Proyecto de Software 2021

Trabajo Integrador (TI)

Todos los años se trata de coordinar el trabajo integrador (TI) de la materia con necesidades y/o demandas de la comunidad. Para la cursada 2021, y acercándonos al 10mo. aniversario de la inundación de la ciudad de La Plata, se trabajará en una aplicación que permita conocer y gestionar zonas con peligro de inundación y las acciones a llevar a cabo en caso de suceder un evento de este estilo.

El 2 de abril de 2013, la ciudad de La Plata sufrió un evento extraordinario que provocó que distintas zonas de la ciudad quedaran inundadas con graves consecuencias.

El TI de la cursada se enfocará en el desarrollo de un prototipo que brinde información a los y las habitantes de la ciudad de La Plata sobre:
.- zonas inundables de la ciudad;
.- puntos de encuentro en caso de una emergencia;
.- recorridos de evacuación.
También se podrán realizar denuncias respecto a alcantarillas tapadas, basurales.

El trabajo se llevará a cabo en tres (3) etapas, en las cuales se deberá entregar un prototipo en funcionamiento en el servidor provisto por la cátedra.
Se implementarán dos aplicaciones que permitirán gestionar y brindar la información mencionada. Por un lado, se desarrollará una aplicación privada que potencialmente proveerá la siguiente funcionalidad:
- Gestión de usuarios del sistema.
- Mantener opciones de configuración del sistema
- Gestión de las zonas inundables.
- Gestión de puntos de encuentro establecidos por el municipio.
- Procesamiento de las denuncias realizadas.

También se desarrollará una aplicación pública, que permitirá visualizar la información de interés para los y las ciudadanos/as. A través de esta aplicación se podrá:
.- Visualizar zonas inundables.
.- Ubicar los puntos de encuentro, que son establecidos por el municipio, pudiendo mostrar los más próximos a la ubicación del usuario. Podemos encontrar puntos de encuentro en el siguiente informe realizado por la Facultad de Ingeniería.
.- Realizar denuncias sobre: alcantarillas tapadas y basurales. También se pueden pensar en otros tipos de denuncias que pueden ser de inteŕes para la aplicación.
.- Visualizar recorrido de evacuación de acuerdo a la ubicación del usuario.

Etapa 1. Aplicación Privada

Fecha de entrega: 15/10 a las 23:59

1.1 Manejo de sesiones

Deberá implementarse un manejo de sesiones adecuado, verificando la sesión y permisos cuando corresponda. Para cada módulo se indicará si requiere autenticación o no y los permisos necesarios.

1.2 Layout

Se deberá implementar el layout de la aplicación que es la base para todas las vistas de la aplicación privada. El resto de las vistas estarán contenidas en este layout sobreescribiendo el contenido central.

La aplicación deberá contar con un menú de navegación que tenga los enlaces a todos los módulos del sistema y esté visible en aquellas vistas del sistema que se consideren necesarias.

Se debe incluir un espacio donde se pueda visualizar la cuenta del usuario con acceso al perfil y link para cerrar la sesión, similar a la Figura 1.

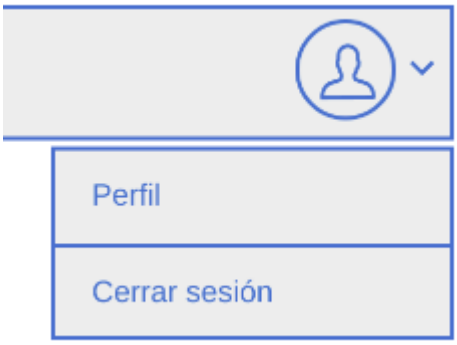

Figura 1. Posible visualización de la cuenta del usuario en el sistema.

También pueden crear un logo para la aplicación y mostrar el mismo en forma coherente en todas las secciones de la aplicación.

1.3 Módulo de usuarios

Desarrollar el **módulo de usuarios** que deberá contemplar **al menos** la siguiente funcionalidad:

- CRUD de usuarios: en esta sección deben validar que no existan dos usuarios con el mismo nombre de usuario, o mismo email, es decir, tanto el nombre de usuario como el email son únicos.

