aprobación.

12. Debe crear permisos para usuarios, de forma que algunos puedan ser administradores: crean schemas, habilitan formularios, dan acceso a los usuarios a los formularios, crean rutas de aprobación.
    Usuarios no administradores: se les debe asignar a cuáles documentos tiene acceso.

PORTAL DE DOCUMENTOS

13. Los usuarios entran a un portal, donde se les habilitan todos los documentos que pueden crear. Allí seleccionan hacer un nuevo documento y se le presenta el formulario para llenar, con los campos del schema de formulario y las validaciones por campo.
    Llenado el formulario pasa a aprobaciones o queda creado (puede manera un estado)

14. En el portal, el usuario podrá ver todos sus documentos creados y pendientes.

15. En el portal, los administradores podrán crear y actualizar schemas, asignar permisos, crear y actualizar usuarios, etc.

INDICACIONES

a. **Entrega:** 20 de noviembre a las 23:45 al TEC Digital. La revisión será con citas.

b. **Grupos:** la tarea podrá ser realizada en grupos de hasta 4 personas. (1-4)

c. **Tools:**

    a. **Base de datos**: Mongo. Puede ser local o utilizar Atlas.

    b. **NodeJs + Express**

    c. **Todas las transacciones a la base de datos deben ser mediante servicios REST en NodeJs**

    d. El lenguaje o framework para la interfaz gráfica es de libre elección. Para que el ejercicio sea completo, se recomienda que sea un FrontEnd Web. Considere Angular para completar el MEAN Stack (opcional)