- Se considerarán al menos los siguientes datos para cada usuario: nombre, apellido, email, nombre de usuario, password, activo y roles (Operador/a o Administrador/a).
- Se deben poder realizar búsquedas sobre los usuarios, **al menos** por los siguientes campos:
    - nombre de usuario.
    - activo/bloqueado.

  El resultado de la búsqueda debe estar paginado en base a la configuración del sistema (ver **módulo de configuración**). La paginación deberá realizarse del lado del servidor, es decir, la cantidad de registros retornada debe ser la indicada en el módulo de configuración, por ej. 25 registros por página.
- Activar/Bloquear usuario: un usuario bloqueado no podrá acceder al sistema. Se deberá validar que los únicos usuarios que no puedan ser bloqueados, sean aquellos con el rol **Administrador**.
- Asignar o desasignar roles de un usuario, pueden ser varios. En principio se proponen los siguientes roles: **administrador/a** y **operador/a**.

Para el desarrollo del TPI no será *no será obligatorio desarrollar el CRUD de los roles y los permisos, podrán administrarse desde la base de datos*. Los usuarios, roles y permisos sólo podrán ser administrados por un usuario con rol de **Administrador**.
Considerar que un usuario podrá tener más de un rol, y para cada rol se pueden configurar varios permisos. Los permisos necesarios asociados a cada rol deberán deducirse del enunciado, ante la duda pueden **consultar a su ayudante**.

El nombre de los permisos deberá respetar el patrón **modulo_accion**, por ejemplo, el módulo de gestión de las zonas inundables (ver **módulo de gestión de puntos de encuentro**) deberá contemplar los siguientes permisos:

- punto_encuentro_index: permite acceder al index (listado) del módulo.
- punto_encuentro_new: permite cargar una zona inundable.
- punto_encuentro_destroy: permite borrar una zona inundable.
- punto_encuentro_update: permite actualizar una zona inundable.
- punto_encuentro_show: permite visualizar una zona inundable.

Es importante entender el porqué del uso de esta solución para implementar la autorización y seguir este esquema de la forma correcta. Esto se explicará oportunamente en los horarios de práctica.
La Figura 2 muestra un posible modelo de usuarios, roles y permisos, que pueden utilizar en el trabajo.

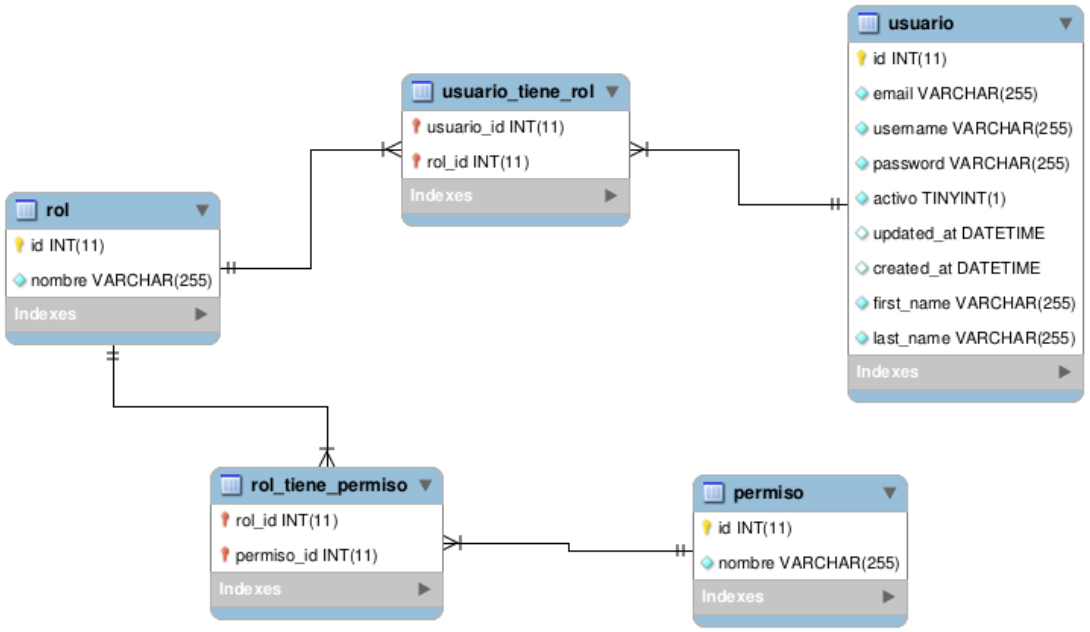

Figura 2. Posible esquema para el manejo de usuarios

1.4 Módulo de configuración

Este módulo permitirá administrar la configuración del sistema, como mínimo deberá contemplar la siguiente configuración:

- Cantidad de elementos por página en los listados del sistema (todos los listados deberán respetar este valor para el paginado).
- Criterio de ordenación por defecto de los listados.
- Paleta de colores de la aplicación privada y pública (deberán definir al menos tres colores para cada aplicación).

La configuración del sistema sólo podrá modificarla un usuario con el rol de **Administrador**. Pueden agregar, en caso de que lo consideren necesario y con acuerdo previo con el ayudante asignado, alguna configuración más para ser administrada por este módulo.

1.5 Módulo de gestión de puntos de encuentro

Este módulo permite a un usuario con el rol de **Operador/a** o **Administrador/a**, gestionar los puntos de encuentro.

c1.5.1 CRUD de puntos de encuentro

El módulo debe brindar **al menos** la siguiente funcionalidad:

- CRUD de puntos de encuentro.

- Se deben listar los campos que permitan ver la información relevante (consultar con el ayudante en caso de alguna duda).
- Se deben poder realizar búsquedas, **al menos** por los siguientes campos:
    - Nombre de punto de encuentro (texto).
    - Estado: publicado o despublicado (select).

  Los resultados también deben estar paginados tomando los parámetros de cantidad de páginas del **módulo de configuración**.

Los datos necesarios para cada punto de encuentro son:

- Nombre*: nombre de la zona inundable (text)
- Dirección*: dirección de la zona inundable (text). Este dato será utilizado para visualizarlo en un mapa.
- Coordenadas[1]: coordenadas geográficas de la zona inundable (text).
- Estado: publicado o despublicado.
- Teléfono: número de teléfono (text).
- Email: dirección de correo electrónico (text).

A continuación se definen los roles necesarios para el resto de las acciones:

- index, show, update, create: **Operador/a, Administrador/a**
- destroy: **Administrador/a**

Consideraciones generales:

- El prototipo debe ser desarrollado utilizando Python, JavaScript, HTML5, CSS3 y MySQL, 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.
- Podrán utilizar librerías que facilitan 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**.

[1] Será necesario tener las coordenadas del eje latitud y longitud para poder ubicar los puntos en el mapa.

- Para la interacción con la base de datos se deberá utilizar un ORM que nos permita además tener una capa de abstracción con la BD.
- Para la aplicación pública se debe utilizar el Framework web VueJS versión 3. Se deberá utilizar la librería vue-router para implementar el ruteo de la aplicación.
- No pueden utilizar un framework/generador de código para el desarrollo de la API de zonas inundables, denuncias, puntos de encuentro y recorridos de evacuación.
- 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.
- 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**.
- 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.
- 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.
- El/la ayudante a cargo **evaluará el progreso y la participación** de cada integrante mediante las consultas online y el seguimiento mediante GitLab.
- Toda vista (**HTML5** y **CSS3**) debe validar contra las especificaciones de la W3C (http://validator.w3.org/ y https://jigsaw.w3.org/css-validator/ respectivamente). En esta oportunidad puede utilizar **Bootstrap** u otro framework similar. En caso de que alguna de las vistas no valide, deberá realizar un breve informe indicando cuales son los errores encontrados (este informe deberá realizarse para la etapa 1).
- Todas las vistas deben cumplir ser web responsive y visualizarse de forma correcta en distintos dispositivos (tener en cuenta para la etapa 2 en adelante). Al menos deben contemplar 3 resoluciones distintas:
    - res < 360
    - 360 < res < 768
    - res > 768
- La entrega es obligatoria. Todos y todas los/as integrantes deben presentarse a la defensa.
- El sistema no debe ser susceptible a SQL Injection, XSS ni CSRF.
- **Importante**:
    - 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.

Información del servidor

Tener en cuenta que el servidor de la cátedra utiliza los siguientes servicios y versiones:

- Servidor de Base de Datos: MariaDB 10.3.31
- Intérprete Python: Python 3.8.10
- Node JS: v14.17.6 (npm 6.14.15)
- Servidor web: Apache 2.4.41 (Ubuntu)